UNIVERSITY OF CALIFORNIA SAN DIEGO

Understanding the Remote Access Trojan malware ecosystem through the lens of the infamous DarkComet RAT

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science (Computer Engineering)

by

Brown Rhodes Farinholt

Committee in charge:

Professor Kirill Levchenko, Chair
Professor Farinaz Koushanfar
Professor Stefan Savage
Professor Alex Snoeren
Professor Geoff Voelker

2019

The Dissertation of Brown Rhodes Farinholt is approved and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2019

DEDICATION

To my parents, for raising me to chase my dreams. I owe this all to you.

EPIGRAPH

The first principle is that you must not fool yourself,
and you are the easiest person to fool.

Richard Feynman

In writing, you must kill all your darlings.

William Faulkner

The truth may be puzzling. It may take some work to grapple with. It may be counterintuitive. It may contradict deeply held prejudices. It may not be consonant with what we desperately want to be true. But our preferences do not determine what's true. We have a method, and that method helps us to reach not absolute truth, only asymptotic approaches to the truth never there, just closer and closer, always finding vast new oceans of undiscovered possibilities. Cleverly designed experiments are the key.

Carl Sagan

TABLE OF CONTENTS

Signature Page iii

Dedication iv

Epigraph v

Table of Contents vi

List of Figures x

List of Tables xiv

Acknowledgements xvii

Vita xx

Abstract of the Dissertation xxi

Chapter 1 Introduction 1

Chapter 2 Remote Access Trojans 5
2.1 A Brief History 5
2.2 DarkComet RAT: A Technical Overview 8
2.2.1 Components 9
2.2.2 Stub Configuration 10
2.2.3 Infection 11
2.2.4 Victim-Controller Handshake 14
2.2.5 Functionality 15
2.2.6 Related Work 15

Chapter 3 HAMELIN: A System for Tracking Remote Access Trojan Controllers 17
3.1 Introduction 17
3.2 Background & Related Work 18
3.2.1 RAT Configuration Extraction 18
3.2.2 Internet-wide Scanning 18
3.2.3 RAT Scanning 19
3.3 RAT-HUNTER: Collection of RAT Binaries 19
3.3.1 Initial DarkComet Binary Collection 19
3.3.2 Secondary DarkComet & njRAT Binary Collection 21
3.4 RAT-HUNTER: Collection of RAT Configurations 22
3.5 RAT-SCAN: Active Detection & Monitoring of RAT Controllers 24
3.5.1 RAT-SCAN Design 24
3.5.2 Scanning Windows: Overview 25
3.5.3 Scanning Window: Honeypot Deployment 25

3.5.4 Scanning Window: Sinkholing 29
3.5.5 Scanning Window: Database Downloading 31
3.6 Limitations 35
3.6.1 ZMap Coverage 35
3.6.2 Sample Unpacking & Configuration Extraction 35
3.6.3 Validating Scanning Results 36
3.6.4 Scanning Result Confidence 36
3.7 Discussion 36
3.7.1 Comparing RAT-SCAN to Shodan & Insikt Group (Recorded Future) . . 36
3.8 Acknowledgements 37

Chapter 4 Understanding Attacker Motivations 39
4.1 Introduction 39
4.2 Background & Related Work 42
4.2.1 Sandboxed Malware Execution 42
4.2.2 Low-Volume Malware Attacks 42
4.3 Methodology 42
4.3.1 Live Operator Monitoring 43
4.3.2 Behavioral Reconstruction 47
4.4 Analysis 48
4.4.1 Operator Behavior Analysis: Overview 48
4.4.2 Sample Execution Schedule 51
4.4.3 Common Actions 52
4.4.4 Data Collection 53
4.4.5 Dropped Files 54
4.4.6 Visiting URLs 55
4.4.7 Command Line Activity 55
4.4.8 Operator Interaction with the User 56
4.4.9 Remote Desktop Sessions 59
4.4.10 Dynamic Language Analysis 61
4.4.11 Operator Time Engagement 63
4.4.12 Operator Final Action 65
4.4.13 Operator Motives 68
4.5 Discussion 69
4.5.1 Honeypot as Tarpit Defense 70
4.5.2 Honeypot as Threat Intelligence Sensor 70
4.5.3 High-Interaction Honeypots: Lessons Learned 71
4.6 Ethical Considerations 72
4.7 Limitations and Biases 74
4.8 Conclusion 77
4.9 Acknowledgements 77

Chapter 5 Measuring Residual Victims of RAT Campaigns 79
5.1 Introduction 80

5.2 Background & Related Work . . . 82
5.2.1 Sinkholing . . . 82
5.2.2 Dynamic DNS . . . 83
5.3 Data Collection & Processing . . . 83
5.3.1 DDNS CLAIMER . . . 86
5.3.2 RAT-HOLE Operation Summary . . . 86
5.4 Analysis . . . 87
5.4.1 Victim Analysis . . . 87
5.4.2 Attacker Campaign Analysis . . . 89
5.5 Discussion . . . 91
5.5.1 Protecting Victims . . . 91
5.6 Ethical Considerations . . . 91
5.7 Limitations . . . 92
5.7.1 RAT Family Scope . . . 92
5.8 Conclusion . . . 92
5.9 Acknowledgements . . . 92

Chapter 6 Quantifying Victim Harm in the RAT Ecosystem . . . 94
6.1 Introduction . . . 94
6.2 Background & Related Word . . . 97
6.2.1 Downloading Victim Databases . . . 97
6.2.2 Hack Pack Sharing . . . 98
6.2.3 Protocol Vulnerabilities in RATs & Other Malware Controllers . . . 98
6.2.4 Estimating Infected Population . . . 99
6.2.5 Data Set Pollution . . . 99
6.3 Data Collection . . . 100
6.3.1 Victim Database Acquisition . . . 100
6.4 Post-Processing . . . 109
6.4.1 Database Attribution . . . 109
6.4.2 Identifying Victim Pollution . . . 115
6.5 Analysis . . . 120
6.5.1 The DarkComet RAT Ecosystem at Scale . . . 124
6.5.2 Quantifying Observed Harm to Victims . . . 126
6.6 Discussion . . . 134
6.6.1 Understanding Victim Harm . . . 134
6.6.2 Campaign Tracking . . . 134
6.6.3 Real Victim Determination . . . 134
6.7 Ethical & Legal Considerations . . . 135
6.8 Limitations . . . 136
6.9 Conclusion . . . 136
6.10 Acknowledgements . . . 137

Chapter 7 Examining the Geographic Relationship between Attackers & Their Victims 138
7.1 VirusTotal Sample Submissions vs. Operators . . . 138

7.2 DDNS Domain Resolutions vs. Victims 140
7.3 Operators vs. Victims 146
7.4 Discussion 150
7.5 Acknowledgements 150

Chapter 8 Conclusion 152
8.1 Dissertation Summary 152
8.2 Future Directions 153

Bibliography 155

LIST OF FIGURES

Figure 1.1. The DarkComet RAT control panel [84]. An attacker uses this interface to manage and issue commands to victims individually. 2

Figure 2.1. The control panel for NetBus [98], the original RAT interface. 6

Figure 2.2. Valeros [133] provides a chronological view of the 300 most popular RAT families. 7

Figure 2.3. A Google search for any RAT family (e.g. DarkComet) links to YouTube tutorials on its deployment and usage, often "disguised" as educational or ethical hacking tutorials. 8

Figure 2.4. DarkComet's builder panel, used to create stubs. The builder offers a number of customization options, such as persistence and cloaking. 9

Figure 2.5. RAT infection process. 11

Figure 2.6. Network diagram of the DarkComet handshake. 12

Figure 2.7. An example DarkComet *Victim Info* packet with individual components extracted and labelled. Not from a real infection. 12

Figure 2.8. The denial-of-service options offered by DarkComet [76]. 16

Figure 2.9. DarkComet's so-called "Fun Manager," which offers functions for harassing the victim [76]. 16

Figure 3.1. A timeline of the measurements performed by HAMELIN; particularly, RAT operator interaction honeypotting, RAT-associated DDNS domain sinkholing, and RAT victim database downloading. 18

Figure 3.2. Scanning breakdowns by day and hour. 27

Figure 3.3. Number of DarkComet controller nodes online each hour, binned by hour. We detected 9,877 unique controllers in total. Counts are from an Internet-wide scan for active DarkComet controllers, and include controllers for which we do not have a RAT sample. 28

Figure 3.4. Total DarkComet controllers online per week. 32

Figure 3.5. Total number of DarkComet controllers online per day. 32

Figure 3.6. Total number of *new* DarkComet controllers per day. 33

Figure 3.7. Average number of DarkComet controllers online per hour of the day. 33

Figure 4.1. Our data collection and processing workflow. Each box displays the number of corresponding entries after each step. 43

Figure 4.2. Visual summary of complete experiment workflow. 44

Figure 4.3. A CDF showing the relative age of our samples, where age is the number of days between submission to VirusTotal and execution in our sandbox. . 45

Figure 4.4. Composite flowchart of prevalent operator behaviors and sequences broken down by RAT interaction phase and category. 49

Figure 4.5. Malware sample submissions by day and by hour. Colors from colorbrewer2.org [24]. 51

Figure 4.6. Comparison of the time spent in the honeypots by operators in the first and second experiments, in minutes. Operators are distinguished by their use of RDP, as per Table 4.2. 63

Figure 4.7. Comparison of the time spent between honeypot personas (in only the second experiment), in minutes. Only sessions with active RDP are reported, as these are representative of the whole. Operators are distinguised by the honeypot persona with which they interacted. 64

Figure 5.1. *Intelligence pollution* obfuscates the stakeholders in the RAT ecosystem. . 80

Figure 5.2. The partial contents of an email from the now-defunct DtDNS dynamic DNS service provider (formerly dtdns.com) announcing cessation of service due to continued abuse of their service by customers. 84

Figure 5.3. The major components of our operation and their interactions with the subjects of our study. 85

Figure 5.4. Timeline of data collection phases of our study: Binary Acquisition, Controller Domain Resolution, Scanning for Controllers, Domain Claiming, and Sinkholing. 85

Figure 5.5. PDF showing the probability that a domain we sinkholed would yield a victim connection *N* days after its most recent registration by another party. 88

Figure 5.6. CDF showing the number of victims (by fingerprint) received by a given domain. This plot only includes the 975 domains which yielded victim connections. 90

Figure 6.1. Illustration of our data collection methodology, from the sourcing of DarkComet configurations from IoC feeds to the downloading of databases from detected hosts. 101

Figure 6.2. Network diagram of the DarkComet file download procedure. The highlighted command is omittable, allowing for arbitrary file downloads. Descriptions of the packets in this diagram are available in prior work [28, 11]. 102

Figure 6.3. Window displayed during file download. 104

Figure 6.4. The schemas of the tables of importance in the DarkComet database. 106

Figure 6.5. Illustration of the database inheritance tree reconstruction algorithm. 111

Figure 6.6. Fragment of the reconstructed DarkComet database phylogenetic tree, after the tree creation stage in Figure 6.5b. 112

Figure 6.7. Fragment of the reconstructed DarkComet database inheritance tree. 113

Figure 6.8. Total number of databases downloaded per controller. 121

Figure 6.9. Total number of unique IP addresses per controller. 121

Figure 6.10. Daily new victim infection rate per controller. 122

Figure 6.11. Total new victims per controller during observation period. 122

Figure 6.12. Total number of victims per controller. 122

Figure 6.13. Days from victim first keylog to last keylog. 123

Figure 6.14. Victim keystrokes per day. 123

Figure 6.15. CDF of victim infection duration (n=4,687). 125

Figure 6.16. CDF of controller age (n=334). 126

Figure 7.1. Correlation between the geolocated countries of the VirusTotal uploaders and those of the controllers accessing our honeypots. Countries are sorted by decreasing number of controllers. 139

Figure 7.2. Relational matrix comparing geolocations of actively-probed controller IP addresses to received victim IP addresses, per sinkholed domain. Proxy IP addresses are filtered. 143

Figure 7.3. Relational matrix comparing geolocations of historic controller IP addresses to received victim IP addresses, per sinkholed domain. Proxy IP addresses are filtered. 145

Figure 7.4. Number of victims for each combination of controller and victim country. Rows denote controller countries and columns victim countries, based on geo-located IP address (excluding VPN providers). 146

Figure 7.5. Number of controllers for each combination of controller and victim country, with each controller contributing 1 count. Rows denote controller countries and columns victim countries, based on geo-located IP address (excluding VPN providers). 149

LIST OF TABLES

Table 2.1. Some of the information in the DarkComet stub's embedded configuration. 10

Table 2.2. Fields of information sent in DarkComet `infoes` handshake packet. *PII* indicates whether we consider the field to be potential PII of the victim. . . . 13

Table 3.1. Languages used in DarkComet stub configurations for each unique and alphabetic domain, campaign name, and submitted filename (sanitized for more accurate detection). *Other* is any other spoken or written language. . . 20

Table 3.2. Counts of RAT samples downloaded, both total and unique, by family. *Other* are RAT samples that matched our YARA signatures incorrectly. *Failed Decoding* are samples from which configurations could not be extracted. . . 21

Table 3.3. Breakdown of C&C domains in our RAT sample population by Dynamic DNS provider. *Unknown* encompasses all domains unrelated to a known DDNS provider. 22

Table 3.4. Sources of DarkComet sample configurations, including resultant discovered DarkComet hosts and downloaded databases. *After resolution of domains. 23

Table 3.5. Percentages are reported as percentage of the total number of controllers monitored (n=9,877). The sum of the controllers in *Total* exceeds n, as some controllers were discovered by multiple sources. Likewise, the sum of the controllers in *Unique* falls short of n, for the same reason. 25

Table 3.6. Countries of the IP addresses of a) the global population of scanned DarkComet controllers, and b) the controllers to which our live trials connected, as resolved by MaxMind's GeoLiteCity database [90]. Addresses without resolution are omitted. 26

Table 3.7. User-types of the IP addresses of a) the global population of scanned DarkComet controllers, and b) the controllers to which our live trials connected, as resolved by MaxMind's GeoIP2 Insight service [89]. Addresses without resolution are omitted. 29

Table 3.8. Monitored controllers and collected samples that operate on the standard DarkComet port (1604) vs. non-standard ports. 29

Table 3.9. Breakdown of RAT controllers detected on IP addresses responsive to RATSCAN. Some IP addresses hosted multiple types of RAT controller. 30

Table 3.10. VPN providers for all hosts running DarkComet controller software, and those from which we downloaded the victim database. 34

Table 3.11. Dynamic DNS providers for all hosts running DarkComet controller software, and those from which we downloaded the victim database. *Some hosts use more than one provider.* 35

Table 4.1. Summary statistics of the two experiments in the dataset. 49

Table 4.2. Categories of network signatures used in the experiment. 50

Table 4.3. Results of monitoring access attempts to the online accounts in our honeypots. 54

Table 4.4. Motivations of all communications received from operators during live trials. 57

Table 4.5. Common patterns of behavior exhibited by operators employing active remote desktop sessions. 62

Table 4.6. Languages of metadata obtained during live trials, including chat messages, remote desktop keystrokes, and filenames of dropped files. *Other* is any other spoken or written language. 62

Table 4.7. Last operator actions before session termination for trials with manual interaction. Actions that appear in less than 1% of trials are omitted. 66

Table 4.8. Last operator actions before uninstalling malware for trials with manual interaction. Actions that appear in less than 1% of trials are omitted. 66

Table 5.1. Summary of connections received by RAT-HOLE [121], grouped by peer type, fingerprint (an internal representation of the connection), source IP address, ASN, and country. 87

Table 6.1. Database download failure reasons and occurrence. 104

Table 6.2. Download metrics. 104

Table 6.3. The schemas of the tables of importance in the DarkComet database. 105

Table 6.4. Sample of information in the DarkComet `INI` file. 108

Table 6.5. Records and UUIDs filtered by our anomaly detection logic. Many records exhibit more than one anomaly. 117

Table 6.6. Victim operating systems. 125

Table 6.7. Keystrokes captured and hours monitored, shown in aggregate and per victim. *Clipboard* refers to capturing text copied to the clipboard. 127

Table 6.8. Regular expressions and categories against which we compare victim active windows from `dc_keyloggers`. **DarkComet chat is handled as a special case, lest `chat` match all chat windows. 128

Table 6.9. Victim keystrokes for specific online accounts. 129

Table 6.10. Controller group suspected motivations. 132

Table 6.11. Controller group suspected languages. 132

Table 6.12. Example group names and translations from one French-speaking operator. 133

Table 7.1. Geolocations of historic controller IP addresses based on DNS history. 141

Table 7.2. Geolocations of probed controller IP addresses. 142

Table 7.3. Geolocations of victim IP addresses. 142

Table 7.4. Sample of controller locations and victim counts. Controller footprint is the total number of victims controlled by all controllers from a given country. . 147

ACKNOWLEDGEMENTS

I am indebted to my adviser, Kirill Levchenko, whose constant encouragement and support made this all possible. Thank you for showing me how to presevere through any setbacks, and instilling in me the principles of critical thinking. I am a vastly more confident researcher having worked with you these past years.

I am forever grateful for my wonderful co-authors collaborators, and professors, without whom none of the works here would have been possible: Stefan Savage, Geoff Voelker, Alex Snoeren, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Kevin Yin, Stevens Le Blond, Damon McCoy, Sam Crow, Brian Johannesmeyer, Karl Koscher, Steve Checkoway, Aaron Schulman, Devin Lundberg, Ryan Mast. And to Cindy Moore, for your tireless efforts in supporting my research projects, and for always wishing me luck storming the castle.

To my friends, here and home, who have kept me sane: Charlie Blanchard, Nima Nikzad, Brittni Reveles, and Kelly Tynan, of House Archer; Brad Meredith and Carl Cross, of North Clemson; K.C. Cameron and Ty Wilson; Sam Wasmundt, Gary Johnston, John Sarracino, Nate Speidel, Michael Walter, Garrett Rodrigues, and the rest of the undefeated CSE flag football team; Matt Der, Ben Whitman, Dimo Bounov, Clay and Claire Stiles, Alex Brown, Travis Hamblen, Thomas Shockley, Edward Rives, Turner and Dani Blake, Andrew Johnson, Briggs Cocke, and Capt. Robert Allen. Special thanks to Louis Dekoven, my companion through the endgame; there are many more riptides on our horizon. Special thanks to Edward Sullivan; I would not have been on this journey without your record collection and your selfless help since our Clemson days. Special thanks to Porter; this was really all for you.

To my team and peers at Lastline; it has been a pleasure working with you these past two years: Clemens Kolbitsch, Adam James, Dario Simoes Fernandes Filho, Haofan Shi, Danny Knight, Anne Trueblood, Aditya Mhamunkar, and Cristina Polini.

To Jamie Williams and Bill Lasser, for giving the opportunity that would set this all in motion, and to the Clemson National Scholars 2009 cohort (Anna Merryman, Chris Covey, Liz Johnson, Matt Kofoed, Nadine Luedicke, and Taylor Wells) for all the great memories.

To my brothers, Max and Sinclair, for teaching me humility and inspiring me to push beyond my confort zone. I am so proud of you both. To Janie and Katie O'Connor, for your constant friendship and visits to San Diego. And to my godparents Susie and Rick, Amelia, Ella, and Brody, for all your ever-present support and encouragement.

Finally, I want to thank Mary Byrnes; for all the support on late nights when I was in the lab; for all the cross-country flights and days spent traveling; for our journey through Richmond, New Orleans, Washington, Portland, and San Diego; for your unbounded patience and tireless love as I pursued this dream.

Chapter 3, in part, is a reprint of the material as it appears in Proceedings of the 38th IEEE Symposium on Security and Privacy 2017. Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blond, Damon McCoy, Kirill Levchenko, 2017. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material as it appears in Proceedings of the 27th USENIX Security Symposium 2018. Mohammad Rezaeirad, Brown Farinholt, Hitesh Dharmdasani, Paul Pearce, Kirill Levchenko, Damon McCoy, 2018. The dissertation author was the primary investigator and author of the components of the paper presented in this dissertation.

Chapter 3, in part, has been submitted for publication of the material as it may appear in Proceedings of the 26th ACM Conference on Computer and Communications Security, 2019. Brown Farinholt, Mohammad Rezaeirad, Damon McCoy, Kirill Levchenko, 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the material as it appears in Proceedings of the 38th IEEE Symposium on Security and Privacy 2017. Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blond, Damon McCoy, Kirill Levchenko, 2017. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in part, is a reprint of the material as it appears in Proceedings of the 27th USENIX Security Symposium 2018. Mohammad Rezaeirad, Brown Farinholt, Hitesh Dharmdasani, Paul Pearce, Kirill Levchenko, Damon McCoy, 2018. The dissertation author was the

primary investigator and author of the components of the paper presented in this dissertation.

Chapter 6, in part, has been submitted for publication of the material as it may appear in Proceedings of the 26th ACM Conference on Computer and Communications Security, 2019. Brown Farinholt, Mohammad Rezaeirad, Damon McCoy, Kirill Levchenko, 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 7, in part, is a reprint of the material as it appears in Proceedings of the 38th IEEE Symposium on Security and Privacy 2017. Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blond, Damon McCoy, Kirill Levchenko, 2017. The dissertation author was the primary investigator and author of this paper.

Chapter 7, in part, is a reprint of the material as it appears in Proceedings of the 27th USENIX Security Symposium 2018. Mohammad Rezaeirad, Brown Farinholt, Hitesh Dharmdasani, Paul Pearce, Kirill Levchenko, Damon McCoy, 2018. The dissertation author was the primary investigator and author of the components of the paper presented in this dissertation.

Chapter 7, in part, has been submitted for publication of the material as it may appear in Proceedings of the 26th ACM Conference on Computer and Communications Security, 2019. Brown Farinholt, Mohammad Rezaeirad, Damon McCoy, Kirill Levchenko, 2019. The dissertation author was the primary investigator and author of this paper.

VITA

2013	Bachelor of Science, Clemson University
2013–2019	Research Assistant, University of California, San Diego
2016	Master of Science, University of California, San Diego
2019	Doctor of Philosophy, University of California, San Diego

PUBLICATIONS

"A Software-Reconfigurable Federated Avionics Testbed," Sam Crow, Brown Farinholt, Brian Johannesmeyer, Karl Koscher, Stephen Checkoway, Stefan Savage, Aaron Schulman, Alex C. Snoeren, Kirill Levchenko. In submission, 2019.

"Dark Matter: Uncovering the DarkComet RAT Ecosystem," Brown Farinholt, Mohammad Rezaeirad, Damon McCoy, Kirill Levchenko. In submission, 2019.

"Schrödinger's RAT: Profiling the Stakeholders in the Remote Access Trojan Ecosystem," Mohammad Rezaeirad, Brown Farinholt, Hitesh Dharmdasani, Paul Pearce, Kirill Levchenko, Damon McCoy. In Proceedings of the 27th USENIX Security Symposium (USENIX), Baltimore, Maryland, August 2018.

"To Catch a Ratter: Monitoring the Behavior of Amateur DarkComet RAT Operators in the Wild," Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blond, Damon McCoy, Kirill Levchenko. In Proceedings of the 38th IEEE Symposium on Security and Privacy (Oakland), San Jose, California, May 2017.

"On The Security of Mobile Cockpit Information Systems," Devin Lundberg, Brown Farinholt, Edward Sullivan, Ryan Mast, Stephen Checkoway, Stefan Savage, Alex C. Snoeren, Kirill Levchenko. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), Scottsdale, Arizona, November 2014.

ABSTRACT OF THE DISSERTATION

Understanding the Remote Access Trojan malware ecosystem through the lens of the infamous DarkComet RAT

by

Brown Rhodes Farinholt

Doctor of Philosophy in Computer Science (Computer Engineering)

University of California San Diego, 2019

Professor Kirill Levchenko, Chair

The value of traditional malware is predicated on the mass infection of victim devices. Botnets engaging in spamming, click fraud, and the like directly derive monetary value from each new infection. While costly in aggregate, each individual victim's loss is attenuated by an attacker's ability to extract value from such victims at scale. Remote access trojans (RATs) are conversely predicated on the unique value of each individual infection. While most traditional malware infections are automated or controlled at scale, RATs require hands-on operator interaction with each compromised host in exchange for flexible and near-comprehensive control over the victim. The use of off-the-shelf RATs like DarkComet to perpetrate sextortion

and cyber-stalking, voyeurism, and, in rare cases, targeted state actor attacks, has received considerable attention in the media and from security vendors. Despite this, RAT usage and impact have not been investigated at scale in the same manner as traditional botnet malware.

Understanding the scale and nature of the criminal usage of RAT malware is critical to devising effective deterrents and mitigating the harm inflicted on its victims; however, measuring the RAT ecosystem presents challenges not inherent to measuring other types of malware infection. The victims of RAT infections are difficult to identify; as a rule, they do not typically participate in large-scale, noisy behaviors like denial-of-service, spamming, or click fraud. Likewise, RAT backdoors are often targeted rather than broadly distributed, making their command-and-control servers harder to discover. And finally, understanding the motivations of RAT operators is comparatively onerous. Whereas most malware either has a specific purpose (e.g., ransomware), or issues commands to an entire botnet at once (e.g., denial-of-service), RAT infections are individually, manually controlled. In this thesis, I address these challenges while investigating the ecosystems of two popular, commodity-grade RATs, hitherto unexplored at scale. Modifying well-established security techniques like honeypotting, Internet-wide scanning, and domain sinkholing, I develop and deploy tools for measuring and understanding the participants in these ecosystems - the attackers, their motivations, the infrastructure they use, and their victims.

Chapter 1

Introduction

The value of traditional malware is predicated on the mass infection of victim devices. Botnets engaging in spamming [64, 74, 80], click fraud [106], or cryptomining [107, 39, 71] directly derive monetary value from each new infection, as do ransomware extortion campaigns [68, 60], phishing campaigns [30, 58], and banking trojans [40]. Denial-of-service botnets [7, 70] like booters [67, 66] require a critical mass of victims to function at all. While costly in aggregate, each individual victim's loss is attenuated by an attacker's ability to extract value from such victims at scale.

Remote access trojans (RATs), an emerging class of malware, are conversely predicated on the unique value of each individual infection. A RAT is a type of malware that gives a remote attacker total interactive access to a victim machine. Most RATs allow an attacker to capture audio and video from an attached webcam and microphone, log keyboard input, browse files on the machine, edit the machine's Windows registry, and so on. Figure 1.1 shows the control panel of DarkComet, a popular RAT, in the midst of managing a victim. While most traditional malware infections are automated or controlled at scale, RATs require hands-on operator interaction with *each* compromised host in exchange for flexible and near-comprehensive control over the victim. Using RAT malware, an attacker can scour through the victim's file system and private data, spy on the victim though the webcam and microphone, or harass the victim using the computer's speakers and user interface. As such, they are the tool of choice for targeted or personal attacks.

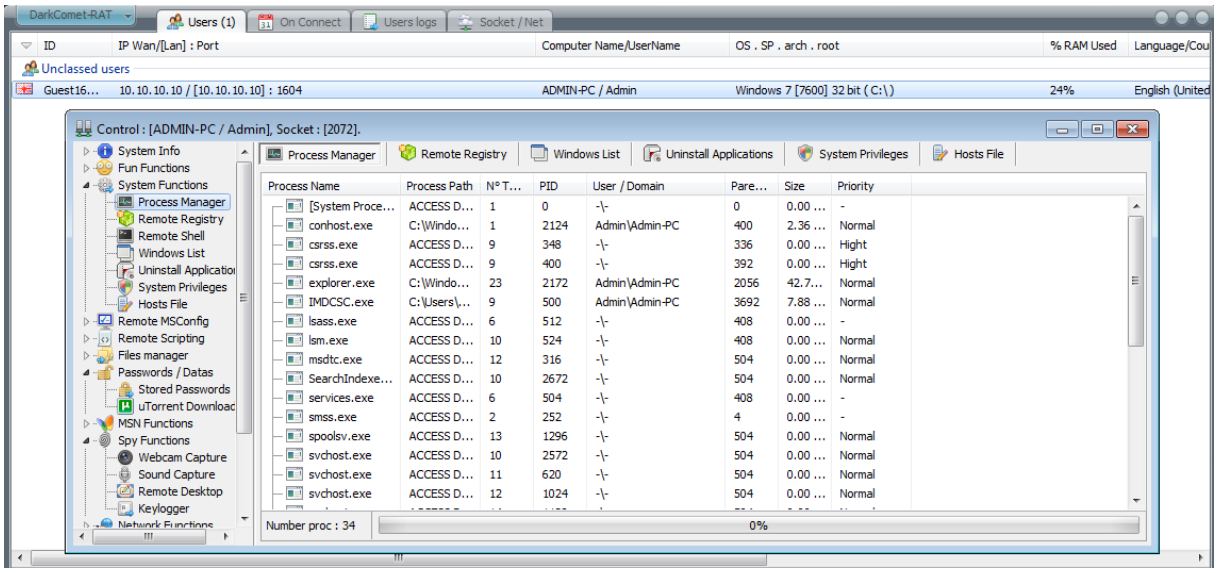


Figure 1.1. The DarkComet RAT control panel [84]. An attacker uses this interface to manage and issue commands to victims individually.

The use of off-the-shelf RATs like DarkComet to perpetrate sextortion and cyber-stalking [29, 4], voyeurism [31], and, in rare cases, targeted state actor attacks [85, 134, 42] and trade secret theft [144, 78, 77], has received considerable attention in the media and from security vendors. Epitomizing this spectrum, DarkComet in particular achieved infamy for its usage by a sextortionist who targeted Miss Teen USA [5, 10] and by the Syrian government to monitor political dissidents during the Syrian Civil War [128, 92, 48, 112]. Unsurprisingly, in recent years law enforcement agencies have begun to arrest the authors [72, 118, 73] and criminal users [45, 120, 20, 104, 83, 97] of RATs internationally. Despite this, RAT usage and impact have not been investigated *at scale* in the same manner as traditional botnet malware.

Understanding the scale and nature of the criminal usage of RAT malware is critical to devising effective deterrents and mitigating the harm inflicted on its victims; however, measuring the RAT ecosystem presents challenges not inherent to measuring other types of malware infection. The victims of RAT infections are difficult to identify; as a rule, they do not typically participate in large-scale, noisy behaviors like denial-of-service, spamming, or click fraud. Likewise, RAT backdoors are often targeted rather than broadly distributed, making their

command-and-control servers harder to discover. And finally, understanding the motivations of RAT operators is comparatively onerous. Whereas most malware either has a specific purpose (e.g. ransomware), or issues commands to an entire botnet at once (e.g. denial-of-service), RAT infections are individually, manually controlled.

In this thesis, I address these challenges while investigating the ecosystems of two popular, commodity-grade RATs, hitherto unexplored at scale. Modifying well-established security techniques like honeypotting, Internet-wide scanning, and domain sinkholing, I develop and deploy tools for measuring and understanding the participants in these ecosystems - the attackers, their motivations, the infrastructure they use, and their victims.

I describe the design and implementation of a modular system, called HAMELIN, for tracking and measuring DarkComet and njRAT command-and-control servers. This system follows multiple threat intelligence feeds, hunting for RAT samples, configurations, and other indicators of compromise. It engages in the constant, active probing of RAT command-and-control servers through indiscriminate, Internet-wide scanning as well as targeted scanning based on the IoCs obtained from the feeds it follows. Further, it tracks and compiles passive DNS information for all obtained domain names associated with RAT activity. I use this corpus of information to estimate the size of these RATs' user bases, as well as to identify the infrastructure commonly used by RAT operators.

As aforementioned, measuring the victim populations of RAT campaigns is challenging. I expound on two studies that overcome this issue using novel vantage points. In the first study, I modify HAMELIN to poach dynamic DNS (DDNS) subdomains previously been used by DarkComet and njRAT operators, directing them to a sinkhole designed by a collaborator, Mohammad Rezaeirad.[1] I use the results of this domain takeover and sinkholing operation to provide a lower bound on the victim populations of these two RATs. In the second study, I modify HAMELIN's targeted scanning functionality to download from DarkComet command-

[1]My collaborator, Mohammad Rezaeirad, and I collaborated on all the work presented in this thesis. I only present on my contributions to this joint work; however, I do mention and attribute his contributions as necessary for context.

and-control servers a database of all victims ever infected, taking advantage of a blind file retrieval vulnerability built into DarkComet's command protocol. I use this database to quantify victim harm in the DarkComet ecosystem. In both studies, I examine the geographic relationship between RAT users and their victims.

Furthermore, determining the motivations behind individual RAT campaigns is an arduous process. I detail a study in which I modify HAMELIN to submit real DarkComet samples obtained from the malware repository VirusTotal to a high-fidelity honeyfarm of realistic virtual machines, so as to capture live operator interactions with supposed victims. Each honeypot records all subsequent operator activity, particularly through network traffic captures and desktop screenshots. Using a DarkComet network traffic decoder written by a collaborator, I decode and analyze live RAT operator attacks, command by command, and use the results of this study to understand operator motivations.

The studies described in this dissertation focus on two commodity RATs, DarkComet and njRAT. Despite this narrow scope, the majority of the measurement techniques I describe are applicable to many of the hundreds of distinct RAT families [133]. Further, I argue that both DarkComet and njRAT are indicative of the overall RAT ecosystem, and that the lessons and findings I expound upon are likewise generalizable.

This disseration is structured as follows. Chapter 2 provides the necessary background material on RATs. Chapter 3 describes the design of a system for tracking RAT command-and-control servers. Chapter 4 illustrates the use of a honeyfarm to determine RAT attacker motivations. Chapters 5 and 6 describe the use of two techniques for measuring the victim populations of these RATs, the latter of which quantifies the harm done to the victim populations under observation. Chapter 7 explores the geographic relationship between RAT victims and their attackers. Chapter 8 concludes the dissertation and muses on future research directions.

Chapter 2

Remote Access Trojans

2.1 A Brief History

In the spring of 1998, a Swedish programmer named Carl-Fredrik Neikter released NetBus, a program for remote administration of personal computers over a network. While remote administration tools had existed for at least a decade prior [133], Neikter's program was distinct in that it was intended to be used as a backdoor into unsuspecting users' computers, ostensibly for mischief.[1] Later that summer, the hacking group known as the Cult of the Dead Cow debuted a program named Back Orifice at DEF CON 6 [27], embodying comparable functionality to NetBus and with the same intended use. These two programs became the first widely distributed *remote access trojans*, nearly identical to legitimate remote administration tools but with additional trappings to faciliate stealthy behavior, like the ability to hide the program from the user, or to persist after attempted uninstallation.

In response to the release of Back Orifice, a spokesperson for Microsoft dismissed the potential dangers of such tools, stating, "This is not a tool we should take seriously, or our customers should take seriously." [122] Within the year, NetBus had been used to plant child pornography on the personal computer of a law researcher at Lund University in Sweden [102]. By the year 2000, Back Orifice 2K was being deployed in denial-of-service attack chains [69] and NetBus 2.0 Pro was actively being marketed and sold as a "spy tool" [145].

[1] Indeed, NetBus translates from Swedish to "NetPrank."

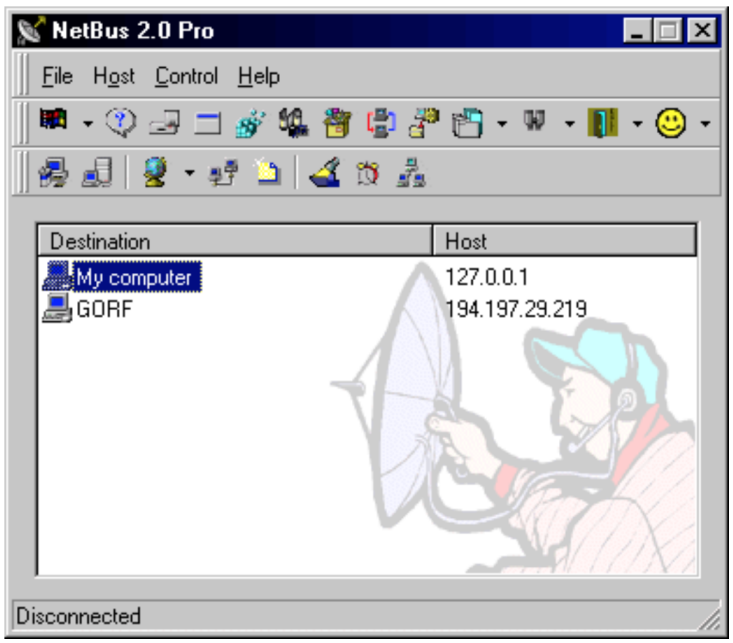


Figure 2.1. The control panel for NetBus [98], the original RAT interface.

Two decades after the release of NetBus, RATs have become a ubiquitous form of malware. There are over 4,000 variants, or families, of RAT malware in existence [133]; Figure 2.2 shows 300 of the most prominent ones. Most of these RATs are freely available for download from forums like HackForums [56] or on GitHub [111]. Some more sophisticated or actively developed RATs like WebMonitor [139] are sold by their authors, often marketed as legitimate remote administration tools while being advertised in hacking forums. Despite the wide-ranging functionality they offer, these RATs are designed to be easily wielded by non-technical users; indeed, there are thousands of instructional videos available on YouTube [31] detailing their deployment and usage (see Figure 2.3). In fact, RATs are so easy to obtain and use maliciously that the United Nations Office on Drugs and Crime considers RAT malware to be a popular entryway into cyber-criminality [83, 97].

By design, RATs occupy a particularly disreputable niche with regards to its use in cyber-criminality. Between the flexibility and control they offer over victims and the required manual operation of each infection, they are the tool of choice for targeted or personal attackers, like voyeurs, (s)extortionists, and their ilk. Fittingly, Symantec even categorizes RATs as

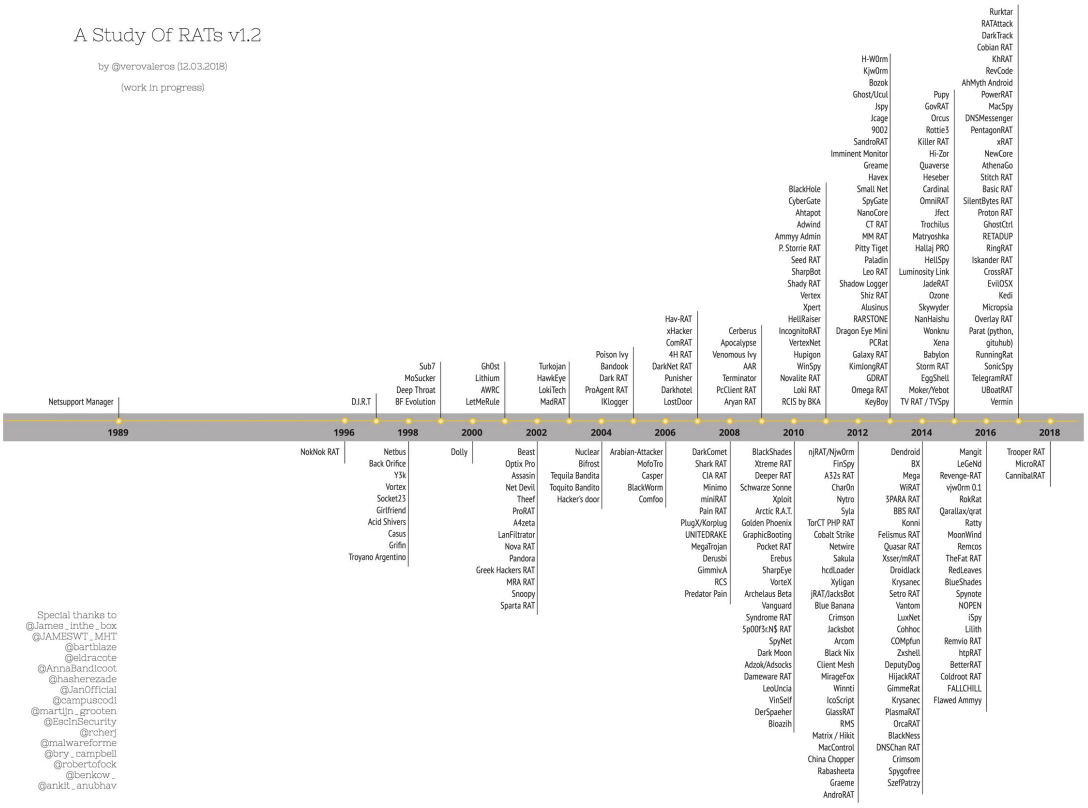


Figure 2.2. Valeros [133] provides a chronological view of the 300 most popular RAT families.

creepware [131]. The first widely-publicized incident involving RAT malware occured in 2013, when a high school classmate of the current Miss Teen USA infected her with DarkComet and Blackshades, two popular RATs, and subsequently attempted to sexually extort her [5, 10, 29, 4]. Her attacker was apprehended by the FBI, who found evidence of hundreds of other victims. Unfortunately, this appears to be the case with many such RAT infections; most victims never realize they are infected [20], or when they do, lack a proper course for remediation [31].

The functionality offered by RATs also makes them attractive for espionage. Surveillance companies like HackingTeam offer professional solutions for offensive monitoring of targets; however, state actors are also known to deploy commodity RATs against targets of surveillance and espionage. For instance, during the Syrian Civil War, the Syrian government disguised DarkComet as an encrypted chat tool and distributed it surreptitiously to dissidents and protestors

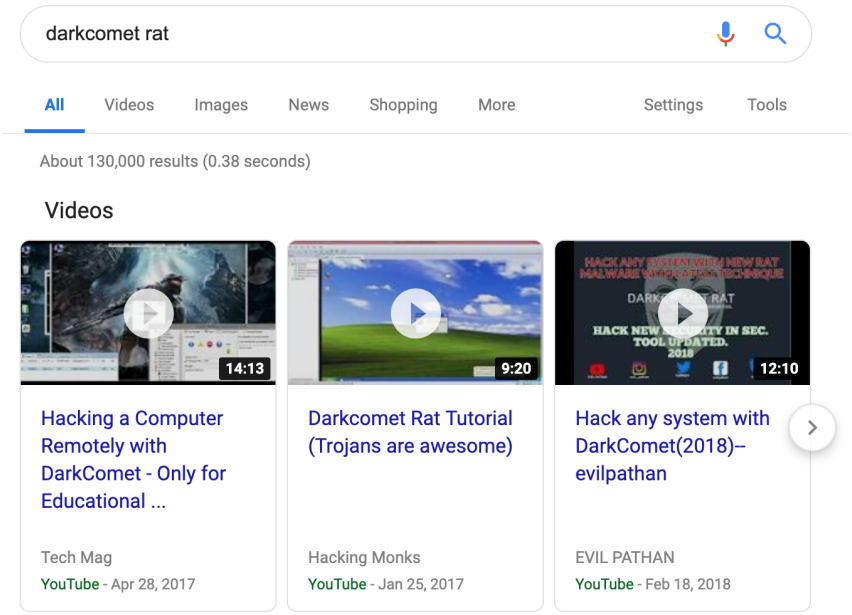


Figure 2.3. A Google search for any RAT family (e.g. DarkComet) links to YouTube tutorials on its deployment and usage, often "disguised" as educational or ethical hacking tutorials.

to monitor [85, 128, 92, 48, 112].

Though a fairly recent development, law enforcement has begun to take seriously the illegal use of RATs. Since 2012, the authors of Nanocore [109], Blackshades [118], LuminosityLink [72], and OrcusRAT [73] have been arrested or subjected to criminal investigation. In 2014, hundreds of users of Blackshades were arrested in an international take-down operation [45, 120, 97]. Targeted investigations have apprehended indivduals like Miss Teen USA's attacker, or a Ukrainian man with almost ten thousand victims at the time of his arrest [20], or a Netherlands teen with access to thousands of women's webcams [41]. Despite this, it is suspected that these operations have had a negligible impact on illegal RAT usage [83, 119].

2.2 DarkComet RAT: A Technical Overview

DarkComet is the quintessential RAT, popular for its functionality, freely available for download online, and supported by hacking forum communities and a plethora of tutorial videos on YouTube [31]. It has been used broadly since 2011 by cybercriminals for sextortion [4],

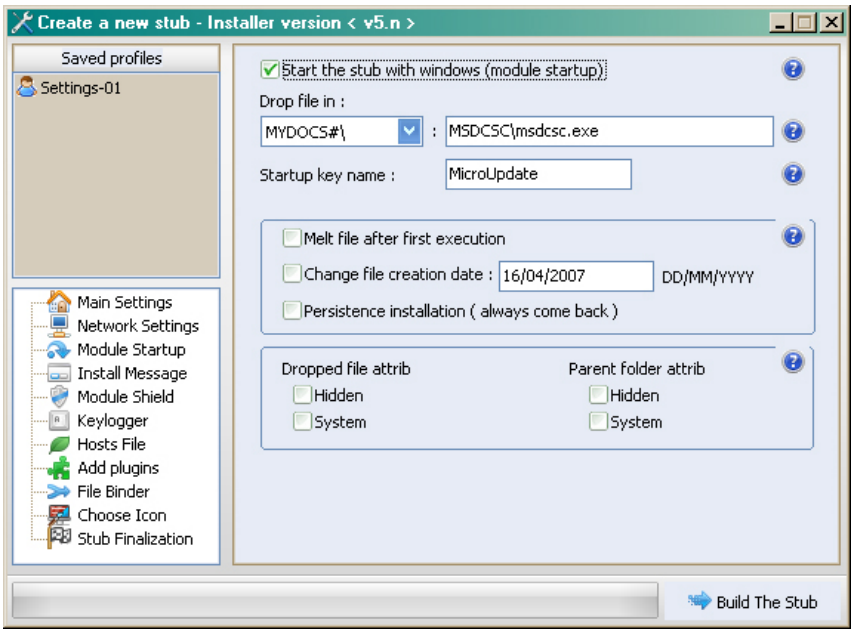


Figure 2.4. DarkComet's builder panel, used to create stubs. The builder offers a number of customization options, such as persistence and cloaking.

voyeurism [31] and, in rare cases, nation state attacks [85, 42] and theft of trade secrets [78, 77]. Marczak *et al.* [85] provide a particularly detailed examination of DarkComet's usage against dissidents during the Syrian Civil War. The studies in this dissertation focus primarily on DarkComet. As such, in this section I use DarkComet as an example to provide the necessary technical background on RAT operation.

2.2.1 Components

A typical RAT software package consists of two components: a *builder* program and a *controller* program. These components can be downloaded freely on hacking forums like HackForums as part of "hack packs," or bundles of hacking software. At the start of a malware campaign, the attacker uses the builder program to create a *stub* for installation on a victim's computer. Figure 2.4 shows DarkComet's builder panel. The stub contains the code that will run on the victim's computer with parameters such as the host name of the command-and-control server to contact upon infection. During the campaign, the attacker runs the controller software on the command-and-control server to interact with the victims. In most cases (e.g., for

DarkComet), the controller provides a graphical user interface and runs directly on the attacker's computer. The attacker, also called the RAT *operator*, interacts with the victim via the controller interface. We use the following terminology throughout this paper:

- **Operator:** Person using DarkComet to control a victim's computer.
- **Victim:** Person whose computer is infected by the DarkComet stub and thus controlled by an operator.
- **Controller:** DarkComet software used to control victim's computer. Also the host on which it is running.
- **Stub:** Malware on the victim's computer that communicates with controller, giving operator control over victim's computer.

2.2.2 Stub Configuration

Table 2.1. Some of the information in the DarkComet stub's embedded configuration.

Field	Description
EDTDATE	Edit timestamp to assign to stub file
EDTPATH	Where to install the stub
FTPHOST	FTP hostname (for remote keylogging)
FTPPASS	FTP password
FTPPORT	FTP port
FTPROOT	FTP root
FTPSIZE	FTP chunk size
FTPUPLOADK	FTP flag
FTPUSER	FTP user
FWB	Firewall bypass flag
GENCODE	Internal DarkComet code for recreating stub
INSTALL	Install stub
MELT	Delete stub after installation
MUTEX	Mutex used to prevent duplicate infection
NETDATA	List of controller domains and IP addresses
OFFLINEK	Offline keylogger flag
PERSIST	Install stub to persist through restarts
PWD	Password for stub/controller communication
SID	Campaign ID set by operator

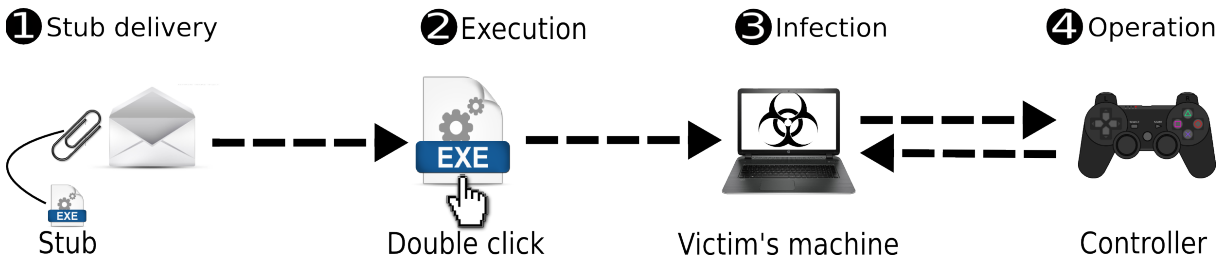


Figure 2.5. RAT infection process.

Table 2.1 shows the information embedded in the DarkComet stub. It mainly contains information pertaining to controller addressing, installation settings, and an optional FTP server for storing capture keystrokes remotely.

2.2.3 Infection

Figure 2.5 illustrates the RAT infection process. A RAT is made up of two pieces of software: a *stub* residing on the victim's machine, and a *controller* on the attacker's machine. A RAT infection process starts with the stub being delivered to the victim, for example, as an email attachment (❶). For the infection to be successful, the victim must execute the stub (❷). (The related work showed that stubs often masquerade as images or documents via the manipulation of their icons an/or extensions [78].) Every stub is configured with its controller address, which is either a hardcoded IP address or a domain name for resolution at the time of infection. Upon infection, the stub beacons to the controller on a preconfigured port until it establishes a connection with a controller (❸). Once connected, the stub executes commands sent to it by its controller, which serves as both the command and control server for the infected machine and the RAT operator's interactive interface to victims' machines (❹). A RAT operator, colloquially known as a *ratter*, interacts with the victim's machine via a GUI that allows even unsophisticated attackers to operate a RAT.

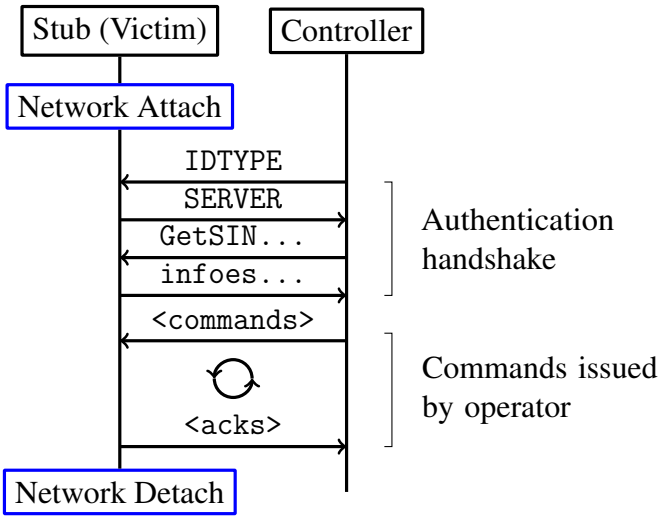


Figure 2.6. Network diagram of the DarkComet handshake. All data transmitted is first RC4-encrypted (with a static, pre-shared key) and base64-encoded. Detailed descriptions and examples of the packets in this diagram are available in prior work [28, 11]; however, Table 2.2 explains the fields in the `infoes` packet while Figure 2.7 provides an example.

```
00000000  69 6e 66 6f 65 73 43 72  61 63 6b 65 64 50 68 6f  |infoesCrackedPho| # <CAMPAIGN_ID>: CrackedPhotoshopSeeding
00000010  74 6f 73 68 6f 70 53 65  65 64 69 6e 67 7c 33 32  |toshopSeeding|32| # <WAN_IP>: 32.245.251.132
00000020  2e 32 34 35 2e 32 35 31  2e 31 33 32 20 2f 20 5b  |.245.251.132 / [| # <LAN_IP>: 192.168.53.71
00000030  31 39 32 2e 31 36 38 2e  35 33 2e 37 31 5d 20 3a  |192.168.53.71] :| # <PORT>: 1604
00000040  20 31 36 30 34 7c 41 43  43 4f 55 4e 54 49 4e 47  | 1604|ACCOUNTING| # <PC_NAME>: ACCOUNTING-ADMIN-PC
00000050  2d 41 44 4d 49 4e 2d 50  43 20 2f 20 41 64 6d 69  |-ADMIN-PC / Admi| # <USERNAME>: Administrator
00000060  6e 69 73 74 72 61 74 6f  72 7c 37 36 39 37 33 34  |nistrator|769734| # <NONCE>: 769734
00000070  7c 30 73 7c 57 69 6e 64  6f 77 73 20 58 50 20 53  ||0s|Windows XP S| # <PING>: 0s
00000080  65 72 76 69 63 65 20 50  61 63 6b 20 33 20 5b 32  |ervice Pack 3 [2| # <OS>: Windows XP Service Pack 3
00000090  36 30 30 5d 20 33 32 20  62 69 74 20 28 20 43 3a  |600] 32 bit ( C:| # <BUILD>: 2600
000000a0  5c 5c 20 29 7c 78 7c 7c  55 4b 7c 51 75 61 72 74  |\\ )|x||UK|Quart| # <BIT>: 32
000000b0  65 72 6c 79 20 46 69 6e  61 6e 63 69 61 6c 20 52  |erly Financial R| # <PATH>: C:\\
000000c0  65 70 6f 72 74 20 44 52  41 46 54 20 28 43 6f 6e  |eport DRAFT (Con| # <ADMIN_FLAG>: x
000000d0  66 69 64 65 6e 74 69 61  6c 29 20 2d 20 4d 69 63  |fidential) - Mic| # <WEBCAM_FLAG>:
000000e0  72 6f 73 6f 66 74 20 45  78 63 65 6c 7b 58 58 58  |rosoft Excel|{XX| # <COUNTRY>: UK
000000f0  58 58 58 58 58 58 2d 58  58 58 58 2d 58 58 58 58  |XXXXXX-XXXX-XXXX| # <ACTIVE_WINDOW>: Quarterly Financial Report
00000100  2d 58 58 58 58 2d 58 58  58 58 58 58 58 58 58 58  |-XXXX-XXXXXXXXXX| #      DRAFT (Confidential) - Microsoft Excel
00000110  58 58 7d 7c 38 33 25 7c  45 4e 47 4c 49 53 48 20  |XX}|83%|ENGLISH | # <HWID>: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
00000120  28 55 4e 49 54 45 44 20  4b 49 4e 47 44 4f 4d 29  |(UNITED KINGDOM)| # <RAM_USAGE>: 83%
00000130  20 47 42 20 2f 20 20 2d  2d 20 7c 30 36 2f 30 31  | GB /  -- |06/01| # <LANGUAGE>: ENGLISH (UNITED KINGDOM) GB
00000140  2f 32 30 31 38 20 41 54  20 30 37 3a 32 33 3a 33  |/2018 AT 07:23:3| # <INSTALL_DATE>: 06/01/2018 AT 07:23:34 PM
00000150  34 20 50 4d 7c 35 2e 33  2e 30                    |4 PM|5.3.0|       # <VERSION>: 5.3.0
```

Figure 2.7. An example DarkComet *Victim Info* packet with individual components extracted and labelled. Not from a real infection.

Table 2.2. Fields of information sent in DarkComet `infoes` handshake packet. *PII* indicates whether we consider the field to be potential PII of the victim.

Field	Description	PII
CAMPAIGN_ID	Stub's identity which operator defines	✓
WAN_IP	Public IP address of victim's machine	-
LAN_IP	Private IP address of victim's machine	-
PORT	Port used for stub/controller communication	-
PC_NAME	PC name of victim's machine	✓
USERNAME	Username of victim's machine	✓
NONCE	Nonce used to maintain handshake integrity	-
PING	Ping time between stub and controller	-
OS	Operating system name of victim's machine	-
BUILD	Operating system build number	-
BIT	Operating system architecture	-
PATH	Path to stub installation	-
ADMIN_FLAG	Denotes current user has admin privilege	-
WEBCAM_FLAG	Webcam capture is supported	-
COUNTRY	Geolocation of victim's machine	✓
ACTIVE_WINDOW	Header of currently open window	✓
HWID	Hardware identity of victim's machine	✓
RAM_USAGE	Current RAM usage of Victim's machine	-
LANGUAGE	Language setting of victim's machine	-
INSTALL_DATE	First day on which stub was executed	-
VERSION	RAT version	-

2.2.4 Victim-Controller Handshake

Once installed on the victim's machine, the DarkComet stub opens a TCP connection to the pre-configured address of its controller. After this connection is established, the stub and controller complete an authentication handshake in which the stub sends identifying information to the controller about itself and the infected host. Following this handshake, communication between the controller and stub consists of manual commands issued by the operator. All communication, including the handshake, is RC4-encrypted using a static, pre-shared key concatenated to a password configured by the operator. The key and password are embedded in the stub and can be recovered, allowing us to decrypt all DarkComet communications. Figure 2.6 depicts this handshake.

Note that DarkComet speaks a custom, application-layer protocol over TCP; many RATs eschew HTTP and other common protocols for custom command and control protocols. The stub establishes the TCP connection, as is customary with RAT infections, but then listens for the controller to identify itself - a behavior we classify as "passive." Many RATs, like njRAT, exhibit "active" protocols wherein the stub will both establish the connection *and* send the first identifying packet. The handshake itself is simple; the controller identifies itself, after which the stub does the same. The controller then asks for information, to which the stub replies with identifying information about itself (e.g. version number, campaign ID) and the victim machine (e.g. username, hostname). Table 2.2 explains the fields in this packet, while Figure 2.7 provides an example with each field extracted and labeled. All communication is RC4-encrypted with a pre-shared key. The stub will not respond to a DarkComet controller's first message if it is not encrypted with the correct key; thus, by choosing a unique password, operators are able to protect their stubs from being controlled by another operator or sinkhole, as well as prevent fake stubs from flooding their control panels with bogus greetings. Our analysis of the DarkComet protocol draws heavily on work by Denbow and Hertz [28].

2.2.5 Functionality

Kujawa [76] provides a comprehensive analysis of DarkComet's features and functionality, and in Table 4.2 I will enumerate each possible command. Here, however, I highlight the more interesting or dismaying functions DarkComet offers:

- **Computer power:** Remotely shutdown or restart the victim machine.
- **File manager:** Interactively browse and search the victim's filesystem.
- **Direct victim communication:** Open a chat window on the victim's desktop. Harass the victim with a number of functions entitled "fun functions," like hiding the file explorer. Figure 2.9 shows some of these functions.
- **Network exploration:** Access network shares. Port scan the local network. Launch denial-of-service attacks, as shown in Figure 2.8.
- **Password collection:** Automatically collect credentials and cookies from common applications or browsers.
- **Remote scripting:** Run scripts in various languages on the victim machine.
- **Spy functions:** Access the victim's webcam or microphone. Deploy a keylogger. Use remote desktop to control the victim machine.
- **System functions:** Use the process manager. (Un)install applications. Modify the registry. Access a remote shell.

2.2.6 Related Work

Its high profile usage has naturally made DarkComet the focus of analyses by industry and academia alike. Malware researchers have studied individual DarkComet campaigns in depth [47, 8, 76, 142, 12, 112], and have thoroughly analyzed its network protocol and behavior [28, 15, 16]. Denbow and Hertz [28] and Kevin Breen [12, 15, 16, 11] performed the seminal reverse engineering of DarkComet's network protocol handshake and executable configuration, respectively.

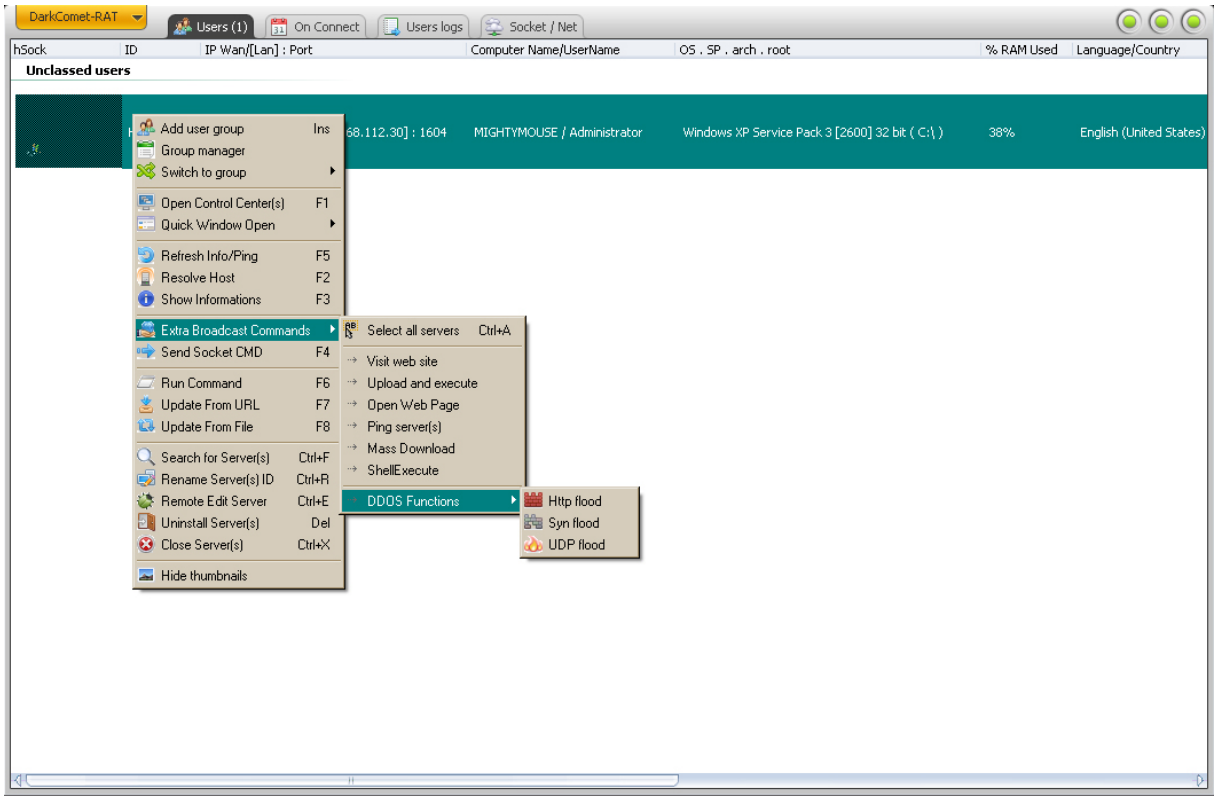


Figure 2.8. The denial-of-service options offered by DarkComet [76].

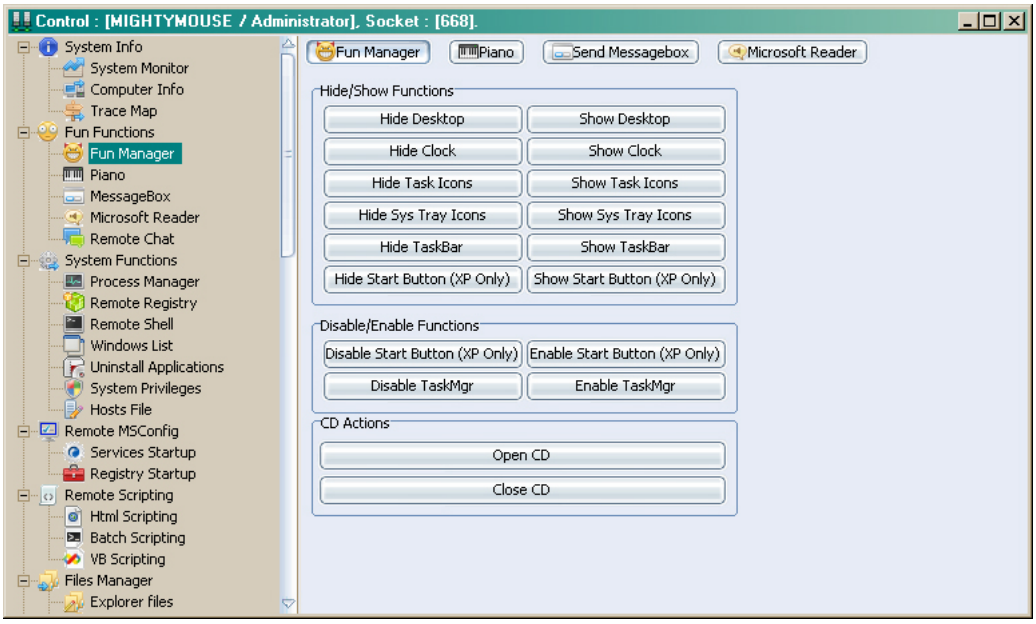


Figure 2.9. DarkComet's so-called "Fun Manager," which offers functions for harassing the victim [76].

Chapter 3

HAMELIN: A System for Tracking Remote Access Trojan Controllers

3.1 Introduction

HAMELIN is a modular system we designed and built to facilitate RAT active measurement experiments. It is comprised of two base components, RAT-HUNTER and RAT-SCAN. RAT-HUNTER is an extensible set of components that monitor malware repositories and indicator-of-compromise (IoC) feeds for new RAT binaries and configurations. RAT-SCAN is a targeted Internet scanner that continuously detects and monitors live RAT command-and-control servers (controllers), based on the configurations and binaries collected by RAT-HUNTER. RAT-SCAN is also extensible; although our experiments only require scanning for DarkComet and njRAT, RAT-SCAN supports and accepts modules for any RAT family protocol handshake. Together, RAT-HUNTER and RAT-SCAN continuously identify and track RAT controllers across the Internet.

Figure 3.1 provides a detailed timeline of the various components of HAMELIN and when they were active. RAT-HUNTER's binary and IoC collection is broken down by source; we describe it in Section 3.3 and Section 3.4. RAT-SCAN's scanning operations are broken into individual components as well; we describe these in Section 3.5.

Most importantly, the timeline in Figure 3.1 depicts the three active measurement experiments we performed: honeypotting, DDNS sinkholing, and DarkComet victim database

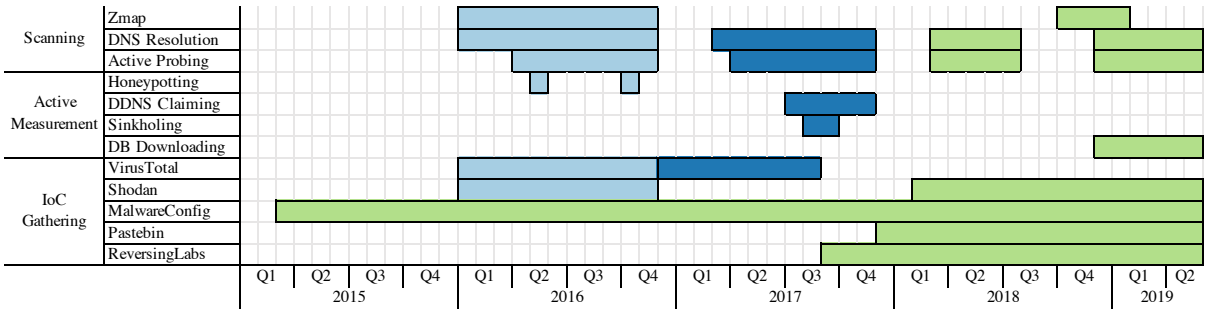


Figure 3.1. A timeline of the measurements performed by HAMELIN; particularly, RAT operator interaction honeypotting, RAT-associated DDNS domain sinkholing, and RAT victim database downloading.

downloading. The base scanning and threat hunting components of HAMELIN exist primarily to faciliate the running of these experiments; we will discuss our active measurement experiments and their findings in depth throughout the remaining chapters.

3.2 Background & Related Work

3.2.1 RAT Configuration Extraction

Kevin Breen pioneered much of the existing work regarding the extraction and decoding of configurations from common RAT families' binaries [14, 13]. RAT-HUNTER is reliant on his decoders. Breen also runs MalwareConfig, from which RAT-HUNTER sources configurations.

3.2.2 Internet-wide Scanning

Internet-wide scanning is a popular technique for measurement of Internet-exposed services, particularly in the threat detection community. For instance, it was used to measure the high-profile Mirai botnet outbreak in 2016 [7]. Many open-source tools exist to make rapid scanning of IPv4 space accessible to researchers, including ZMap [36], Masscan, and Unicornscan. Further, services like Shodan [88] and Censys [35] (a service based on the ZMap software suite) conduct continuous Internet-wide scanning and provide researchers access to the results. RAT-HUNTER follows all RAT controller detections reported by Shodan, while RAT-

SCAN uses ZMap to conduct untargeted Internet-wide scans for RAT controllers on common ports.

3.2.3 RAT Scanning

BladeRunner [37] was the first scanning-based system to actively discover RAT controllers by emulating RAT victims. Since then, Shodan partnered with Recorded Future [55] to add active probing and banner identification for numerous RAT families, including DarkComet, to the Shodan Malware Hunter project [87]. Recorded Future [62] recently presented on its use of Shodan Malware Hunter to identify active RAT controllers and corporate infections over four years of operation, including publishing the IP addresses of detected controllers [61]. To the best of our knowledge, Shodan Malware Hunter represents the state of the art in RAT controller detection in industry. Marczak *et al.* [85] created a scanner that was able to detect stealthy APT controllers by triggering error conditions.

3.3 RAT-HUNTER: Collection of RAT Binaries

We collected binaries from VirusTotal in two phases: first, we downloaded only DarkComet binaries to facilitate our DarkComet operator honeypotting experiment; second, we downloaded both DarkComet and njRAT binaries to facilitate our RAT dynamic DNS (DDNS) domain sinkholing experiment. We report on the results of each below.

3.3.1 Initial DarkComet Binary Collection

Between 2016-01-01 and 2016-12-01 we obtained samples of DarkComet by regularly querying VirusTotal Intelligence using a set of `YARA` rules [2], collecting fresh samples from the service's newest submissions [51]. We used an up-to-date, open-source set of DarkComet `YARA` rules [26]. On average we acquired 10 new, unique DarkComet samples per hour, resulting in 19,109 total unique DarkComet samples over the course of the study.

Malware is often packed to evade antivirus detection. DarkComet offers two runtime

Table 3.1. Languages used in DarkComet stub configurations for each unique and alphabetic domain, campaign name, and submitted filename (sanitized for more accurate detection). *Other* is any other spoken or written language.

	Domain Names		*Campaign IDs*		*Filenames*	
Source	*Cnt*	*Pct*	*Cnt*	*Pct*	*Cnt*	*Pct*
English	321	24%	236	48%	381	38%
Turkish	56	4%	32	6%	38	3%
German	49	3%	3	-	6	-
Spanish	22	1%	7	1%	9	-
Vietnamese	18	1%	5	1%	12	1%
Other	269	20%	93	19%	131	13%
Undetermined	555	43%	110	22%	401	41%
Total	1,290		486		978	

packing options, UPX [100] and MPress [95]. Of the 19,109 samples we collected, 18% were packed with one of these tools. 74% were not packed at all, and the remaining 8% were malformed. In total, we were able to automatically unpack 17,516 samples. For each unpacked sample, we extract its configuration information (recall Table 2.1) using an open-source RAT decoder [14]. In particular, we store the following:

- The **password** used to encrypt network communication to the controller.
- The **version** of DarkComet, also used in network communication encryption.
- The **campaign ID** assigned to the stub by the operator, used to manage multiple campaigns.
- A list of addresses of the stub's controller(s): **domain names** and/or **IP addresses**, plus **ports**.

Of the 17,516 samples from which we were able to extract configuration information, 13,339 were configured with valid addressing information - domain name(s) or IP address(es).

We perform automated language analysis of each sample's campaign ID, domain name(s), and submitted filename(s) using Google's language detection API [50]. Table 3.1 lists the results. English is the top language in all categories, while Turkish is the second most common, also

across all categories. Nothing prevents operators from using languages other than their native language; however, DarkComet appears to be quite popular in Turkey currently, as Section 7 will explore further.

Each extracted domain name is resolved hourly. This allows us to track controllers that use dynamic DNS services to frequently change IP addresses, a very common trend among RAT operators.

3.3.2 Secondary DarkComet & njRAT Binary Collection

Table 3.2. Counts of RAT samples downloaded, both total and unique, by family. *Other* are RAT samples that matched our YARA signatures incorrectly. *Failed Decoding* are samples from which configurations could not be extracted.

Family	# Sample	% Sample	# Unique
DarkComet	22,362	66.6	22,124
njRAT	5,049	15.0	4,535
Other	5	<0.1	-
Failed Decoding	6,144	18.3	-
Total	33,560	100.0	26,659

Between 2016-12-01 and 2017-08-17 we monitored VirusTotal for all known versions of DarkComet and njRAT using `YARA` rules. Over those 9 months we collected 33,560 samples, from which attempted to extract configurations [13]. Per Table 3.2, we obtained configurations from 22,124 *unique* samples of DarkComet and 4,535 *unique* samples of njRAT. Our `YARA` rules cover more subfamilies than we can decode, hence our failure to decode all samples.

Many of these RAT configurations contain domain names rather than hard-coded IP addresses. Table 3.3 shows that the majority of these domains are associated with free Dynamic DNS (DDNS) providers, with No-IP encompassing 60% of all discovered domains and 77% of known DDNS domains.

In order to maintain an updated list of potential command-and-control addresses, we resolved each of the 14,273 domains we extracted from our malware samples hourly, beginning

Table 3.3. Breakdown of C&C domains in our RAT sample population by Dynamic DNS provider. *Unknown* encompasses all domains unrelated to a known DDNS provider.

Controller Type	# Domain	% Domain
No-IP	8,564	60.0
DuckDNS	2,459	17.2
FreeDNS	92	<0.1
DynDNS	38	<0.1
Total Dynamic DNS	11,153	78.1
Unknown	3,120	21.9
Total	14,273	100.0

on 2017-04-21 and ending on 2017-11-26. Over this period, we recorded 67,023 resolutions to unique IP addresses. We augmented these with passive DNS records dating back to 2010 from Farsight [44], VirusTotal, and PassiveTotal [123].

3.4 RAT-HUNTER: Collection of RAT Configurations

We supplement our collection of RAT binaries from VirusTotal by directly downloading extracted DarkComet configurations and controller indicators-of-compromise from the following sources:

MalwareConfig

MalwareConfig is a malware repository that specializes in extracting configurations from RATs. It makes public all samples that have been submitted to it.

Shodan

Shodan is search engine for Internet-connected devices that continuously scans the Internet for various services, including RATs like DarkComet [88], and publishes their findings.

Pastebin

RAT binaries are frequently stored on Pastebin for download as a second-stage payload. A tool called PasteHunter [105] allows for the scanning of Pastebin with various `YARA` rules; the Twitter account @ScumBots runs this tool with common RAT `YARA` signatures and publishes discovered RAT configurations.

ReversingLabs

ReversingLabs provides a feed of the configurations of RAT samples its customers encounter.

Table 3.4. Sources of DarkComet sample configurations, including resultant discovered DarkComet hosts and downloaded databases. *After resolution of domains.

Source	Sample IPs	Sample Domains	Total IPs*	DarkComet Hosts	DBs
MalwareConfig	808	4,908	6,843	104	49
ReversingLabs	10,842	47,485	77,528	1,894	1,026
@ScumBots	436	2,638	70,546	282	116
Shodan	5,371	0	5,371	593	340
VirusTotal	58,864	14,169	99,835	311	151
Total	74,969	64,343	220,918	3,345	1,509

From December 1, 2016 to April 4, 2019, we collected 139,312 unique DarkComet controller configurations from the described malware feeds, the details of which are shown in Table 3.4. Many configurations use domain names to address their DarkComet controllers, so we resolve each suspected DarkComet domain name continuously to augment our list of suspected DarkComet host IP addresses. Of the sources in Table 3.4, only Shodan does *not* provide domain names; however, all IP addresses provided by Shodan's feed were also either present in configurations from the other sources or discovered during domain resolution.

3.5 RAT-SCAN: Active Detection & Monitoring of RAT Controllers

Upon collecting RAT configurations — namely, RAT controller addresses — from RAT binaries and other IoC feeds, we use an asynchronous, targeted scanner of our own design to track and monitor those controllers that are active. Below, we describe the design of this scanner.

3.5.1 RAT-SCAN Design

RAT-SCAN is a fast, asynchronous, active network scanner that tests for the presence of various RAT controllers running on target IP addresses and domain names. It takes as input a list of target IP addresses and domain names, as well as the ports on which they may be running RAT controller software; these targets are provided by RAT-HUNTER. It is modular and extensible, in that RAT-SCAN can probe a given target for any number of RAT controller families (provided the module is enabled); we primarily use it to scan for DarkComet and njRAT. Upon connection to a target, for each of the RAT families enabled, RAT-SCAN emulates a RAT stub on a victim machine and attempts to shake hands with the target service. (An example handshake can be found in Figure 2.6, wherein RAT-SCAN would take the place of the stub.) If the target service can complete the *entire* handshake, it is considered an active controller of the given RAT family.

If the controller can only complete *part* of a given RAT handshake, RAT-SCAN considers the entity to be a sinkhole instead of a legitimate controller. Security vendors and academics will sometimes run sinkhole scripts emulating RAT controllers on known bad domains, but fail to implement the entire handshake; we detect this. Further, RAT-SCAN will test those services considered controllers for improper responses (e.g. with a different password than was used to first detect it); sinkholes will often fall for this as well.

ZMap

ZMap is an open-source tool that conducts rapid, Internet-wide network scans [36]. RAT-SCAN has the ability to launch indiscriminate, Internet-wide scans on individual ports (e.g.

Table 3.5. Percentages are reported as percentage of the total number of controllers monitored (*n*=9,877). The sum of the controllers in *Total* exceeds *n*, as some controllers were discovered by multiple sources. Likewise, the sum of the controllers in *Unique* falls short of *n*, for the same reason.

	Total		*Unique*	
Source	*Cnt*	*Pct*	*Cnt*	*Pct*
ZMap	5,451	56%	4,417	45%
Domain Resolution	4,385	45%	3,604	37%
Shodan	810	8%	456	5%
Hard-coded IP Address	226	2%	125	1%

DarkComet's default port, 1604) to discover controllers for which we do not have configurations. We had considerable success using ZMap, as evinced by Table 3.8; however, the usefulness of these discovered controllers in active measurement experiments is sometimes limited due to our missing their controllers' stub configurations.

3.5.2 Scanning Windows: Overview

As shown in the timeline in Figure 3.1, we conducted three periods of active scanning to facilitate our three experiments (honeypotting, sinkholing, and database downloading). We will now describe each scanning window individually.

3.5.3 Scanning Window: Honeypot Deployment

Chronologically, our first active measurement experiment involved the deployment of live DarkComet samples in high-fidelity honeypots to observe operator activity. As we were resource-constrained by the number of honeypots we could run simultaneously, we decided to only run samples when their corresponding controller was definitively online; hence, we needed to continuously monitor all DarkComet controllers for which we had samples.

Between 2016-04-01 and 2016-12-01, we discovered and monitored 9,877 unique DarkComet controllers across the Internet. Table 3.5 breaks down the sources of our controllers. The high numbers of unique controllers indicate that each source largely contributes its own

Table 3.6. Countries of the IP addresses of a) the global population of scanned DarkComet controllers, and b) the controllers to which our live trials connected, as resolved by MaxMind's GeoLiteCity database [90]. Addresses without resolution are omitted.

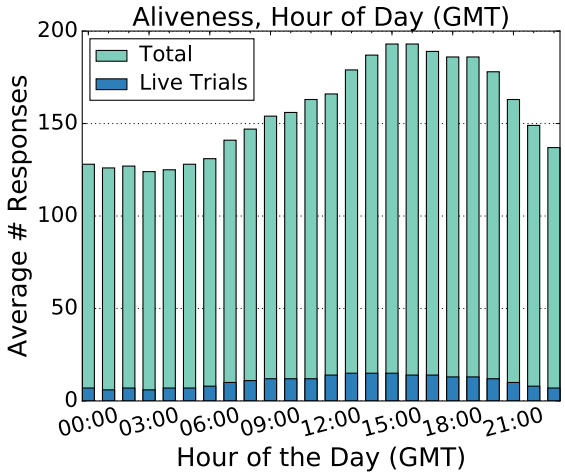
	Global Scanning		*Live Trials*	
Country	*Cnt*	*Pct*	*Cnt*	*Pct*
Turkey	3,680	37%	222	25%
Russian Federation	1,495	15%	188	21%
United States	319	3%	36	4%
Brazil	306	3%	40	4%
France	283	2%	22	2%
Ukraine	282	2%	52	5%
Other	3,512	36%	307	35%
Total	9,877		867	

population to the scanner; for instance, those controllers found by ZMap did not tend to overlap with the controllers configured in our samples.

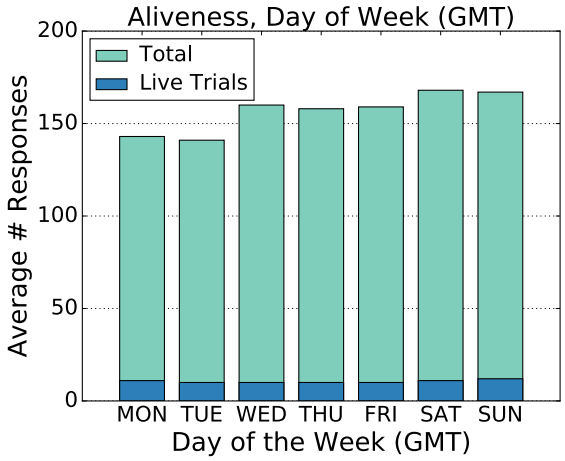
As noted in our methodology, we developed a method for scanning the Internet for DarkComet controllers. This allows us to poll all discovered controllers on the Internet at any given time. Note that this includes controllers for which we do not have a sample. Figure 3.3 shows the results of the continuous scan, with a series for all monitored controllers as well as a series for just those controllers that connected to our honeypots during live trials. At any given time, we are monitoring about 175 online DarkComet controllers. As our scanning is not comprehensive, these numbers only provide a baseline for the actual number of DarkComet controllers on the Internet.

The cumulative number of unique controllers discovered during scanning increased essentially linearly over the course of the measurement. Many controller domain names from our samples used a dynamic DNS service. We suspect that this steady growth is at least partly due to IP address churn. Indeed, 45% of monitored controllers were discovered by continuous resolution of DarkComet-associated domain names.

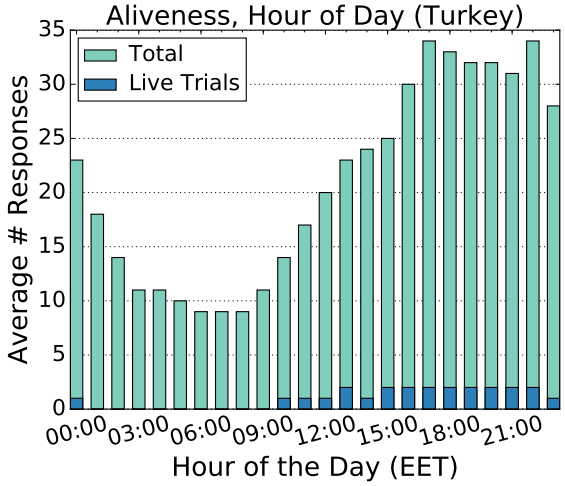
Table 3.6 shows the geographic distribution of controller IP addresses that we monitored



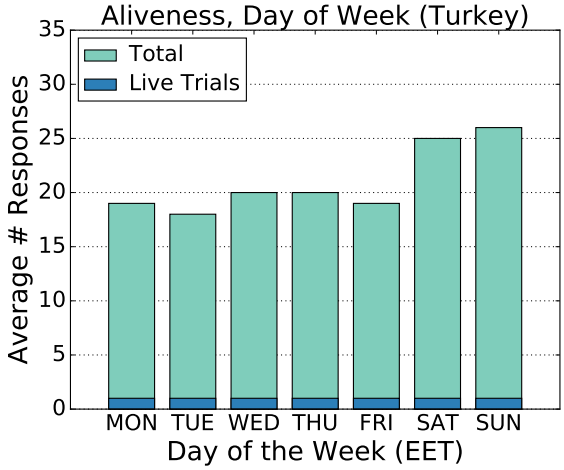
(a) Average number of DarkComet controller nodes online per hour of the day.



(b) Average number of DarkComet controller nodes online per day of the week.



(c) Average number of DarkComet controller nodes online per hour of the day in Turkey.



(d) Average number of DarkComet controller nodes online per day of the week in Turkey.

Figure 3.2. Scanning breakdowns by day and hour.

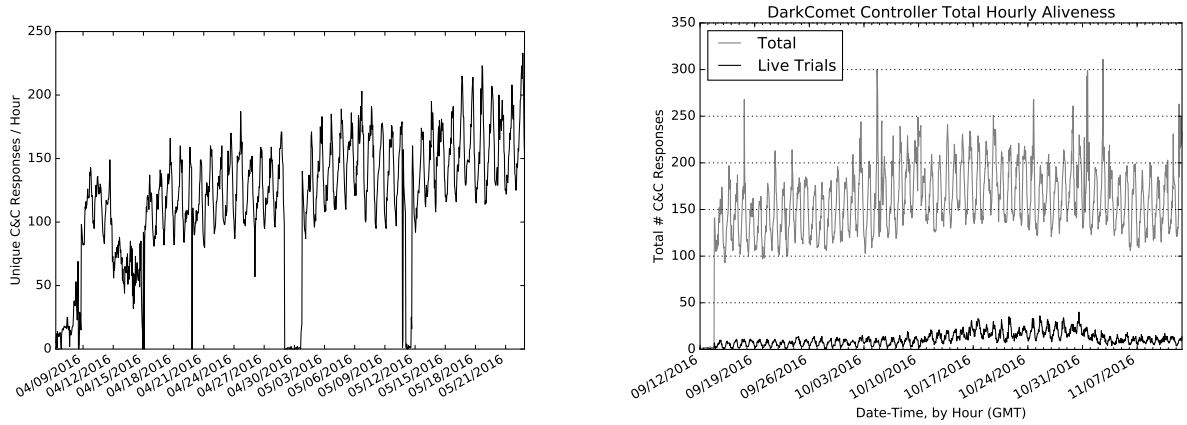


Figure 3.3. Number of DarkComet controller nodes online each hour, binned by hour. We detected 9,877 unique controllers in total. Counts are from an Internet-wide scan for active DarkComet controllers, and include controllers for which we do not have a RAT sample.

globally and that connected to our honeypots. Russia and Turkey are the most prevalent, but operators may be using a VPN service. If this is the case, then the IP location will indicate the location of the VPN rather than the actual operator location. Therefore, Table 3.6 should be interpreted with caution.

Figure 3.2b shows the average number of controllers online, binned by day of the week. Note that controller aliveness trends upwards towards the weekend, with about 20% more controllers online on Sunday than Monday. We suspect that this is due to our focus on "casual" RAT operators who may be online only during weekends. Figure 3.2a shows the same data by hour of the day. If we assume that the geolocation data in Table 3.6 represents true operator location, then the peak between 16:00 and 17:00 UTC falls in the early evening in Eastern Europe, while the trough at 2:00 UTC falls in the very early morning in Eastern Europe. This again suggests that at least some of the monitored RAT operators are "casual" ratters. Nevertheless, even at the lowest point, there are over 100 controllers online.

Figure 7.1 gives us reason to suspect that the data in Table 3.6 is indicative of the actual operator geolocation, so we produce the previous graphs for just the controllers that geolocate to Turkey, adjusted to EET (Turkey's timezone). The results, illustrated in Figures 3.2c and 3.2d,

Table 3.7. User-types of the IP addresses of a) the global population of scanned DarkComet controllers, and b) the controllers to which our live trials connected, as resolved by MaxMind's GeoIP2 Insight service [89]. Addresses without resolution are omitted.

	Global Scanning		*Live Trials*	
Country	*Cnt*	*Pct*	*Cnt*	*Pct*
Residential	8,830	89%	779	90%
Hosting	704	7%	54	6%
Cellular	288	3%	24	3%
Other	47	-	6	-
Undetermined	8	-	4	-
Total	9,877		867	

Table 3.8. Monitored controllers and collected samples that operate on the standard DarkComet port (1604) vs. non-standard ports. Samples are sometimes configured with multiple controller addresses and ports, explaining why the sum of samples in the *Cnt* column exceeds the total. †: Comprised of 246 *unique* ports. ‡: Comprised of 1,209 *unique* ports.

	Global Scanning		*Sample Configs*	
Port	*Cnt*	*Pct*	*Cnt*	*Pct*
Standard Port	7,928	80%	8,971	67%
Non-standard Ports	1,949†	20%	6,016‡	45%
Total	9,877		13,339	

clearly support the expected "casual operator" trends: weekend activity is significantly higher than weekday activity, and early evening activity dwarfs early morning activity.

The user-type distribution reported in Table 3.7 further supports this trend. Almost 90% of the discovered controllers operate behind IP addresses with residential user-types. This suggests that the majority of controllers are run on residential networks, likely with little of the operational security often seen in botnet proxies.

3.5.4 Scanning Window: Sinkholing

Our second active measurement experiment involved the sinkholing of dynamic DNS domains associated with njRAT and DarkComet activity (e.g., included in RAT sample configurations) in order to measure the residual victim populations of RAT campaigns. The scanning

operation we ran simultaneously served two purposes: first, it allowed us to complement our victim measurements, and second, it informed our attempted poaching of recently abandoned domains.

Table 3.9. Breakdown of RAT controllers detected on IP addresses responsive to RAT-SCAN. Some IP addresses hosted multiple types of RAT controller.

Controller Type	# IP	% IP
njRAT	4,584	71.6
DarkComet	2,032	31.7
DarkComet (Unknown Password)	11	0.2
Total	6,401	100.0

Our scanning operation began on 2017-05-11 and ended on 2017-11-25, for a total of 198 days. We continuously probed each of these 67,023 IP addresses hourly for evidence of RAT controller software. During this period, we established 86,694 connections to 6,401 IP addresses exhibiting behavior indicative of RAT controller software; 2,032 DarkComet controllers and 4,584 njRAT controllers, with some IPs hosting both. Table 3.9 provides a summary of our scanning operation.

Other than on our sinkhole itself (RAT-HOLE), our sinkhole detection logic did not trigger during this study. We are led to believe that all controllers reported here are either legitimate instances of the controller software, or services that have implemented the handshake properly and maintain a single configuration. We suspect that such services exist; however, we currently have no way of distinguishing them from legitimate controllers. Further, we have no reason to believe that we encountered any high-fidelity sinkholes similar to RAT-HOLE.

IPjetable VPN: 141.255.144/20

Of the 6,401 IP addresses RAT-SCAN successfully probed, a full 2,635 (or 40.2%) came from this address space. Further, these IP addresses accounted for over 40% of *all* connections made during the six months of active scanner operation, exhibiting abnormal longevity compared to other controllers. This space is owned by IPjetable [63], a French company that provides

free VPN services and that is recommended by hundreds of RAT instruction videos available online [3]. IP addresses belonging to IPjetable are even present in the Recorded Future IoC dataset from 2015.

Relakks VPN: 93.182.168/21

Though not nearly as large as the IPJetable address space, this space contained 167 IP addresses probed by RAT-SCAN (2.6% of all IP addresses), accounting for nearly 2% of all RAT-SCAN connections. This address space belongs to Relakks VPN [117], a Swedish company that provides free VPN services and is likewise recommended by RAT instruction videos [132] and HackForums members.

VPS providers

In addition to using VPN's, we found the use of VPS instances from prominent services like Amazon AWS, Microsoft Azure, and Digital Ocean, as well as less reputable providers like OVH.

3.5.5 Scanning Window: Database Downloading

Our final active measurement experiment involved the downloading of DarkComet victim databases from DarkComet controllers using a blind file retrieval vulnerability inherent to DarkComet controller software. As such, RAT-SCAN was the core component of the study, both discovering DarkComet controllers and downloading their victim databases (using a modified DarkComet scan module).

From December 5, 2018 to April 4, 2019, continuously probed a final list of 220,918 target hosts for DarkComet network signatures and made contact with 3,345 live DarkComet hosts. From 1,509 of these hosts, we successfully downloaded databases, recording both their hostnames and resolved IP addresses.

Figure 3.4 shows the total number of unique DarkComet controllers online per week during the study's 120-day observational period. It also shows that fraction of said controllers

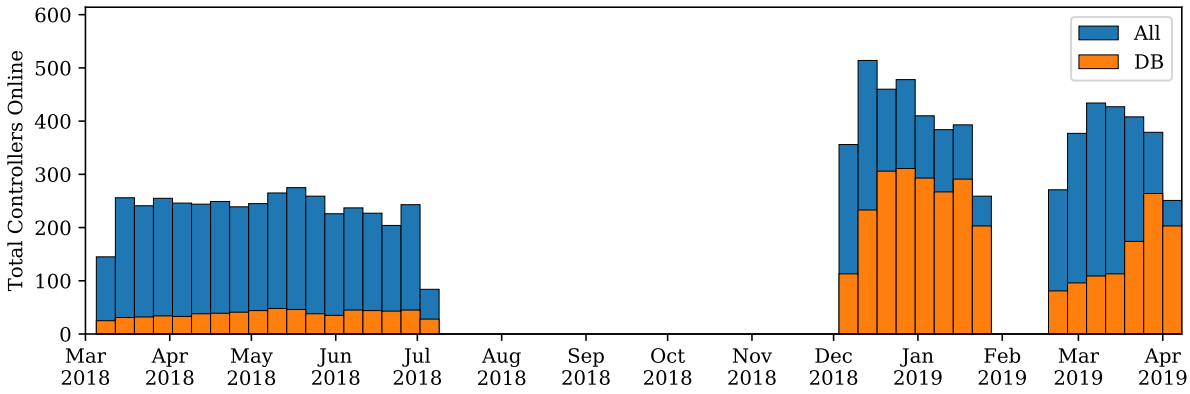


Figure 3.4. Total DarkComet controllers online per week.

from which our downloader acquired a database. Figure 3.5 provides the same data broken down by day rather than week, while Figure 3.6 shows the number of *new* controllers seen each day never before observed by our scanner. All data is timezone-adjusted per each controller's IP address location. Controller activity varies dramatically throughout the day, trending upwards between 6:00 AM and 8:00 PM, with about 50% more controllers online at 8:00 PM than 6:00 AMḞigure 3.7 showcases this trend. Though expected, we see no significant change in controller activity on weekends.

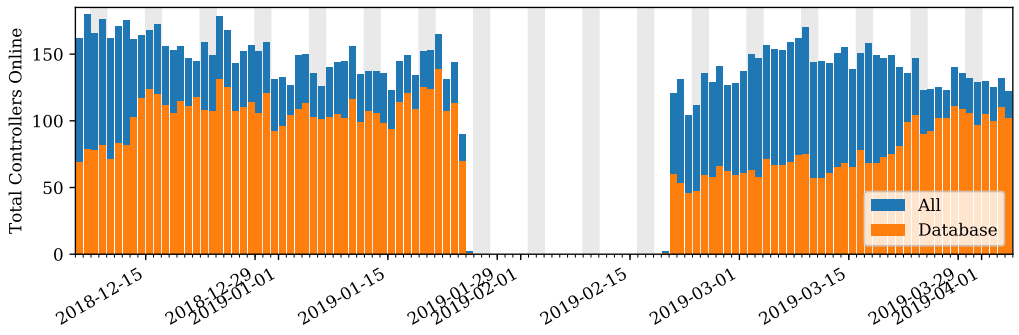


Figure 3.5. Total number of DarkComet controllers online per day.

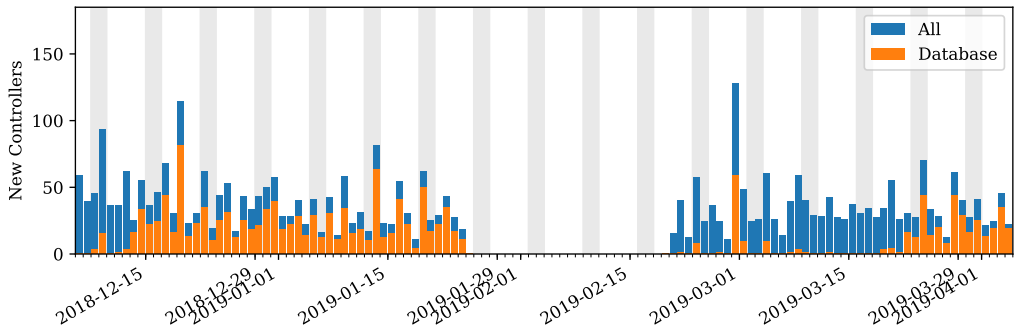


Figure 3.6. Total number of *new* DarkComet controllers per day.

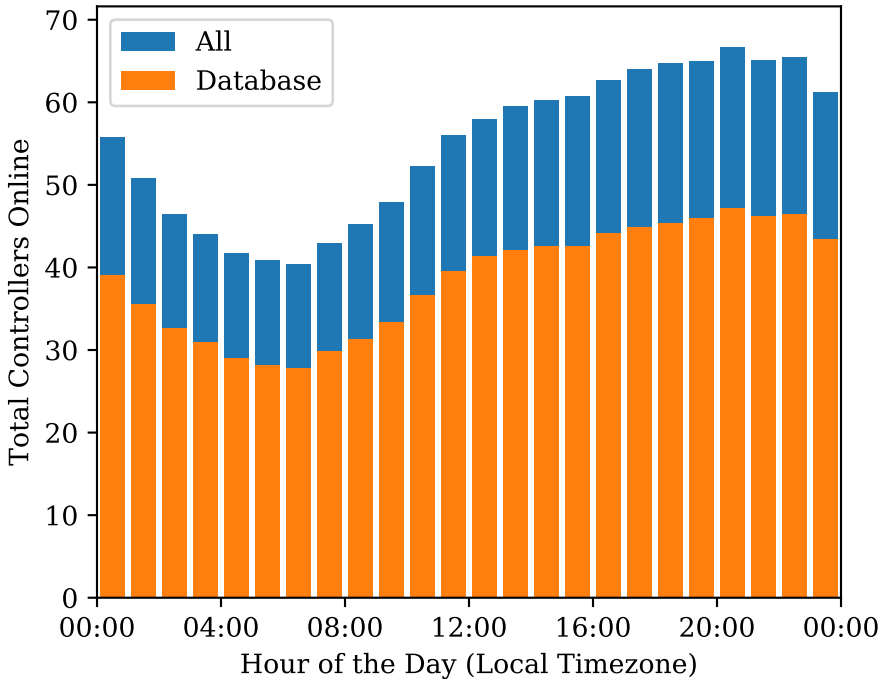


Figure 3.7. Average number of DarkComet controllers online per hour of the day.

Table 3.10. VPN providers for all hosts running DarkComet controller software, and those from which we downloaded the victim database.

VPN/VPS Provider	Total Hosts		Database Hosts	
IPjetable	177	5.3%	67	4.4%
Relakks VPN	46	1.4%	19	1.3%
anmaxx	16	0.5%	1	0.1%
gchao.com	12	0.4%	-	-
Amazon AWS	10	0.3%	7	0.5%
Infium, UAB	10	0.3%	5	0.3%
ngrok.io	6	0.2%	-	-
live-servers.net	6	0.2%	6	0.4%
M247	5	0.1%	1	0.1%
H88	5	0.1%	2	0.1%
Digital Ocean	4	0.1%	1	0.1%
Other (99 services)	150	4.5%	61	-
Unknown	2898	86.6%	1339	88.7%
Total	3345	100.0%	1509	100.0%

Anonymizing Infrastructure (VPN/VPS) Usage

Shown in Table 3.10, 13.4% of scanned DarkComet hosts use known VPN or VPS services, mainly IPjetable and Relakks VPN.[1] Such a relatively small population of hosts using anonymizing infrastructure suggests that the DarkComet operators in our data set may lack even basic operational security measures. Therefore, Internet-wide scanners like ZMap could potentially locate most of these operators' actual gateways.

Dynamic DNS Usage

As Dynamic DNS (DDNS) is a popular tool among DarkComet operators, we compare the domain names found in the RAT configurations in our data set against a list of 1,193 domains belonging to 121 prominent DDNS providers.[2] Table 3.11 describes the results of this comparison. We find that most of the domains used by DarkComet operators belong to one of two free DDNS providers, No-IP and DuckDNS. DarkComet's controller software explicitly

[1] We use MaxMind and Recorded Future to compile a list of anonymized IP ranges.

[2] We will make this list of DDNS providers and their domains publicly avaiable.

Table 3.11. Dynamic DNS providers for all hosts running DarkComet controller software, and those from which we downloaded the victim database. *Some hosts use more than one provider.*

DDNS Provider	Total Hosts		Database Hosts	
No-IP	1344	40.2%	689	45.7%
duckdns.org	572	17.1%	333	22.1%
DNS Exit	80	2.4%	36	2.4%
Q-See	79	2.4%	33	2.2%
Dynu Systems	34	1.0%	18	1.2%
AirVPN DDNS	14	0.4%	7	0.5%
ChangeIP	13	0.4%	7	0.5%
freedns.afraid.org	12	0.4%	5	0.3%
dyn.com	2	0.1%	-	-
PubYun	1	¡0.1%	-	-
THDDNS	1	¡0.1%	1	0.1%
DNSdynamic	1	¡0.1%	-	-
Unknown	1561	46.7%	543	36.0%
Total	3345	100.0%	1509	100.0%

interfaces with No-IP's update client, a likely source of its popularity in particular.

3.6 Limitations

3.6.1 ZMap Coverage

During our first experiment, our ZMap scans only ran on the default DarkComet port, 1604. Table 3.8 demonstrates the limitations of this strategy; while 80% of our discovered DarkComet controllers run on the default port, a full 45% of collected samples are configured with at least one controller using a non-default port. However, these samples were configured with 1,209 unique non-default ports; scanning globally on that many ports was simply beyond our capacity.

3.6.2 Sample Unpacking & Configuration Extraction

Sample unpacking and decoding is the most thorough method for obtaining RAT configuration information from our samples; however, its applicability is limited. We encountered little

packer diversity in our sample set, with unpackers being freely available for the two types we did mainly encounter. However, we were still unable to unpack and decode 8% of obtained samples. Likewise, the RAT configuration decoders we used have finite coverage; were we to expand our study to include multiple RAT families (of which there are hundreds), the effort required to produce a general decoder would be immense. Developing each individual RAT decoder is a non-trivial reverse engineering endeavour.

3.6.3 Validating Scanning Results

We have little ground truth to evaluate methods for distinguishing between legitimate RAT controllers and sinkhole operations, other than our own sinkhole. As future work we will explore additional methods of ethically probing controllers, such as calling rarely used API functions that are unlikely to be implemented by sinkholes.

3.6.4 Scanning Result Confidence

RAT controllers are taciturn by design, revealing practically nothing to RAT-SCAN during the handshake. DarkComet controllers acknowledge a victims' correct password; njRAT controllers do not even do this. Therefore, when we classify a host as a DarkComet sinkhole we are fairly confident, but when we label a host a controller it is possible that it is a high-fidelity sinkhole or sandboxed controller.

3.7 Discussion

3.7.1 Comparing RAT-SCAN to Shodan & Insikt Group (Recorded Future)

Between 2018-12-01 and 2019-01-09, Insikt Group at Recorded Future conducted a global RAT controller detection effort and released their findings. As RAT-SCAN was operational during this period and RAT-HUNTER was monitoring Shodan, we are able to compare our performance detecting DarkComet controllers with our closest competitors.

During this period, RAT-SCAN actively scanned and detected 1,548 DarkComet controllers and 59 suspected sinkholes. In the same period, Shodan published 262 DarkComet detections and Insikt Group published 247. Neither Shodan nor Insikt Group report on illegitimate controllers like sinkholes, so we assume they consider all findings legitimate. Insikt Group found 68 controllers which we did not detect actively; however, they missed 1,370 of our detections and included 11 sinkholes in their results. Likewise, Shodan found 52 controllers we did not, but missed 1,339 controllers and reported on 22 sinkholes.

In short, Shodan only detected 14% of the controllers we did, Insikt only 11%. Both detect a small number of controllers we missed, but both also include known DarkComet sinkholes in their detection results. These results are expected; RAT-SCAN targets RAT controllers based on the wide array of sources RAT-HUNTER follows, including Shodan itself. Further, RAT-SCAN implements logic to detect DarkComet sinkholes (e.g., attempting to download the victim database) beyond the capabilities of a typical banner grabber.

Shodan and Recorded Future jointly operate Shodan Malware Hunter, so we would expect them to have the same findings over this period; however, they differ slightly — Shodan and Insikt only share 226 detections.

3.8 Acknowledgements

Chapter 3, in part, is a reprint of the material as it appears in Proceedings of the 38th IEEE Symposium on Security and Privacy 2017. Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blond, Damon McCoy, Kirill Levchenko, 2017. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material as it appears in Proceedings of the 27th USENIX Security Symposium 2018. Mohammad Rezaeirad, Brown Farinholt, Hitesh Dharmdasani, Paul Pearce, Kirill Levchenko, Damon McCoy, 2018. The dissertation author was the primary investigator and author of the components of the paper presented in this dissertation.

Chapter 3, in part, has been submitted for publication of the material as it may appear in Proceedings of the 26th ACM Conference on Computer and Communications Security, 2019. Brown Farinholt, Mohammad Rezaeirad, Damon McCoy, Kirill Levchenko, 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Understanding Attacker Motivations

The first active measurement experiment we launched using HAMELIN was intended to glean the motivations of DarkComet operators at scale. Compared to large-scale malware like denial-of-service botnets, determining the motivations of the users of manual malware (e.g., RATs) is challenging due to its versatility; RATs have been used by actors of all levels of sophistication, for espionage, information theft, voyeurism and extortion. To address this challenge, we modified HAMELIN to submit live DarkComet samples to a cluster of high-fidelity honeypots (a *honeyfarm*), monitoring their operators as they engage with supposed new victims. To our knowledge, this experiment was the first large-scale systematic study of RAT use.

4.1 Introduction

Recent events indicate that malware usage has started to shift from large-scale threats like botnets to lower-volume threats designed to spy on specific users or systems (e.g., [21, 43, 86, 140]). In a botnet, each machine is an indistinguishable bundle of resources. Though imposing in size, the use of botnets was banal—spamming, click fraud, and the like. Low-volume threats, on the other hand, aim to extract something of greater value from each infection. In this regime, the preferred tool for exploiting individual infections is the RAT, as it gives a *human user* interactive remote access to an infected machine.

Because of their great flexibility, RATs have been used by a broad range of actors. For

example, intelligence agencies and governments use RATs to spy on dissidents, journalists, and other governments [21, 86, 140]; while voyeurs use these tools to spy on victims, collecting pictures stored on the computer, capturing live webcam images, and recording audio [31, 43]. The latter's intentions range from pure voyeurism to extortion and blackmail, with victims from celebrities like Miss Teen USA [43] to countless unnamed users worldwide.

The subject of this chapter is the behavior of RAT operators in newly infected machines. Though use of RATs is well documented, knowledge of *how* they are used by these actors is limited. Indeed, studying RATs presents its unique challenges. The first of these is procuring fresh malware samples that are still in operation. In practice, low-volume malware typically does not have the same broad distribution as, say, a botnet executable. An attacker often infects the intended victims by sending them an email message with a malicious attachment or luring them to a Web page that exploits a browser vulnerability. To obtain such samples, researchers must obtain them from victims or a vantage point between attackers and victims, and before they cease operating.

We obtain our samples from VirusTotal [51], an online virus scanner, using RATHUNTER. VirusTotal is often used to check a suspicious file or URL received via email, social media, etc., and thus provides a unique vantage point for studying low-volume malware. For example, recent related work leveraged VirusTotal to measure and analyze malicious documents employed in targeted attacks against two ethnic groups and 12 countries spanning three continents [77]. In this chapter, we show that we can rely on VirusTotal to collect fresh RAT samples and leverage controllers' aliveness to monitor them while they are in operation.

The second challenge in studying RATs is monitoring what the attacker does when connected. Attackers expect a successful infection to give them access to a victim's computer. Preliminary experiments showed that executing a RAT in a typical VM used to study malware may lure the attacker to connect, but will quickly give away the setup when examined more closely. To elicit natural behavior, we disguised our machines as real users' PCs, suitably personalized, though not linked to a real user. Finally, we need to capture the activity of the RAT

operators to reliably reconstruct their behaviors.

During this experiment, we obtained 19,109 samples of DarkComet, a popular RAT used by threat actors of all levels of sophistication. In most cases, our sample was the result of running a dropper executable that may have been dropped itself. Based on the primary infection vectors, at least some of the instances appear to be genuine attempts to infect a victim. We took each sample we obtained, ran it in a Cuckoo sandbox [54], and recorded all commands issued to the RAT by the controller. We then used the data collected to reconstruct the behavior of the operator in our system and carry out our analysis.

In particular, we analyzed operator behavior in order to infer the operator's purpose for infecting the machine. Though some actions like searching for password files were common to most sessions, others gave us insight into operator goals. In 61% and 26% of the cases, respectively, operators attempted to monitor the user through the webcam and microphone. More niche groups of operators stole credentials and bitcoin wallets, or dropped malware and hacking tools to use the host as a staging point for further infection.

The contributions described in this chapter are as follows:

- We describe a system (a modification to HAMELIN, as described in Chapter 3) for automatically executing RAT samples in high-interaction honeypots intended to faithfully resemble real users, and thus elicit genuine operator behavior.
- We describe the results of a measurement study of DarkComet operator behavior. We executed 1,165 unique samples of DarkComet over two separate two-week periods, resulting in 785 interactive sessions with live operators, totaling 52.9 hours of engaged operator interaction with our honeypots.
- We describe the use of RAT honeypots as a defensive measure, both as a tarpit defense, drawing attacker attention and resources from legitimate targets, and as a threat intelligence sensor. We use our experiments to assess the viability of each.

The rest of the chapter is organized as follows. Section 4.2 provides necessary back-

ground information. Section 4.3 describes our honeypot system and measurement methodology. Section 4.4 presents our results. Section 4.5 discusses our results and examines possible applications of honeypots. Section 4.6 discusses the ethics of our methodology. Section 4.7 presents the work's limitations. Section 4.8 concludes the chapter.

4.2 Background & Related Work

To our knowledge, however, we are the first to systematically study *RAT operator behavior*. We do this by executing DarkComet samples in a sandboxed malware analysis environment.

4.2.1 Sandboxed Malware Execution

Executing malware in a sandboxed environment is a common form of *dynamic malware analysis*, and an established malware analysis technique [9, 64, 75, 110, 136, 141, 141, 110]. The use of realistic honeypots to ensnare *manual* malware operators was conceived by Stoll [129]. Conversely, techniques for countering dynamic analysis by detecting failures in honeypot realism have also been developed [124].

4.2.2 Low-Volume Malware Attacks

The closest academic works on low-volume attacks have primarily focused on the reconnaissance phases occurring *before* infection [57, 77, 78, 85]. Although Marczak *et al.* [85] looked at the possible real-world consequences of these attacks, their conclusions were based on conjectures instead of live compromise monitoring.

4.3 Methodology

Recall that the goal of this work is to understand how one particular RAT, DarkComet, is used in the wild. Our study has two parts. The first part is concerned with the global population of DarkComet RAT operators. We collected data for this part of the study by scanning the

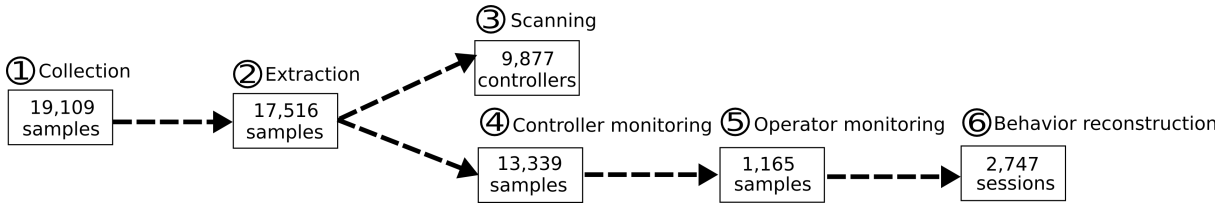


Figure 4.1. Our data collection and processing workflow. Each box displays the number of corresponding entries after each step.

IPv4 address space for DarkComet controllers. We previously described our global scanning methodology and dataset in Section 3.5.3. The second part of our study is concerned with operator behavior. We collected data for this experiment by executing samples of DarkComet in a contained environment and monitoring operator behavior. We carried out two rounds of experiments, executing 1,165 samples over 31 days. Sections 3.3.1 through 4.3.2 describe these experiments, which also rely on the scanning data collected in Section 3.5.3. We conclude the section with a discussion of limitations of this work (Section 4.7) and ethical considerations (Section 4.6).

4.3.1 Live Operator Monitoring

We conducted two separate experiments, each lasting two weeks. The first began on 2016-05-04, the second on 2016-10-16. The purpose of these experiments was to monitor the behavior of live DarkComet operators in realistic machines by executing the samples obtained in Section 3.3.1. Over the course of the two experiments, the executions of 1,165 unique samples resulted in 2,747 total runs for analysis.

Sample Selection

Samples whose controllers have responded to our scanner more recently are ranked higher in our sample submission priority queue. We submit the top $\mathscr{N}$ samples that: (a) have not previously been submitted for live analysis more than $\mathscr{M}$ times, (b) have responded to a probe in the past $\mathscr{T}$ seconds, and (c) have disjoint controller addresses. $\mathscr{N}$ is the total number of honeypots available, while $\mathscr{T}$ and $\mathscr{M}$ were determined experimentally to be 3,600 and 3,

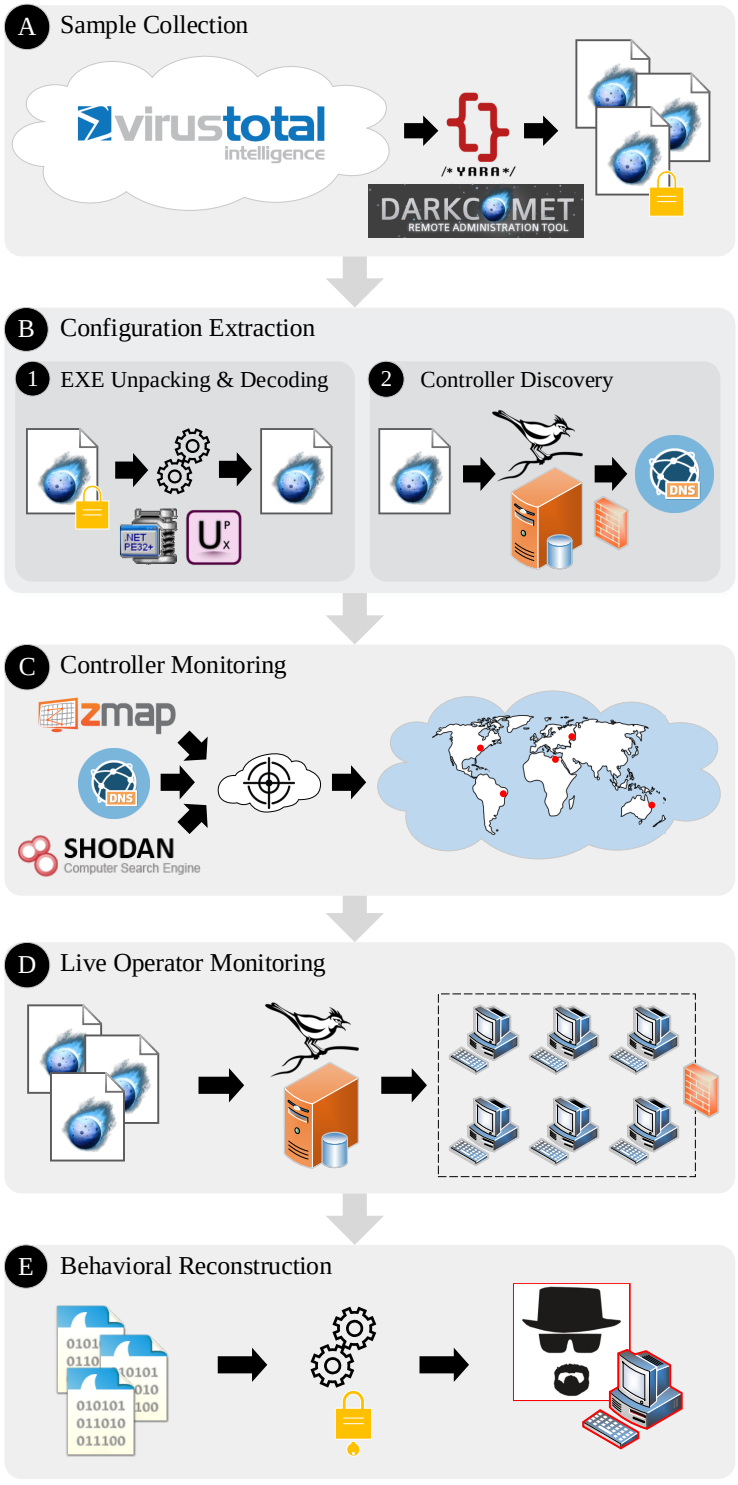


Figure 4.2. Visual summary of complete experiment workflow.

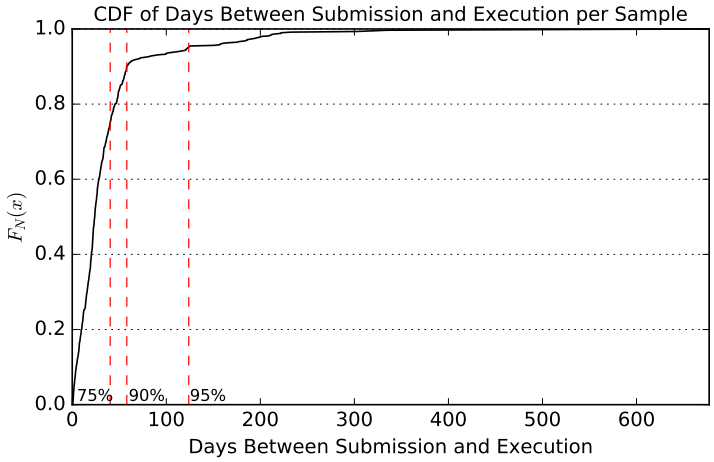


Figure 4.3. A CDF showing the relative age of our samples, where age is the number of days between submission to VirusTotal and execution in our sandbox. We report on 596 unique samples for which (1) we have submission information, and (2) resulted in manual operator interaction. 75% are executed within 40 days of submission, 90% in 58 days, and 95% in 124 days. The oldest sample that resulted in manual interaction was an astonishing 677 days old; the newest, 12 hours.

respectively.

Knowning the relative age of the samples we executed, meaning the difference in time between when they were received by VirusTotal and when we executed them, is important to interpret the measurements in this study. Figure 4.3 shows the ages of all samples which resulted in manual interaction with our honeypots.

Honeypot Design

The honeypots to which we submitted samples for operator monitoring were designed by my collaborator, Mohammad Rezaeirad. I will describe them here, so as to understand our results; however, their design is his contribution.

In Experiment 1, we ran 20 honeypots concurrently. The honeypots were identical, save minimal cosmetic differences, and did not ellicit radically different responses from operators. In Experiment 2, we ran only 8 honeypots concurrently (per new system limitations), but each with a carefully designed, unique persona:

- Control: Unmodified Windows installation.
- PC gamer (male).
- Medical doctor (male).
- U.S. political figure (female).
- Academic researcher (male).
- Bitcoin miner.
- College student (female).
- Bank teller.

The honeypots are all Windows 7 VMs deployed on VMware vSphere Hypervisor [135]. All installed browsers have full access to the Internet for realism. The results of each experiment are discussed in Section 4.4, while a comparison of the personas in Experiment 2 is shown in Figure 4.7 and discussed in corresponding Section 4.4.11.

We use the open-source Cuckoo Sandbox as our analysis platform [54]. Cuckoo includes a variety of malware-monitoring capabilities. We leverage these capabilities to (1) store all files created, deleted and downloaded during execution, (2) capture network traffic, (3) take screenshots upon screen buffer change, (4) counter anti-sandbox and anti-VM techniques, and (5) interface with many virtualization platforms (e.g. VMware vSphere). We defined and enforced a set of containment policies to minimize the possibility of our honeypots being used to launch attacks against systems not under our control. We implemented a restrictive policy in which our honeypots' gateways filtered through a transparent SSL interception proxy. The proxy allowed HTTP(S) traffic only if the user-agent string matched one of our installed browsers. We only allowed outbound traffic over HTTP(S), and rate-limited burst traffic. We also employed a firewall, dynamically whitelisting the controller IP addresses of the samples under analysis. Experiment 1 was conducted under this set of restrictions. We updated our containment policy for Experiment 2, allowing out all TCP and UDP traffic and removing the transparent SSL interception proxy. Instead, we deployed an IPS with DDoS prevention, SSL Blacklist rules on

outbound traffic [127], and burst and bandwidth limit enforcement.

4.3.2 Behavioral Reconstruction

The behavior of each RAT operator is reconstructed from the `.pcap` of the related trial. We decrypt the DarkComet traffic present in the file, and then use a signature engine build by my collaborator, Mohammad Rezaeirad, to decode DarkComet's protocol and extract the sequence of operator commands and relevant metadata.

Network Decrypter

Given a `.pcap`, we reassemble each TCP flow and then decrypt any DarkComet traffic using the encryption key we extracted from its sample's configuration. DarkComet uses RC4 for both network encryption and resource storage (e.g., RAT configurations). The encryption key is a combination of version number, a constant (depending on version), and an optional password set by the RAT operator.

Signature Engine

My collaborator, Mohammad Rezaeirad, developed a comprehensive set of DarkComet network signatures for versions 5.0+ using a combination of static analysis and exhaustive testing of DarkComet commands. His signature engine takes its input from the network decoder. If a network signature matches the decoded traffic, the engine returns the action performed (e.g., `Uploads & Executes Binary File`), the source of the action (controller or victim), the mode of the action performed (automatic or manual), and other tags for metadata extraction.

Remote Desktop (RDP)

DarkComet includes RDP functionality, allowing the operator to interact with the victim's OS directly. DarkComet RDP works by periodically sending the operator a JPEG-encoded image of the victim's screen, while the operator can issue mouse click and keyboard events to the victim machine. Our signature engine can decode operator interaction during RDP (click coordinates

and keystrokes), but at such fine granularity cannot infer the operator's actual actions on the victim machine. Rather, to fully reconstruct RDP sessions, we configured Cuckoo to capture and timestamp the screen any time it changed. We then manually annotated these images in order to reconstruct operator RDP sessions, using detected keystrokes and clicks as a filter on the screenshots.

4.4 Analysis

4.4.1 Operator Behavior Analysis: Overview

While measuring the global population of DarkComet controllers yields interesting results, it is, in fact, a secondary contribution; our primary contribution is to understand DarkComet operator behavior in the wild. To do so, we ran 1,165 unique DarkComet samples over the course of two, several-week-long experiments following the methodology described in Section 4.3. Overall, our experiments ran for nearly 2,400 combined hours, divided approximately equally between the honeypots executing in parallel. In all, the experiment accumulated 52.9 hours connected to a DarkComet controller. The average DarkComet session lasted about 4 minutes, while the average DarkComet session with RDP lasted about 7 minutes. In this section, we report what operators did during these sessions. In some cases, operator actions give us a clear indication of *motive* and *process*. In addition, we examine the actionable information we can glean about operators, and whether there are elements of our setup that hindered our ability to observe DarkComet operators. This section presents an analysis of those executions.

Table 4.1 provides a summary of the two experiments, broken down by unique subject and trial result. We monitored 2,747 interactive sessions overall. Of those sessions, 785 resulted in direct operator engagement with our environment. Almost 75% of samples that resulted in operator interaction were executed within a month of submission to VirusTotal, and a full 95% within four months. For more detail about the relative age of our samples, see Figure 4.3. Additionally, for information about the schedule by which we executed samples, see

Table 4.1. Summary statistics of the two experiments in the dataset. *Unique Samples* is the total number of unique samples executed; *Unique Controllers* is the total number of unique controllers to which the samples connected; *Total Runs* is the total number of individual executions we performed; *Total Connections* is the total number of trials in which the sample connected to a live controller; *Total Interactions* is the total number of trials in which an operator interacted with a honeypot; *Start Date* and *End Date* are the date range over which the experiment was conducted.

	Experiment 1		*Experiment 2*	
	Total	*Daily*	*Total*	*Daily*
Unique Samples	793	49.6	478	34.1
Unique Controllers	432	27.0	439	31.4
Total Runs	1,725	107.8	1,022	73.0
Total Connections	830	51.9	461	32.9
Total Interactions	531	33.2	254	18.1
Start Date	2016-05-04		2016-10-16	
End Date	2016-05-20		2016-10-31	

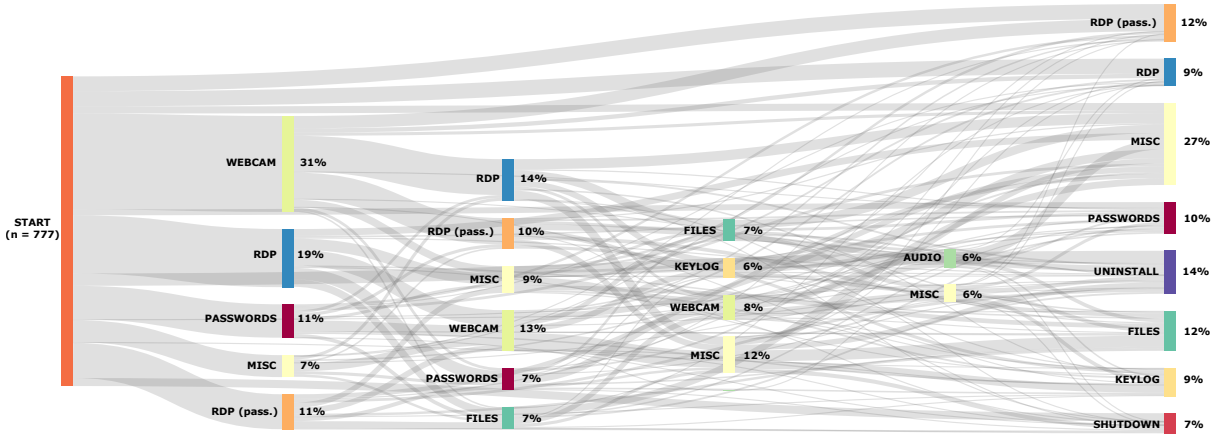
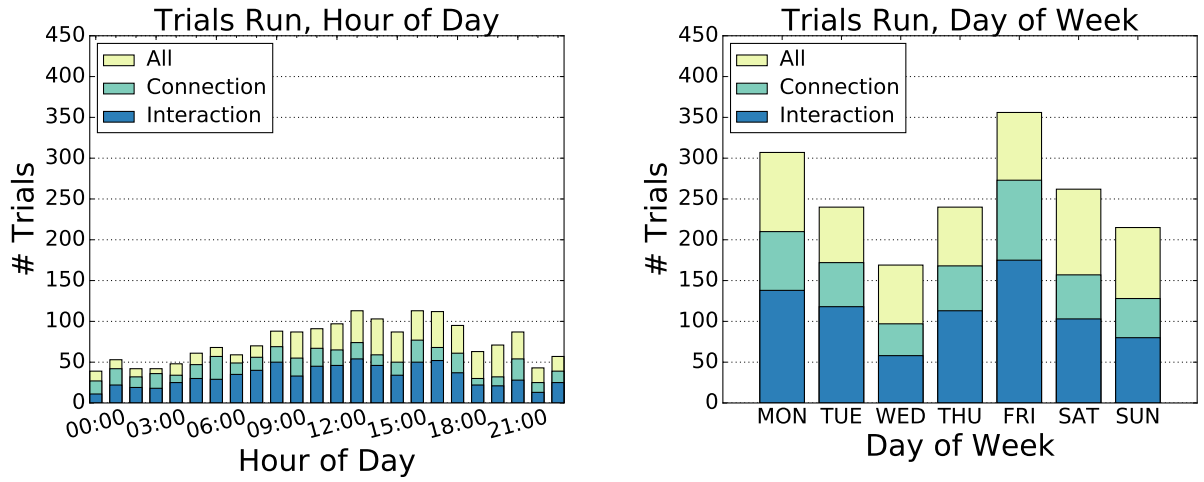


Figure 4.4. Composite flowchart of prevalent operator behaviors and sequences broken down by RAT interaction phase and category. Individual paths are labeled with the percentage of all executions that traversed that edge. Sequences occurring in fewer than 5% of engaged executions are omitted. The figure shows an operator preference for engaging with remote desktop or surveillance first in a majority of trials. See Section 4.4.3 for additional details.

Table 4.2. Categories of network signatures used in the experiment. These categories abstract the 151 network signatures present in the dataset. Prevalence of each category broken down by trials that contained remote desktop activity (367 trials), those that contained passive remote desktop sessions (287 trials), those without remote desktop (134 trials), and trials overall (788 trials). Remote desktop activity is defined as any remote desktop I/O event.

	RDP			
Category	*Act.*	*Pass.*	*None*	*Total*
AUDIO CAPTURE				
Record Microphone Sound	43%	15%	2%	26%
AUTOMATIC				
URL Download	–	–	1%	–
URL Visit	–	2%	1%	1%
IP Scanner	–	–	1%	–
Uninstall Server	–	–	1%	–
COMPUTER POWER				
Restart	5%	1%	1%	3%
Shutdown	4%	2%	4%	3%
Poweroff	1%	–	1%	1%
Logoff	1%	–	1%	–
FILE MANAGER				
Explore Files	52%	36%	15%	40%
Upload File	14%	5%	2%	9%
Download File	12%	7%	–	8%
Delete File	7%	3%	–	5%
Modify File	4%	2%	–	3%
Search for Files	2%	–	–	1%
FUN FUNCTIONS				
Fun Manager	19%	3%	2%	10%
Chat	15%	1%	2%	8%
Message Box	11%	1%	–	6%
Piano	11%	2%	1%	6%
Microsoft Reader	4%	–	1%	2%
MISC FUNCTIONS				
Clipboard	2%	1%	–	2%
Print Manager	1%	–	–	–
MSN FUNCTIONS				
MSN Control	2%	–	–	1%
MSN Contacts	1%	1%	–	1%
NETWORK FUNCTIONS				
Browse Page	7%	1%	–	3%
LAN Computers	5%	3%	–	4%
WIFI Access Points	5%	2%	–	3%
URL Download	4%	1%	1%	3%
Active Ports	3%	1%	–	2%
URL Visit	3%	1%	–	2%
IP Scanner	2%	1%	1%	2%
Net Gateway	2%	1%	–	2%
Network Shares	1%	1%	–	1%
DDOS Attack	1%	1%	1%	1%
SOCKS5 Proxy	1%	–	–	–

	RDP			
Category	*Act.*	*Pass.*	*None*	*Total*
PASSWORDS DATA				
Stored Passwords	51%	36%	31%	43%
uTorrent Downloads	9%	1%	–	5%
REMOTE MSCONFIG				
Services Startup	4%	2%	–	2%
Registry Startup	1%	1%	–	1%
REMOTE SCRIPTING				
HTML or VB Scripting	4%	2%	1%	3%
Binary	3%	1%	10%	4%
Batch Scripting	1%	–	–	1%
URL	1%	–	1%	–
RESTART SOCKET				
Server	–	–	–	–
SERVER ACTIONS				
Remote Edit Server	2%	2%	1%	2%
Restart Server	2%	2%	–	1%
Lock Computer	1%	–	–	–
Upload and Execute	–	–	–	–
SERVER REMOVAL				
Uninstall Server	7%	17%	27%	14%
SERVER SHUTDOWN				
Close Server	5%	6%	16%	7%
SPY FUNCTIONS				
Keylogger	41%	27%	13%	31%
SYSTEM FUNCTIONS				
Process Manager	20%	7%	4%	12%
Remote Shell Start	9%	3%	1%	6%
Remote Shell Stop	9%	3%	1%	6%
Windows List	8%	1%	–	4%
Remote Shell Command	7%	1%	1%	4%
Uninstall Applications	5%	1%	–	3%
Remote Registry	5%	1%	–	3%
System Privileges	4%	1%	–	3%
Hosts File	2%	1%	–	1%
SYSTEM INFO				
Computer Info	11%	5%	–	7%
Trace Map	5%	1%	–	3%
UPDATE SERVER				
From File	1%	1%	2%	1%
From URL	–	–	–	–
WEBCAM CAPTURE				
Attempt Webcam Access	76%	64%	16%	61%



(a) Number of samples run on the particular hour of the day.

(b) Number of samples run on the particular day of the week.

Figure 4.5. Malware sample submissions by day and by hour. Colors from colorbrewer2.org [24].

Section 4.4.2.

DarkComet provides two different ways of interacting with the victim machine: directly using RDP or indirectly using DarkComet commands. Table 4.2 reports observed commands separately by sessions where there was RDP activity (47% of sessions), where RDP was started but there was no interaction (36% of sessions), and where RDP was not started (17% of sessions) in columns *RDP Act.*, *RDP Pass.*, and *RDP None*, respectively. The rest of this section outlines various facets of operator behavior, from actions performed to engagement.

4.4.2 Sample Execution Schedule

Figure 4.5 shows the total number of trials that were run by hour of the day and day of the week, respectively. Note that we only ran samples whose controllers were determined to be online; because of this, Figure 4.5a matches our scanning data in Section 3.5.3 quite nicely. Figure 4.5b bucks this trend, instead likely demonstrating the rate at which new samples arrive at, or are made available by, VirusTotal.

4.4.3 Common Actions

Table 4.2 shows detailed information about all operator activity broken down by RDP status and activity category. The most common actions we observed across all three session types were webcam monitoring, password theft and file exfiltration. Operators attempted to access our webcam in 61% of all trials. Stored passwords were grabbed in 43% of all trials. The victim filesystem was explored in 40% of all trials. Other types of user monitoring were prevalent as well, with operators attempting audio capture and keylogging in 26% and 31% of trials, respectively. The prevalence of actions attempting to collect physical information about the user as well as their files suggests that surveillance is a dominant use of DarkComet, and, in fact, its intended use.

Some operator actions tended to occur more frequently with or without RDP. For example, webcam capture occurred in 76% of active RDP runs, compared to only 16% of non-RDP runs. Conversely, operators attempted to uninstall the malware stub in 7% of RDP runs compared to 27% of non-RDP runs. Operators checked the running processes of the victim in 20% of RDP runs when compared to only 4% of non-RDP runs. Also, a binary was dropped in only 3% of the RDP runs as compared to 10% of non-RDP runs, showing us that DarkComet is also used in an automated fashion to drop malware. These differences continue to lesser degrees across many other actions. The contrast in actions between RDP and non-RDP runs may indicate a variety of uses of DarkComet depending on operator goals.

While Table 4.2 shows aggregate statistics of operator actions, it says nothing about the *order* in which the actions take place. Figure 4.4 is a Sankey diagram showing the sequences of the actions most reported in Table 4.2. Each node shows the percentage of total trials that perform the corresponding action at that point in their sequence. For example, 14% of trials initiate an active RDP session as their second action. All percentages are based on the total trials.

One of the main takeaways from this diagram is that most trials are comprised of a subset of these very common actions: webcam access, remote desktop, password theft, file exploration,

and audio capture and keylogging. This is perhaps one of the most prominent indicators of our operators' motivations, which we discuss further in Section 4.4.13.

Other findings: A non-trivial number of trials interact with the victim machine solely through RDP. A full 14% of operators uninstall the stub as their final action, discussed more in Section 4.4.12. 31% of operators attempt to access the webcam before any other action, despite the fact that a webcam light could expose them to an online victim; the desire to view the victim environment is that great, whatever the motivation.

4.4.4 Data Collection

An operator conducting reconnaissance against a newly infected system has near-total access to the system through the DarkComet GUI, including full filesystem and registry access; lists of open windows, startup services, processes, and applications; and a plethora of network interrogation tools.

As we observe in Table 4.2, 40% of live trials involve filesystem exploration, roughly the same amount as all other forms of reconnaissance combined. Further, 8% of trials involve downloading files from the honeypot to the operator's machine, across 600 distinct events.

Operators that downloaded files exhibited certain patterns. 34% collected files from the desktop. 8% collected the 'My Documents' folder. And in instances where it was available, 33% stole a bitcoin wallet.

DarkComet also offers a variety of other data exfiltration tools beyond downloading files, such as automated functionality for collecting and downloading passwords, user account information, keylogger logs, and clipboard text. Table 4.2 shows us that, although only 8% of trials involve the downloading files or directories, a full 43% do password collection, and 31% attempt to collect user keystrokes using the keylogger.

After Experiment 1 indicated such high levels of password collection, we decided to provide actual account credentials in Experiment 2 to monitor access attempts. We created a number of accounts for each persona at popular websites, saving the credentials in the browser.

Table 4.3. Results of monitoring access attempts to the online accounts in our honeypots (only in Experiment 2). We used two-factor authentication both to prevent account abuse and to record each access attempt, the language of the machine attempting access, and the exact time of the attempt. Recall the experiment ran from 10-16-2016 to 10-31-2016.

Account	*Date*	*Language*
Twitch	10-18-2016	English
Google	10-18-2016	English
Google	10-20-2016	*Unknown*
Google	10-20-2016	Russian
Twitch	10-24-2016	English
Google	10-24-2016	Turkish
Google	10-25-2016	Turkish
Google	10-27-2016	Turkish
Google	10-28-2016	Russian
Yahoo	10-29-2016	English
Yahoo	10-29-2016	English
Steam	10-29-2016	English
Reddit	10-30-2016	English

We protected each with two-factor authentication, both to prevent account abuse and to record each access attempt, the language of the machine attempting access, and the exact time of the attempt. Over the course of the experiment, we observed 13 access attempts: nine attempts for two popular email services, three for two different gaming-related services, and one for a social media site. Of these attempts, three were made from machines configured for Turkish, two for Russian, and the remainder English.

4.4.5 Dropped Files

In 28 of our trials, we observed files being dropped by the operator. We consider only files dropped explicitly by the operator during interaction, not embedded files automatically dropped during sample execution. In 29% of these trials, an operator performed only the single action of dropping a file, akin to a traditional malware dropper. In 18% of trials, we observe that an operator dropped a file after checking the webcam of the victim and deploying a keylogger. It is interesting to note that almost an equal number of trials start with webcam as the first

action followed by dropping files; this suggests RAT operators are interested in person-to-person interaction as much as dropping more malicious files.

Over the course of the study, operators dropped 48 unique files. 34 (71%) were executables, 19 of which were not previously seen by VirusTotal. 35 were RAT stubs, 13 being DarkComet stubs with new configurations, 5 njRAT, 3 NetWire, and the remainder custom RATs and worms. Operators also dropped 5 distinct scripts - `HTML` executables, Batch files - for executing destructive actions on the host machine.

4.4.6 Visiting URLs

DarkComet operators visited 123 non-unique URLs from 33 unique domains during the course of our experiment. Out of these URLs, 26 contained adult content, 13 gaming, seven personal blogs, six streaming, and five internet pranks. The remaining URLs were VPN, search, banking, social, education, IP address tools, and file storage websites. 23 URLs generated a 404 error and one had an unregistered domain. 20 of these URLs were written in a language different from English: Nine in Russian, six in Turkish, and five in Japanese. Finally, operators performed 18 visits to eight unique IP addresses (five of them in the same $\backslash 24$ subnet).

4.4.7 Command Line Activity

DarkComet provides a command line interface, allowing operators to interact directly with the victim machine. Our signature engine extracts all commands issued by the operator from the command shell buffer. Inspecting the 92 commands across all trials, we classify the operators' intentions as such:

Reconnaissance

60% of commands pulled information on system configuration, file system, and network configuration.

Manipulation

26% launched or terminated processes.

Destruction

10% damaged the host machine or filesystem (e.g., "`FORMAT C:`").

Concealment

3% hid operator actions (e.g., "`ECHO OFF DEL *.*/Q DEL:VIRUS.BAT`").

4.4.8 Operator Interaction with the User

RAT infections are personal affairs. Not only does a human operator interact individually with each victim machine, but they often also monitor the machine's user as well. Indeed, the image most commonly associated with a RAT infection is that of an attacker watching a victim through their machine's webcam, an image well supported by historical anecdotes [6, 29]. There are a variety of motives that would compel a ratter to interact with their victim (passively or actively), and DarkComet provides an array of capabilities to do them all.

Surveillance

Victim surveillance is, perhaps, the most notorious activity associated with RAT infections. DarkComet offers access to live feeds from the infected machine's webcam and microphone. Though our honeypots offer neither, we do detect attempts to access both. As is shown in Table 4.2, attempts to access these are prevalent, with 61% of sessions attempting to access the webcam, and 26% the microphone.

Operators which engaged in remote desktop tended to use webcam and microphone monitoring more than their counterparts. 76% of remote desktop users accessed the webcam, compared to 16% of non-remote desktop users. Likewise, 43% of remote desktop users accessed the microphone, compared to 2% of non-remote desktop users. This correlation further indicates

Table 4.4. Motivations of all communications received from operators during live trials.

Motivation	*Cnt*	*Pct*
Harassment	327	53%
Misdirection	97	16%
Recognition	57	9%
Extortion	15	2%
Unknown	123	20%
Total	619	

that operators using remote desktop are more engaged in the RAT session overall, as user monitoring and remote desktop are both time-intensive activities.

Direct Communication

Beyond passive surveillance, DarkComet provides operators several means to communicate with the victim directly. These include opening a two-way chat client on the victim's screen, displaying a pop-up alert message, sending text to the victim's printer, and reading messages aloud using Microsoft Reader. We observed operators attempting to communicate with the victim through each medium, as per the "Fun Functions" section of Table 4.2. We classify operator communications with four categories:

Harassment

53% of communications were intended to harass the victim. Threats, attempts to induce fear, and heavy sexual innuendo comprise the majority of victim harassment (e.g., "`YOU ARE NOT ALONE, I SEE AND HEAR YOU`").

Extortion

2% of communications were intended to exact a ransom from the user in return for control of the machine. As our victims could not respond to the operators, none of these demands persisted. Further, we suspect that some portion of harassment leads to extortion in scenarios where a victim is actively replying (e.g., "`HI <redact> WANNA PAY ME $50 TO NOT SHARE`

ALL UR IMFORMATION TO THE WORLD.... EVERYTHING U EVER BEEN ON UR PC").

Misdirection

16% of communications were legitimate-looking message boxes, either to cover the installation of the RAT (in the case of error messages), or to gain a victim's trust and "phish" them (e.g., "THIS IS MICROSOFT (TM)" or "YOU MAY BE AT RISK OF A VIRUS PLEASE LOG OUT OF ALL GAMES AND BANKING SITES ON INTERNET").

Recognition

A full 9% of communications were operators posting their hacker groups and tags, taking responsibility for the "hack." Though juvenile, this provides potentially useful attribution information (e.g., "HACKED BY #ZONED OUT XDDDDDD").

We used Google's language detection API on our operators' communications, but ethereal text like IM chat generates mostly misclassifications. See Section 4.4.10 for results.

Visibility

This level of interaction with the victim highlights an interesting phenomenon: a significant portion of our operators choose to be blatantly visible to the victim, either through direct communication or via actions that manipulate the victim's screen. We can categorize operators by their degree of visibility; specifically, whether they are *intentionally* visible or not. Visible actions comprise any action the operator initiates that is visible to the victim (e.g. opening a chat window), and exclude more discreet actions that could still be detected (e.g. modifying a file's attributes).

We observe that, of the 785 total engaged trials, 62% are visible to the victim. Lack of discretion is often an indication of unsophisticated operators, and we have some confidence that this is the case. However, lack of discretion does not imply harmlessness. Of communications directed at the victim, 2% were extortion demands and 16% were attempts at deception. These operators used visibility intentionally to intimidate or deceive the victim to some goal.

4.4.9 Remote Desktop Sessions

One of DarkComet's most heavily used features is remote desktop (RDP), with 654 total trials (83% overall) containing remote desktop sessions. RDP offers the operator immersive control of the victim, particularly the ability to interact with programs that require GUI interaction; however, the tradeoff is that the operator's actions are completely visible to the victim.

Though visible to the user, remote desktop functionality has been abused by criminals. In the 2016 TeamViewer compromise, attackers hijacked TeamViewer clients (which *only* provide remote desktop) and made purchases with victims' eBay and PayPal accounts late at night to avoid being seen [49]. Our operators' tendencies toward remote desktop usage for user data theft should not be discounted.

Using the methods described in Section 4.3.2, we found that 44% of RDP sessions had no operator input. In this section, we report on the 367 sessions that had *active* RDP engagement; that is, where the operator actually interacted with the victim via RDP. We manually inspected each RDP session to determine the operator's actions, the results of which are summarized in Table 4.5. These findings are complementary to the data in Table 4.2.

Hacking Tools

The deployment of additional hacking tools occured in 4% of our trials. We suspect these trials use remote desktop out of necessity, as the tools deployed were GUI-based. We observed the following: Android mobile phone RAT builders & infectors; webcam and Skype screen recording tools; spam bots; YouTube view-fraud bots; DLL injectors; and the Havij SQL injector. Some of these tools were bundled with the DarkComet stub. We were surprised to find operators using these tools in plain view of the victim, indicating that the operators felt there was no need for secrecy.

Masquerading as Legitimate Programs

A pattern emerged in the remote desktop screenshots which followed pre-existing RAT literature: the bundling of DarkComet with legitimate tools. We found that 5% of remote desktop users pack their DarkComet samples with legitimate software programs.

We also found a number of samples impersonating software programs like Adobe Photoshop and WinRAR, as well as utilities like .NET Framework Setup Cleanup Utility and even Avast Antivirus. Cracked versions of these programs were distributed bundled with DarkComet, suggesting that the initial DarkComet infection vector was pirated software distribution.

The programs our samples most commonly impersonated are gaming tools: ping checkers, cracked versions of popular games (`HackFifa.exe`), game enhancements (Enhanced Minecraft Shaders), and "cheat" tools to gain an unfair advantage in online games. We discuss campaigns against the gaming community further in Section 4.5.2.

User Data Collection

Table 4.2 indicates that our operators tend to target user data, with 43% of trials collecting passwords, 31% using a keylogger, and 41% exploring the filesystem in depth. It comes as no surprise that operators use remote desktop to the same end, with 63% of active remote desktop sessions being used for user data collection.

Operators leveraged built-in browsers to access user accounts, browsing history, and download history. We suspect this is a combination of data collection and gauging level of interest in the victim machine. Most operators checked Chrome and Firefox, and some investigated Internet Explorer. Some operators navigated to popular Web sites. We strongly suspect this is to determine if the user has an active session, and, if so, to hijack the session. The websites most commonly targeted were social media sites: Facebook.com and VK.com (a popular Russian social media platform).

Operators also accessed installed applications, Dropbox and Steam in particular. In fact, 33 of the 48 trials that accessed applications targeted Steam, lending more credibility toward the

gaming campaigns discussed in Section 4.5.2.

Finally, operators inspected system information, typically exhibiting one of two behaviors: checking WiFi and network connectivity settings, or using the DirectX diagnostic tool. Both behaviors appear to be system vetting, especially DirectX, which even reveals the presence of virtualization.

Display of Illicit Material

We noticed that 4% of active remote desktop sessions involved the display of online video pornography, clearly visible to the victim user. There are multiple motivations behind this action. A subset of operators use pornography to enhance their chat interactions with the victim, threatening to continue displaying porn until demands are met or intimidating the victim. We speculate another subset simply use pornography to harass the victim, given the high rates of harassment and "Fun Function" usage in Table 4.2. The display of pornography likely indicates an operator's short-term motivations, given its striking visibility.

Alternative Communication Methods

As a final note of interest, we observed several alternative methods of communication with the victim specific to remote desktop. Operators communicated using Notepad, Paint, the command line, Word documents, and uploaded images. Overall, 6% of remote desktop trials involved some form of alternate communication.

4.4.10 Dynamic Language Analysis

We attempted to use Google's language detection API on our operators' communications (and other metadata), the results of which are shown in Table 4.6. Though English is understandably the top language, the presence of languages like Igbo indicates the shortcomings of applying language detection to ethereal text like IM chat messages. For reference, Igbo is a language spoken in Nigeria, with which the language detection API appears to frequently confuse Turkish slang.

Table 4.5. Common patterns of behavior exhibited by operators employing active remote desktop sessions.

Behavior	*Count*	*Percentage*
User Data Collection	360	63%
System information	170	29%
Browser data	72	12%
Application and Online account data	64	11%
Filesystem exploration	54	9%
Targeting of Gaming-related Data	52	9%
Alternative Communication Methods	37	6%
Masquerading as Legitimate Programs	30	5%
Illicit Material	23	4%
Deployment of Hacking Tools	23	4%
Suspected YouTube View Fraud	20	3%
Personally Identifiable Information	6	1%

Table 4.6. Languages of metadata obtained during live trials, including chat messages, remote desktop keystrokes, and filenames of dropped files. *Other* is any other spoken or written language.

	Chat		*RDP Keystrokes*		*Dropped Files*	
Source	*Cnt*	*Pct*	*Cnt*	*Pct*	*Cnt*	*Pct*
English	265	42%	27	21%	19	13%
Igbo	22	3%	0	-	0	-
Hawaiian	18	2%	0	-	0	-
Turkish	15	2%	5	3%	0	-
Corsican	15	2%	1	-	0	-
Other	31	5%	19	15%	2	1%
Undetermined	252	40%	74	58%	123	85%
Total	618		126		144	

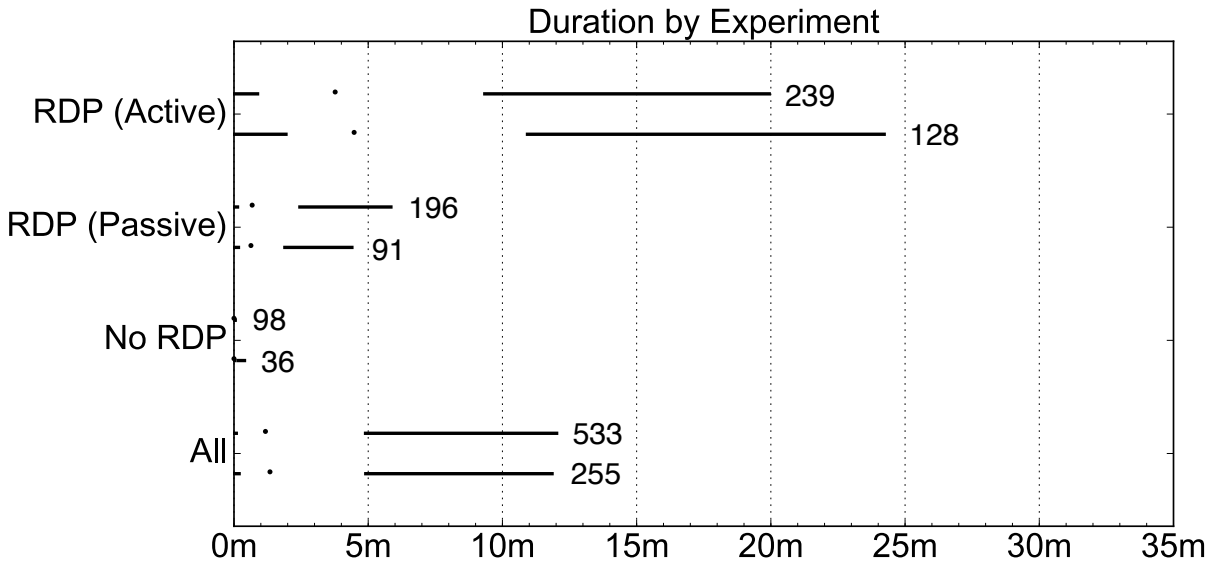


Figure 4.6. Comparison of the time spent in the honeypots by operators in the first and second experiments, in minutes. Operators are distinguished by their use of RDP, as per Table 4.2.

4.4.11 Operator Time Engagement

Most of DarkComet's numerous functions must be manually executed by the operator. From the timestamps associated with these actions, we can develop an understanding of how long an operator engages with each infection, and how that varies by their own actions.

Metric for Determining Operator Engagement Duration

From enumerating all possible DarkComet actions and exploring their functionality, we identified all actions that require manual operator intervention. From our execution traces, we identified all such actions and calculated the time difference between any two adjacent manual actions. If the time difference between these two actions was less than 60 seconds, we considered this to be a single period of engagement. Using this metric we summed the complete engagement time for each trial, and report this as operator engagement.

Engagement Results

Figures 4.6 and 4.7 presents an overview of operator engagement throughout our dataset. Figure 4.6 compares the overall operator engagement times between the two experiments, while

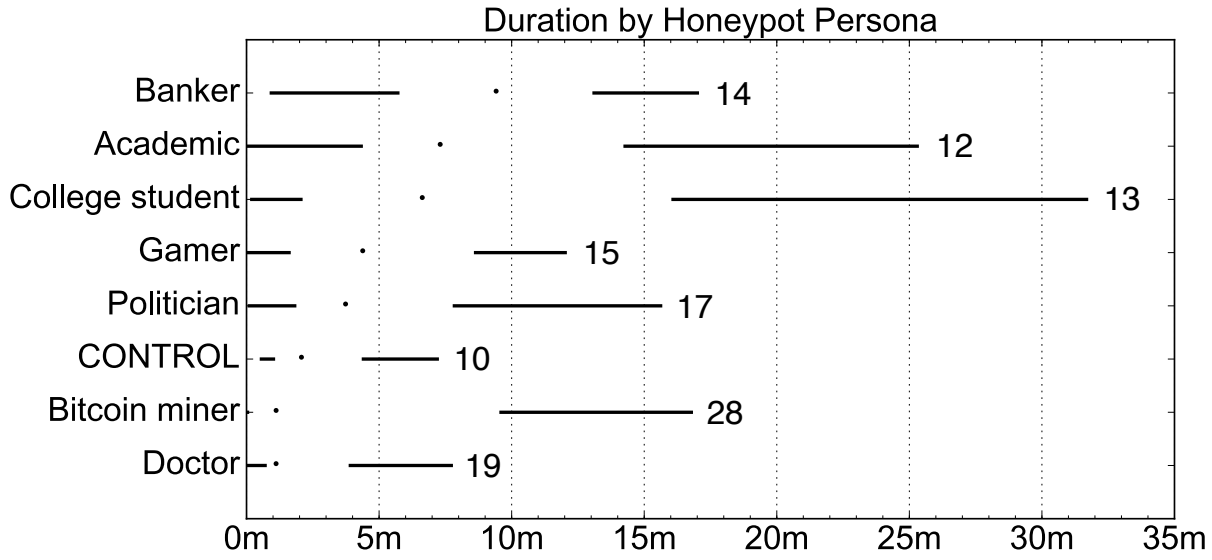


Figure 4.7. Comparison of the time spent between honeypot personas (in only the second experiment), in minutes. Only sessions with active RDP are reported, as these are representative of the whole. Operators are distinguised by the honeypot persona with which they interacted.

Figure 4.7 breaks down engagement time in Experiment 2 by honeypot persona (for active RDP users).

In Figure 4.6, we observe the operators' engagement time with the honeypots, as determined by our metric. The two series represent the two experiments, each of which is broken down by RDP usage, as in Table 4.2. The average overall engagement time in both experiments is 4 minutes.

The extent of operator engagement varies significantly between RDP category. Trials with active RDP had at least 60% more actions and engagement than average across both experiments, while trials with no RDP activity at all showed almost no activity - an average engagement below 20 seconds for both experiments. These differences indicate that the primary way operators manually engage with victims is via RDP, despite using many non-RDP features as well. As such, RDP usage appears to be an indicator of operator engagement and interest in the victim.

In all RDP categories, operator engagement also saw high variance, with standard deviations exceeding the mean in all cases. All categories also saw several large duration runs skew the average significantly. This indicates a wide variety of operator interests, ranging from

cursory to extreme. This non-trivial level of operator engagement has implications for RAT defense, discussed in Section 4.5.1.

Influence of Honeypot Persona

Piqued by the results of Experiment 1, we decided to measure the affect of honeypot persona on operator engagement. Figure 4.7 provides a comparison of operator time engagement across the 8 honeypot personas in Experiment 2. We display only the results of active RDP sessions, which constitute truly engaged operators.

The honeypots (recall Section 4.3.1) are displayed in order of increasing median: doctor, bitcoin miner, *control*, politician, gamer, college student, academic, banker. The results are quite distinct. The three personas that exceed the mean (college student, academic, banker) show engagement nearly three times greater than the control. It is difficult to say exactly what factors specifically contributed to the engagement difference, but one factor that appears to play a prominent role is *file system depth*. An analysis of actions by persona indicates that file system exploration is more prevalent in the three more engaged personas. Further, these personas were designed with more detailed file systems than the *control* or bitcoin miner, for instance. Overall, it seems that the design and depth of the victim machine's filesystem directly affects operator engagement time, though a more regimented study focused on just this aspect is necessary to confirm our hypothesis.

4.4.12 Operator Final Action

In Sections 4.7 and 4.7, we noted that a number of factors would very likely affect our results; in particular, the engagement results just reported in Section 4.4.11. One prominent factor was our lack of realistic user data streams: webcam, audio, and user chat engagement. Another was our inability to exclude targeted operators from our sample source, operators whose targets we had no way of emulating properly.

In an attempt to determine what factors ultimately did cause the operator to leave our

Table 4.7. Last operator actions before session termination for trials with manual interaction. Actions that appear in less than 1% of trials are omitted.

Category	*RDP*			
	Act. (n=367)	*Pass. (n=287)*	*None (n=131)*	*All (n=785)*
REMOTE DESKTOP, ACTIVE: Remote Desktop I/O	32%	-	-	15%
SERVER REMOVAL: Uninstall Server	7%	16%	27%	14%
REMOTE DESKTOP, PASSIVE: Remote Desktop Start	-	30%	-	11%
FILE MANAGER: Explore Files	7%	10%	8%	8%
REMOTE DESKTOP, ACTIVE: Remote Desktop Start	15%	-	-	7%
REMOTE DESKTOP, PASSIVE: Remote Desktop Stop	-	16%	-	6%
REMOTE DESKTOP, ACTIVE: Remote Desktop Stop	12%	-	-	6%
PASSWORDS DATA: Stored Passwords	2%	6%	15%	6%
SPY FUNCTIONS: Keylogger	3%	6%	9%	5%
WEBCAM CAPTURE: Attempt Webcam Access	3%	5%	6%	4%
Session termination was only manual action	-	-	17%	3%
AUDIO CAPTURE: Record Microphone Sound	3%	1%	2%	2%
REMOTE SCRIPTING: Binary	-	-	9%	2%
FUN FUNCTIONS: Fun Manager	2%	1%	-	1%
FILE MANAGER: Download File	-	2%	-	1%

Table 4.8. Last operator actions before uninstalling malware for trials with manual interaction. Actions that appear in less than 1% of trials are omitted.

Category	*RDP*			
	Act. (n=25)	*Pass. (n=45)*	*None (n=36)*	*All (n=106)*
Session termination was only manual action	-	-	83%	28%
REMOTE DESKTOP, PASSIVE: Remote Desktop Start	-	47%	-	20%
FILE MANAGER: Explore Files	4%	18%	3%	9%
REMOTE DESKTOP, ACTIVE: Remote Desktop I/O	32%	-	-	8%
PASSWORDS DATA: Stored Passwords	4%	9%	6%	7%
WEBCAM CAPTURE: Attempt Webcam Access	8%	4%	6%	6%
REMOTE DESKTOP, PASSIVE: Remote Desktop Stop	-	13%	-	6%
SPY FUNCTIONS: Keylogger	4%	4%	-	3%
REMOTE DESKTOP, ACTIVE: Remote Desktop Stop	12%	-	-	3%
SERVER SHUTDOWN: Close Server	8%	-	-	2%
REMOTE DESKTOP, ACTIVE: Remote Desktop Start	8%	-	-	2%

honeypot, we analyze the last actions performed by the operator before disconnection. These actions may be indicative of the factor(s) that dissuaded the operators from continuing the infection, or of their motivation provided they accomplished their goal prior to disconnection.

Table 4.7 details the last actions performed by operators before session termination. Unexpectedly, a full 14% of operators actually *uninstall* the malware prior to disconnection. Thus we also provide Table 4.8, which details the last actions, prior to uninstallation, of those operators that uninstalled the malware.

Remote Desktop

45% of all operators were engaged in RDP prior to leaving. Recall from Section 4.4.9 that 63% and 29% of RDP activities are data collection and system information collection, respectively. As these operators do *not* uninstall the malware before leaving, we suspect they have deemed the honeypot at least somewhat viable as a long-term victim; however, drawing conclusions from RDP sessions is difficult given our manual inspection limitations.

System & User Information

More interesting are the operators that did not engage in remote desktop. 17% of all operators disconnect directly after attempting to access user information: passwords, keylogger, webcam, and microphone. We suspect the absense of user presence in these cases dissuades them. Another 9% disconnect after file system exploration, though only 1% actually download anything; once again, this is indicative of loss of interest due to banal file system contents. While none of these operators uninstalled the malware, (as we will discuss next), it seems probable that our honeypots did not present the stimuli they were seeking.

Malware Uninstallation

A shocking 14% of all operators uninstall the malware before disconnection. Table 4.8 elucidates this. 28% of operators that uninstall the malware do nothing else; it is their sole action. This may be an indication of targeted attacks, to which our machine is a nuisance, or

of sinkhole operations that automatically uninstall infections. Other operators that uninstall follow the previous trends: 39% after RDP, 9% after file exploration, 16% after user investigation (webcam, passwords, keylogger). Apparently motivated by access to interesting user data - credentials, files, or user data streams - these operators either gathered all interesting data in a single session, or (far more likely) were unimpressed by the lack of realism in the honeypot. Our discussion of lessons learned in Section 4.5.3 features these results.

4.4.13 Operator Motives

One of our motivating questions in carrying out this work has been to determine *why* an attacker would compromise a machine and install a RAT on it and *what* an attacker might do after compromise. Although motive is impossible to infer with perfect accuracy, certain operator actions betray attacker intent. Here we consider three potential attacker motives. These motives are not mutually exclusive, and an attacker may be driven by more than one of these.

User Reconnaissance

RATs are unique among malware in that they allow the attacker to interact with a user, and we suspect that this is the primary motivation for many operators. Among the sessions we observed, 18% attempted to harass or extort the user. Furthermore, in 41% of interactive sessions, the operator attempted to access personal information about the user in the form of pictures or documents. (We exclude attempts to collect user credentials like password or cookies from this count.) Together, these two categories indicate that **at least 45% of sessions were motivated by access to a human user.** We note that 63% of sessions attempted to access a webcam or microphone. Unfortunately, our honeypots were equipped with neither, so we cannot determine whether an operator was accessing these devices because he wanted to see and hear the user or simply to gather information about the machine or confirm the identity of the victim.

Credentials

Probably the most easily monetized resources on a compromised PC are user credentials. In 50% of observed sessions, the operator attempted to access files containing credentials, and in 31% of sessions the operator installed a keystroke logger. There were attempts to steal bitcoin wallets and to grab data from an installed Steam account, and we recorded several attempts to access stolen accounts (Section 4.4.4). This leads us to conclude that **at least 58% of RAT operators were motived by access to user credentials.**

Vantage Point

In many cases, a RAT can serve as a valuable vantage point for an attacker to launch other attacks or spread laterally through an organization. We consider an attacker to be motivated by the vantage point of the victim if he attempts to perform any network actions beyond testing network connectivity. These include scanning the network (Section 4.4.7), attempting to launch network attacks (DDoS in Table 4.2), deploying hacking tools (Section 4.4.9), and perpetrating view fraud (Table 4.5). In all, **16% of sessions exhibited some behavior that exploited the victim's vantage point.**

4.5 Discussion

DarkComet is a versatile tool, giving the operator a rich menu of actions to carry out on a remote machine. We were surprised, therefore, to find 47% of sessions involved RDP use, which reveals the presence of the operator to the victim. We expected even amateur operators to try to stay undetected on the machine in an attempt to obtain as much information as possible over time. The large number of RDP sessions and the actions we observed indicate that most operators are not trying to be stealthy. To the contrary, many actively sought to harass the user. This suggests that a substantial portion of operators are using DarkComet either for immediate amusement or with the hope of eventually extorting the user.

4.5.1 Honeypot as Tarpit Defense

The interactive nature of RATs means that attacks are limited by the number of operators available and the time spent interacting with a victim machine, so operator time may be a bottleneck for such attacks. We consider whether deploying multiple, *sufficiently realistic* honeypots could be used to draw operator attention from real targets and potentially even deter operators. We found that in our experiments, sessions with RDP activity lasted an average of about 4 minutes, whereas sessions without RDP lasted on average less than 20 seconds (Figure 4.6). Across all machines during both several-week-long experiments, we accumulated just over 52.9 hours of operator engagement, despite a total uptime of about 10,080 machine-hours. This means that operators were trapped in our "tarpit" for a dismal 0.5% of its total lifetime.

It is clear that our experiment itself did not have an appreciable impact on RAT operators' capacity for wrongdoing. According to the results in Section 4.4.12 and the general breakdown of actions in Table 4.2, it would appear that a major inhibitor to our honeypots' success as a tarpit is its realism. A more realistic target might attract operators for longer periods; however, creating such targets also burdens the defender. The findings in Figure 4.7 suggest this is true, but additional experiments are necessary to determine the effect of realism on operator engagement, and to conduct a cost-benefit analysis of using realistic honeypots as a defensive tarpit.

4.5.2 Honeypot as Threat Intelligence Sensor

Threat intelligence, broadly speaking, is any information about threats that might be operationally useful for a security practitioner. Like a conventional honeypot aimed at extracting information from a malware sample, a RAT honeypot can be used to extract information like command and control IP addresses and samples of additional malware dropped by the RAT. The hands-on nature of RATs allows us to observe an attacker at work. This includes information about what files are searched manually and what tools an attacker installs once he/she determines

that the machine is a real victim. A realistic honeypot can potentially extract information from an attacker beyond that extracted from the malware sample alone.

Though the amateur operators we monitored did not provide particularly fascinating intel, this study validates the capacity for honeypots to act as threat intelligence sensors. Our honeypots give us some insight into aggregate RAT operator demographics (with Russia and Turkey being well represented). We also obtained several malware samples not previously seen on VirusTotal, though they appeared to be updates and RATs of similar quality rather than more sophisticated malware. We witnessed the deployment of other hacking tools, and numerous indicators of the prevalence of attacks against the gaming community. Further, we observed the theft and attempted use of credentials belonging to a number of popular online services. Overall, our experiments illuminated a vibrant gamut of RAT operators engaged in hands-on exploitation of compromised machines.

4.5.3 High-Interaction Honeypots: Lessons Learned

Achieving realism in a high-interaction honeypot is difficult. Ideally, a honeypot would be completely indistinguishable from a real user's workstation; however, in practice this is exceptionally challenging to accomplish. Our experiments have shed light on a number of factors critical to future studies involving honeypots of a similar nature.

Cosmetic Appearence

Most RAT operators use RDP, whether to just inspect the desktop or to actively control the system. Given the overwhelming usage of RDP, the importance of providing a cosmetically-realistic honeypot cannot be overstated.

User Presence

This study confirms the anecdotal supposition that amateur RAT operators are often motivated by access to a live victim user, whether purely for recreational trolling or for more insidious means like blackmail, voyeurism, and sextortion. Indeed, some of the most common actions

performed by our operators are accessing the webcam, recording audio with the microphone, monitoring user keystrokes with the keylogger, and even initiating chat with the victim. No future study should neglect to provide video and audio feeds to the honeypot. Implementing some means of responding to chat communication should also be considered. We suspect that the former would result in higher operator engagement times, while the latter may prompt operators to unveil their motives in the form of chat-based threats and demands.

File System Depth

File system exploration was one of the primary actions across all operators, and represents the second most time consuming action behind RDP. In Section 4.4.11 we saw that personas with more detailed file systems occupied operators longer. Despite its importance, providing a realistic file system is one of the most unscalable challenges in creating realistic honeypots [124], and has been since the very inception of the live operator honeypot [129].

Credentials

10% of operators began by searching for stored passwords, and 43% did so overall. Given that credential theft is the goal of so many operators, providing credentials for them to steal provides multiple benefits. The visceral success of password theft adds realism to the honeypot and may keep operators engrossed. But more interestingly, this allows the seeding of "honey-credentials" to gather more information about attackers. Our trial with honey-credential seeding yielded numerous recorded access attempts.

4.6 Ethical Considerations

Unlike most types of malware analysis, this work entails interaction with human subjects: the RAT operators. Research involving human subjects imposes ethical obligations on us as researchers and requires additional institutional oversight to ensure that those obligations are met. We sought and obtained approval from the UC San Diego Institutional Review Board (IRB)

for this study.

In our experiments, the biggest concern is that exposure of Personally Identifying Information (PII) about the operator may cause harm to the operator. To minimize the risk of harm, the raw data was only analyzed by members of the research team that were part of the institution whose IRB approved the study, and was stored encrypted. All IP addresses and network traces collected were hashed after geo-location was performed using a local copy of the MaxMind GeoLiteCity database [90]. All further analysis was performed on the hashed IP addresses and sanitized network and API traces. The raw screenshots were transcribed and all PII was removed before performing additional analysis. No PII is included in this chapter and after publication all raw data will be deleted, with anonymized versions of the data saved for 2 years. All RAT operators voluntarily interacted with our honeypots and we made no attempt to obtain PII information from them or to actively identify individuals that connected to our honeypots.

The second concern is that our honeypots might be used as stepping stones for attacks targeting other systems not under our control. In order to mitigate this risk, our monitoring honeypots were configured with a conservative firewall containment policy that only allowed outbound connection to the RAT's controller and any other server that initiated a connection with the honeypot first. The one exception is that we allowed outbound HTTP and HTTPS (man-in-the-middled to check headers) traffic from our honeypots through our HTTP proxy server only if the user-agent exactly matched one of the installed browsers. The user-agent could be spoofed by an attacker, but we found no evidence of an attacker doing this in our dataset. We actively analyzed the network activity from our honeypots to detect if attacks were evading our containment and did not witness any attacks that were not blocked by our firewall containment. We feel that our containment methods and protocols for analyzing data minimized potential harms while allowing us to perform measurements that will benefit the security community with increased understanding of the behavior of manual attackers.

During the second round of RAT executions we deviated slightly from the approved IRB protocol in terms of our containment implementation. Instead of implementing a strict set of

firewall rules, we implemented an IDS that was integrated with our firewall and would block any malicious messages detected by the IDS. We have notified our IRB of this slight deviation and requested an IRB protocol amendment that would approve this new containment policy. Our analysis of the network traffic generated by operators indicated that this new containment implementation likely did not allow any harmful messages to be sent using our honeypots.

4.7 Limitations and Biases

The challenges involved in measuring low-volume attacks often result in observational biases. For example, the first studies on targeted attacks were performed in direct collaboration with the Tibetans [57], Uyghurs [78], and political dissidents in the Middle East [85]. As a result, each of them is fundamentally biased towards the threats targeting its respective community. Recent work mitigated that issue by leveraging VirusTotal for data collection and by scaling analysis to hundreds of thousands of samples uploaded by tens of thousands of users [77]; however, VirusTotal uploaders are probably not representative of all victims. In particular, the authors acknowledged that VirusTotal offered "a partial coverage of attacks where individuals and NGOs [were] likely over-represented" [77]. Our measurement of RAT operators' behavior similarly exhibits several main observational biases.

DarkComet

As we focus exclusively on DarkComet, our analysis is limited to operators using this malware family. We chose DarkComet for a variety of reasons. In continuous development since 2008, DarkComet has near comprehensive functionality compared to other RAT families. Additionally, or perhaps consequently, it has been employed by the wide range of actors cited in Section 2 and maintains long-standing popularity in hacking forums like Hackforums [56]. DarkComet was still in widespread use at the time of this work's publication, being a top-five RAT family according to VirusTotal submissions.

However, there remains the possibility that DarkComet is not representative of all other

RATs, and that we could be missing the behaviors of entire classes of actors that eschew DarkComet for whatever reason. As this study is focused on *amateur* RAT operator behavior, a class of operator with whom DarkComet is known to be popular, we accept this risk while acknowledging the potential for bias.

VirusTotal

Our collection of DarkComet samples is limited to those samples uploaded to VirusTotal. While this did not prevent large-scale analysis - indeed, we collected 19,109 unique DarkComet samples during the course of our experiments - it certainly introduces biases. As stated above, certain populations are more likely to upload samples to VirusTotal than others, if at all. It is therefore possible that our sample set is not equally representative of the entire RAT ecosystem, but is instead biased towards more indiscriminate operators attempting to spread infection via public channels (a known behavior [31]), for example.

Targeted Attacks

RATs are often the tool of choice in attacks targeting specific individuals. While this class of behavior is very interesting, it is *not* the intended subject of this study. Our methodology is not designed to emulate any particular target, nor would it be feasible to do so at scale. An operator who is expecting a specific target will encounter any number of indicators that we are not this target, from IP address to system language to username. As such, our study is biased against operators conducting targeted attacks; however, it is conceivable that we receive such samples from VirusTotal, and it would be hard to exclude them from our study before execution. Sections 4.4.11 and 4.4.12 refer to this bias, and how it affects our experimental results.

Infection Longevity

Another class of behavior which our study does not capture is that of "return visits," operators that maintain control of an infected machine over numerous interactive sessions. Anecdotally, voyeurs and sextortionists will monitor their victims for extended periods of time,

even trading or selling long-standing infections to other members of the community [31]. Our methodology is not designed to offer RAT operators extended control of a machine; rather, we observe the initial behavior of an operator gaining access to a newly-infected machine. As RAT infections are hardly stable, we expect operators' initial behavior to be deliberate and revelatory of motivation, but acknowledge that our results are biased against more patient operators.

RDP Inspection

As we will see, 83% of connected operators use DarkComet's remote desktop functionality (RDP) to control the victim machine. Our methodology, manually inspecting screenshots of RDP sessions to build complete behavioral profiles of operators, is not a scalable solution.

Victim User Data Feeds

Much of the observed operator behavior involved attempted interaction with the victim user. 61% of operators attempted to access the victims's webcam, and 26% the microphone. Further, 8% of operators attempted to chat with the victim via DarkComet's chat client. Our honeypot provides neither a webcam nor an audio feed, nor do we simulate a user that can respond to the operators' chat attempts. This is a major limitation which we will discuss in our lessons learned (Section 4.5.3).

Narrow Scope

Our network decoder does not process non-DarkComet actions. Were a DarkComet operator to upload and execute a tool to use through remote desktop, for instance, we would not be able to decode operator interaction with said tool. As shown in Table 4.5, 23 operators did just this. Further, an operator could upload a new configuration (e.g. a new password), or even another type of RAT. Any interaction with this new malware would be un-decipherable. During our trials, operators did, in fact, upload 35 unique RAT executables (see Section 4.4.5), 13 of which are new DarkComet configurations.

Sandbox Evasion

RATs (and operators) can also detect and abort execution in virtualized environments. More specifically, there have been efforts to detect if a binary is running inside a Cuckoo sandbox [116]. Cuckoo implements some countermeasures to these techniques [22], but they are not comprehensive.

4.8 Conclusion

In this chapter, we presented the results of our study focusing on understanding the actions of DarkComet operators. We developed a technique to scan for DarkComet operators active on the Internet. We combine these scans with a collection of DarkComet instances found in the wild that we ran in a realistic environment simulating a real victim machine. From this, we were able to determine how attackers interact with a victim, including time spent on the machine, user data collected, and so on. We found that the most common uses of DarkComet are as a means of accessing a *human* victim for surveillance, harassment, or extortion; stealing user data and account credentials; and abusing the victim machine's vantage point to deploy hacking tools and other malware, probe lateral machines, and launch attacks.

We find that honeypots are a promising tool to monitor the manual actions of DarkComet operators, which enables us to understand the motivations and techniques of these operators. In addition, we demonstrate that honeypot environments show promise as potential tarpit defenses. It is our hope that this initial exploration of the manual attacker ecosystem will spur further investigation.

4.9 Acknowledgements

Chapter 4, in part, is a reprint of the material as it appears in Proceedings of the 38th IEEE Symposium on Security and Privacy 2017. Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blond, Damon McCoy, Kirill Levchenko, 2017.

The dissertation author was the primary investigator and author of this paper.

Chapter 5

Measuring Residual Victims of RAT Campaigns

In our first experiment measuring the RAT ecosystem (described in the previous Chapter 4), we focused on understanding operator motivations, but did not collect any information regarding how many victims were actually impacted by these operators. In the RAT ecosystem, operators are naturally more accessible than victims: controllers must expose themselves to the Internet while awaiting victim connections, and stubs must encode controller IP addresses or domain names. Conversely, victims are challenging to identify; unlike controllers, Internet-wide scanners cannot detect them, and unlike the victims of large-scale malware, they do not participate in typical, observable behaviors like sending spam or creating denial-of-service traffic.

However, from our first experiment we learned that RAT operators favor dynamic DNS (DDNS) as a method for maintaining connectivity with their victims. Subdomains registered with many dynamic DNS providers expire rapidly, particularly if using a free tier of service. This offered us an opportunity to measure a fraction of the victims of RATs. By modifying HAMELIN to track subdomains associated with RAT activity and continuously attempt to claim them ourselves, we were able to squat on subdomains abandoned by RAT campaigns, and even *poach* active subdomains from operators as they expired. Both methods allowed us to measure the residual victims of poorly managed or expired RAT campaigns, as we directed all traffic to a sinkhole designed by my collaborator, Mohammad Rezaeirad, for noise reduction and

measurement.

5.1 Introduction

The unique dangers posed by RAT malware have received considerable attention in the security community and in popular media. Nevertheless, the *relationship* between RAT operators and their victims remains poorly understood. In this chapter, we bring to light the behavior of operators and victims of two popular RAT families, njRAT and DarkComet. Our primary aim is to determine *who* is attacking *whom*, the size of the victim and attacker populations, and how long victims remain vulnerable after a campaign ends.

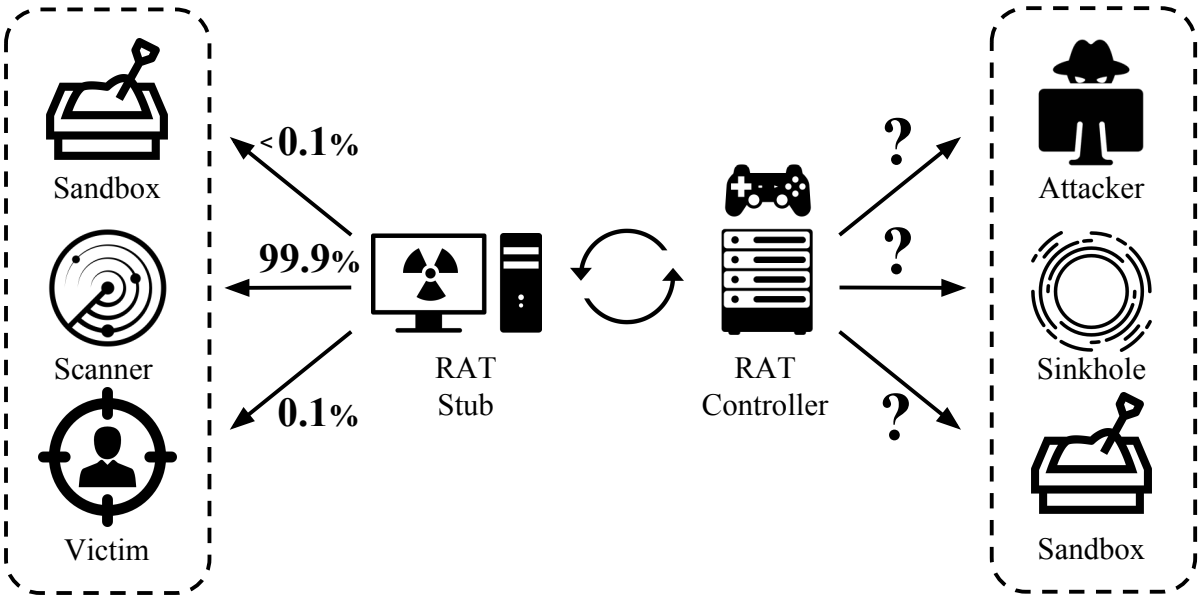


Figure 5.1. *Intelligence pollution* obfuscates the stakeholders in the RAT ecosystem.

Many operators of commodity-grade RATs use free dynamic DNS (DDNS) services for their command-and-control hostname. While DDNS services give operators a free third-level domain name that they can associate with any IP address at any time, DDNS domains are not owned by the operator, and many DDNS services expire and return the third-level name to the pool of available names after 30 days. We exploit this fact to carry out a study of victims, modifying HAMELIN to claim the names of command-and-control servers as soon as they expire

and direct traffic to a server under our control. This technique, called *sinkholing*, has been widely used to capture registrar-registered command-and-control domain names. Our work is the first to apply this technique to short-lived RAT command-and-control domain names.

One of the most significant challenges in studying attackers and victims in a global malware ecosystem is the difficulty of accurately determining their population in the presence of other "active participants," researchers and security professionals impersonating actors in said ecosystem for measurement or defensive purposes. Hosts behaving as victims may in fact be security researchers scanning for command-and-control servers [37]; in fact, HAMELIN's RAT-SCAN falls into this category. Further, apparent command-and-control servers may actually be security vendors or vigilantes operating sinkholes [33, 113, 130]. Thus, a major challenge of conducting a study of the RAT ecosystem is determining who is really a victim or operator, and who is pretending to be one.

The task of identifying victims at scale is made difficult by the volume of traffic sinkholes receive from high-fidelity scanners and sandboxes. The increasing fidelity in RAT scanners that emulate a real victim's behavior and sinkholes that emulate a real RAT controller's protocol has created an arms-race between completing threat intelligence operations, leading to inaccurate measurements and wasted notification efforts. We call this phenomenon *intelligence pollution*, illustrated in Figure 5.1.

My collaborator, Mohammad Rezaeirad, designed and implemented a high-fidelity sinkhole called RAT-HOLE to specifically address these issues. Upon claiming a RAT-associated DDNS domain name, we direct its traffic to RAT-HOLE, which implements the handshake protocol and error triggering tests for two common RATs, DarkComet and njRAT. Based on extensive empirical testing, he designed a set of heuristics to accurately differentiate sandboxes, scanners, and victims, and found that only 6,710 (0.8%) of the over 800,000 IP addresses that connected to RAT-HOLE were likely victims. Further, only 3,231 (69%) of the unique hosts that completed a full RAT handshake with RAT-HOLE were likely victims.

Based on the filtered, high-confidence data set assembled by my collaborator's sinkhole,

in tandem with the controller detection results gathered by RAT-SCAN, I was able to analyze the residual victim populations of abandoned or expired DarkComet and njRAT campaigns. In summary, in this chapter, I will:

- Describe the results of an effort to poach DDNS domain names from active and defunct RAT campaigns.
- Briefly describe my collaborator's methods for classifying RAT sandboxes, scanners, and likely victims based on connection to a sinkhole, which found that only 6,710 (0.8%) of the over 800K IP addresses that connect to the sinkhole were likely victims.
- Analyze the victim populations of the campaigns under observation, and describe the relationship between these victims and their attackers.
- Identify potentially improved interventions that can mitigate the threat of RATs.

The rest of the chapter is organized as follows. Section 5.2 provides necessary background information. Section 5.3 describes our data collection methodology. Section 5.4 presents our results. Section 5.5 discusses the real-world implications of our findings. Section 5.6 considers the ethics of our methodology, while Section 5.7 describes limited scope of the work. Section 5.8 concludes the chapter.

5.2 Background & Related Work

To our knowledge, ours is the first study involving the sinkholing of dynamic DNS domain names actively used in RAT campaigns. Here we provide the necessary background for the rest of the chapter, as well as discuss related work.

5.2.1 Sinkholing

Sinkholing indicates the redirection of the network traffic of infected machines from its intended destinations (e.g., attackers' command-and-control servers [103]), ordinarily to a server designated the *sinkhole*. (Sinkholing can also entail the complete dropping of said traffic.) Local

sinkholing efforts (e.g., those implemented by organizations or individual ISPs) often involve reconfiguring DNS servers and routers to block communication with malicious domains or IP addresses. Larger, coordinated sinkhole operations (e.g., those part of broader takedown efforts) often require cooperation between domain registrars and international authorities. [81, 130]

5.2.2 Dynamic DNS

In Chapter 4, we found that RAT operators often utilize Dynamic DNS (DDNS) services, which allow their controllers to migrate between IP addresses without disruption. DDNS is a service that constantly maps a domain name to a host machine's IP address, persisting across changes in that machine's IP address. Commercial and free DDNS services provide update clients and web portals which their customers can use to sync their domain name with their IP address; some services also provide a base set of domain names from which their customers can create and use subdomains rather than provide their own domains. Many providers of DDNS offer free tiers of service.

Abuse of dynamic DNS services, particularly their free tiers, by malware operators is commonplace and documented [146, 125]. In fact, uring our study, one of the DDNS providers whose domains we were monitoring, DtDNS [34], shut down due to continued abuse. Their closing email, showing in Figure 5.2, implicated the continued abuses by "a few bad actors" in their cessation of operations. Another provider, No-IP [99], actually reached out to us for assistance in confirming the abuse of several of their domain names.

Services like No-IP [99] that offer free tiers of DDNS hostname registrations will typically expire said registrations after 30 days. As we will show, operators often allow their hostnames to expire, providing a large pool of RAT domain names that can be claimed and sinkholed.

5.3 Data Collection & Processing

Figure 5.4 shows a timeline of this experiment. Recall that binary collection, controller domain resolution, and controller active probing are all base components of HAMELIN and

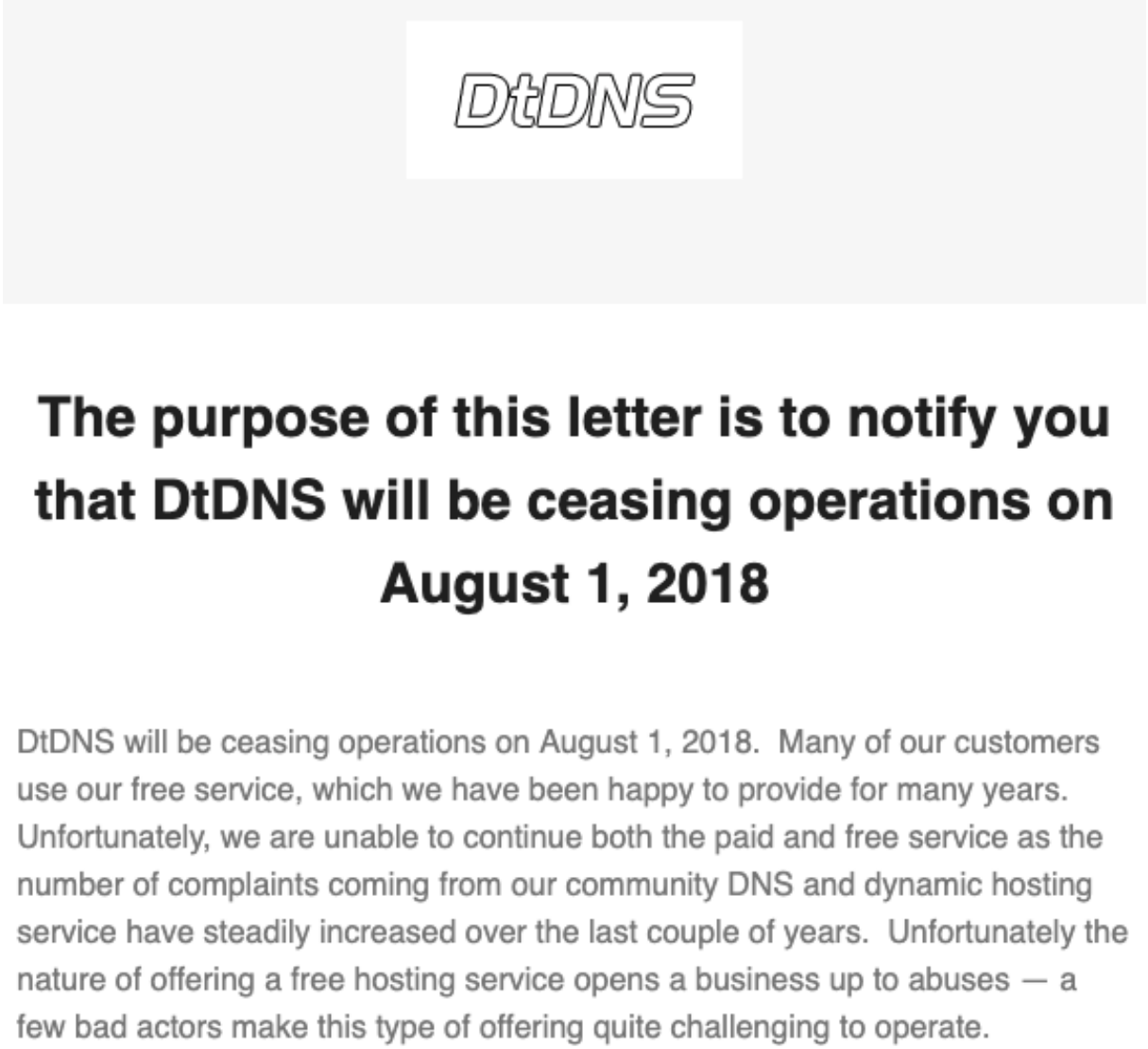


Figure 5.2. The partial contents of an email from the now-defunct DtDNS dynamic DNS service provider (formerly dtdns.com) announcing cessation of service due to continued abuse of their service by customers.

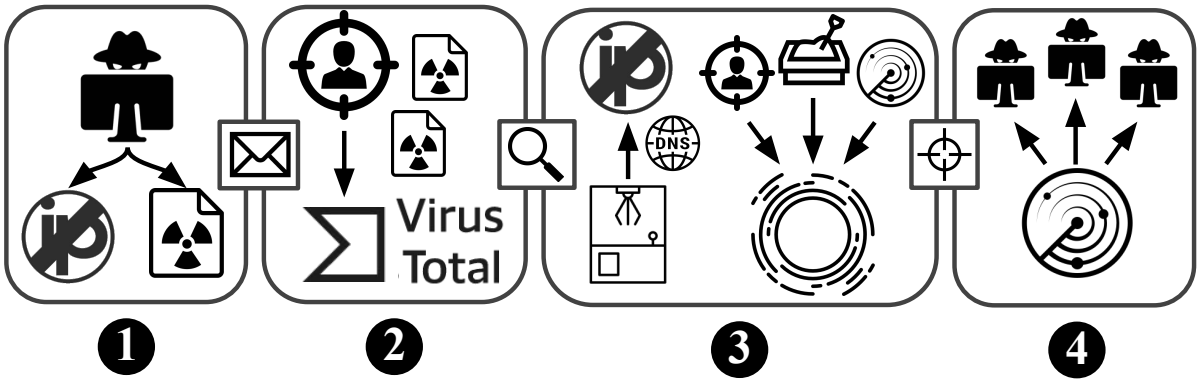


Figure 5.3. The major components of our operation and their interactions with the subjects of our study.

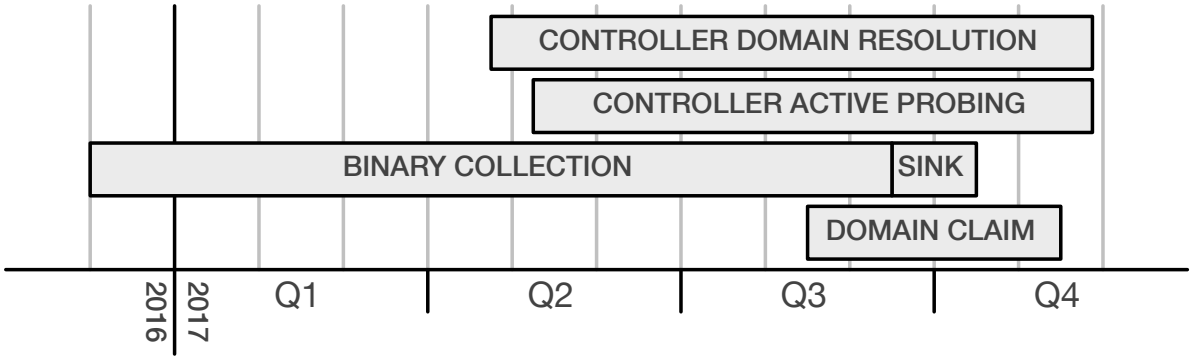


Figure 5.4. Timeline of data collection phases of our study: Binary Acquisition, Controller Domain Resolution, Scanning for Controllers, Domain Claiming, and Sinkholing.

were discussed in Section 3.5.4; here, we only discuss the DDNS domain claiming and sinkhole components created explicitly for this experiment.

Figure 5.3 depicts our system's chronological operation. ❶ An attacker, having registered a domain name with a DDNS provider like No-IP, produces RAT stubs configured with this domain name and spreads them to victims. ❷ Some of these stubs are detected and submitted to VirusTotal. ❸ The stubs trigger HAMELIN's VirusTotal-deployed `YARA` signatures and we download them, extracting their configurations (including domain names and passwords). We determine that some of their domain names belong to No-IP. HAMELIN's DDNS CLAIMER module (described below in Section 5.3.1) registers any domain names it finds available, directing their traffic to RAT-HOLE (described in Section 5.3.2). ❹ Simultaneously, RAT-SCAN probes

all associated controllers for activity.

5.3.1 DDNS CLAIMER

Recall Table 3.3, which showed that most domains we extracted from DarkComet and njRAT stubs were associated with DDNS providers. In particular, No-IP accounted for 60% of all discovered domains, and 77% of all DDNS domains. We developed a web automation toolkit based on Selenium to automate the process of claiming expired DarkComet and njRAT domains owned by No-IP, which we call the DDNS CLAIMER. We did not extend our claimer to other DDNS services due to the non-trivial engineering cost required to support each distinct service; further, as No-IP was the most popular free DDNS provider used by DarkComet and njRAT by far, this concession was justifiable. We only claim expired, available domains; we do *not* attempt to seize owned domains, in order to avoid additional disruption to the ecosystem that we were measuring. We plan to support additional DDNS providers, as well as registered domains, as future work to understand if this affects the results of our analysis.

During the 31-day window between 2017-08-15 and 2017-09-16, DDNS CLAIMER sinkholed 6,897 No-IP domains. 4,493 of these domains came from DarkComet samples, 2,381 from njRAT samples, and 23 were found in samples of both families.

5.3.2 RAT-HOLE Operation Summary

Our sinkhole, RAT-HOLE, serves two functions. First, it imitates both DarkComet and njRAT controller software, recording any victim connection received as a result of the traffic generated by the DDNS CLAIMER. Secondly, it encompasses an extensive suite of tests by which an individual connection from a real victim can be distinguished from that of a scanner or a sandbox executing a stub. RAT-HOLE was designed and implemented by my collaborator, Mohammad Rezaeirad, so I will refrain from describing it in detail here; however, for a thorough description of RAT-HOLE's design, pollution filtering mechanisms, and validation, please refer to our paper [121] or to his dissertation at George Mason University. Instead, I will summarize

the data set it assembled, and continue to my analysis of its results.

Table 5.1. Summary of connections received by RAT-HOLE [121], grouped by peer type, fingerprint (an internal representation of the connection), source IP address, ASN, and country. HF and LF stand for high- and low-fidelity, respectively. †Note that ASN and country show a significant amount of overlap across peer types.

	Connection		Src-IP		Fingerprint (FP)		ASN†		Country†	
Peer Type	**Count**	**Pct.**	**Count**	**Pct.**	**Count**	**Pct.**	**Count**	**Pct.**	**Count**	**Pct.**
Victim	5,320,297	3.5	6,710	0.8	3,231	0.1	1,079	10.1	108	50.0
Sandbox	372,883	0.2	1,181	0.1	877	<0.1	418	3.9	85	39.4
HF Scanner	563,019	0.4	1,349	0.2	589	<0.1	347	3.2	73	33.8
LF Scanner	17,746,010	11.6	1,421	0.2	4,114,064	99.9	390	3.6	80	37.0
Unknown	129,097,791	84.3	815,455	98.5	N/A	N/A	10,418	97.2	216	100.0
Total	153,100,000	100.0	828,137	100.0	4,118,761	100.0	10,722	100.0	216	100.0

RAT-HOLE sinkholed traffic from 6,897 domains for 31 days, for roughly an hour per domain. Overall, RAT-HOLE was in *possession* of domains for 23.1 total days - an average of 17.7 hours per domain, distributed randomly. During this time, it received 153,100,000 TCP connections, the results of which are shown in Table 5.1. In all, less than 1% of all connecting IP addresses were actually RAT victims (6,710 IP addresses likely belonging to 3,231 victims); the remainder were targeted RAT scanners, sandboxed malware executions, and other unidentifiable traffic (e.g., general, Internet-wide scanners).

5.4 Analysis

Here we analyze the 3,231 victims from which we received connections, as well as their presumed attackers.

5.4.1 Victim Analysis

IP Address Churn

During our window of observation, we found that 60% of victims used just one IP address, and 20% of victims used just two. However, we only observed victims for random, hour-long

periods during the 31-day experiment, so it is unlikely that we observe all of a given victims' IP address changes during this period.

Webcam Availability

DarkComet and njRAT victims report if they have a camera device in their handshake, which we record. We found that 1,725 (53.4%) of victims have a RAT-accessible camera, making them susceptible to visual monitoring (unless they have physically covered the camera).

Infected Servers

21 njRAT victims reported a server version of Windows (e.g., Windows Server 2012) running on the peer. We manually investigated the Autonomous System Numbers for the IP addresses used by these peers and confirmed that they were located on corporate networks or cloud hosting providers. This suggests that some higher profile peers associated with companies are infected with njRAT, providing the operator with an entry point into their systems for potential lateral movement.

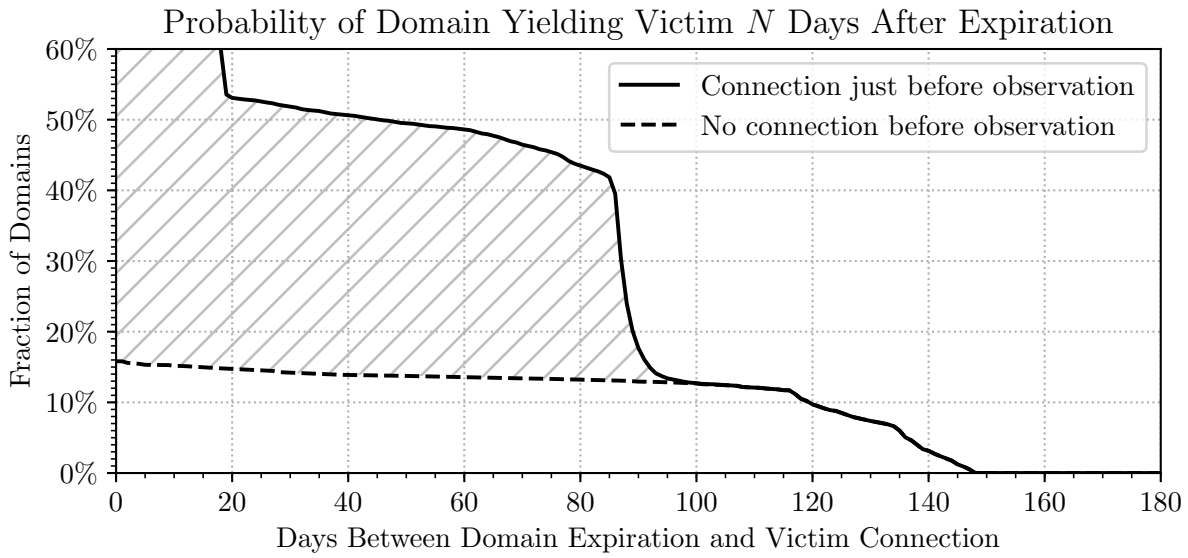


Figure 5.5. PDF showing the probability that a domain we sinkholed would yield a victim connection *N* days after its most recent registration by another party. 1,686 of the 6,897 domains we sinkholed had *no* resolution known to us and were excluded, leaving 5,211 domains (824 of which yielded victim connections).

Infection Longevity

Our methodology is predicated on victims remaining after the command-and-control dynamic DNS domain used by the attacker expires, which occurs 30 days after registration with No-IP. Figure 5.5 shows the fraction of domains still receiving legitimate victim connections as a function of time since the dynamic DNS domain expired. Because our sinkholing period does not span our full domain monitoring period (31 days from 2017-08-15 to 2017-09-16, and 220 days from 2017-04-21 to 2017-11-26; see Figure 5.4), we do not necessarily known victim availability immediately after domain expiration. Figure 5.5 shows an upper and lower bound curve; the upper bound corresponds to the case that at least one victim connection occurred during the period when the command-and-control domain was not monitored, and the lower bound corresponds to the case that no victim connections occurred during the same period. Thus, 120 days after the command-and-control domain expired, 10% of domains were still receiving connections from legitimate victims.

In all, 975 domains received victims, 14% of the 6,897 we sinkholed. 1,686 of these domains had no known historic resolution from any of our sources, including threat intelligence feeds and our own resolver.

5.4.2 Attacker Campaign Analysis

Only 975 of the domains we sinkholed yielded victim connections, yet they received connections from 3,231 unique victims. In Figure 5.6, we examine the number of unique victims any one domain received. 43% of domains received only a single victim; 90% received at most 20 unique victims; 95%, 41 or less. Three outlier domains received over 100 victims. This disparity suggests that some attackers are distributing their malware more widely, or are more proficient at compromising their targets, than others.

We also find that 45% of victims connected to just one domain. 90% of victims connected to four or less different domains, while 95% connected to five or less. These victims connecting

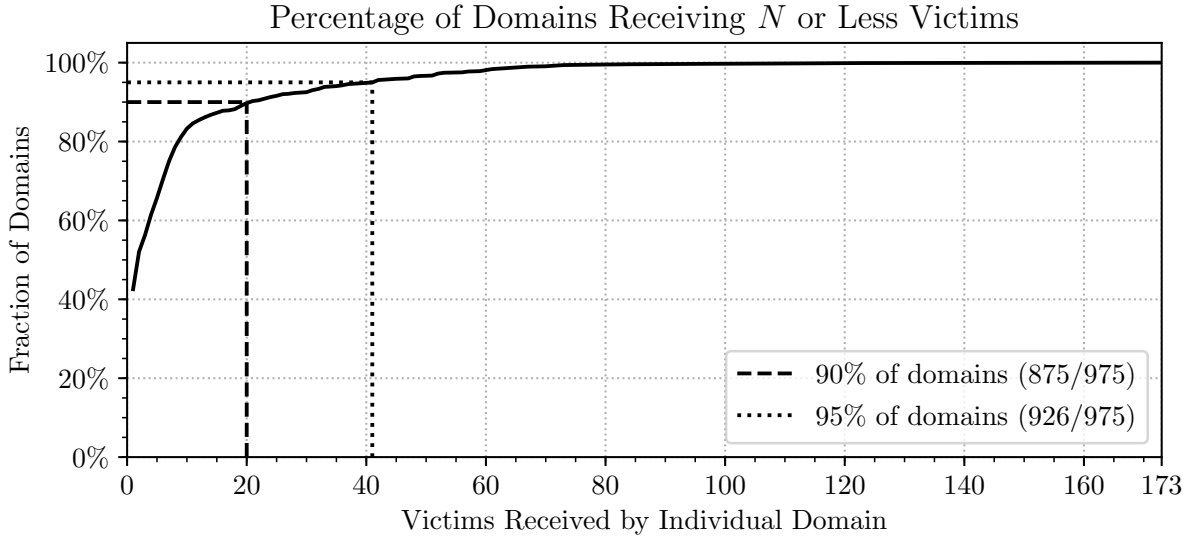


Figure 5.6. CDF showing the number of victims (by fingerprint) received by a given domain. This plot only includes the 975 domains which yielded victim connections.

to multiple different domains, and domains receiving tens of unique victim connections, suggest a number of phenomena. Attackers may be using sets of domains interchangeably, or victims may be infected by multiple different attackers.

To investigate the former, we examine the samples we downloaded from VirusTotal. Our 975 domains are found in 1,429 unique samples. Once again, there is bidirectional overlap. Only eight samples contain more than one domain in their configurations; 1,421 have a single domain. Multiple domains being in a single sample is our strongest indicator of them being shared by an attacker. Oppositely, 246 domains are found in more than one sample's configurations; the remaining 729 domains are each present in just one sample. Some domains are shared by many samples - one being found in 24 unique DarkComet samples. Unfortunately, these domains further muddy our analysis. In the case of the domain shared by 24 samples, only two samples clearly belong to the same actor (based on shared configuration parameters).

Our methodology cannot definitively answer whether attackers use rotating domains, or whether victims are multiply infected by different campaigns. Based on our evidence, both appear probable, and confound our attempts at differentiating attackers and their victims.

5.5 Discussion

5.5.1 Protecting Victims

Our results show that expired RAT domains still have likely victims attempting to connect to them. The 3,231 likely victims we detected could be further victimized by an adversary that claimed these domains. We are in the process of working with some free DDNS providers, including No-IP, to understand if they would be willing to permanently block domain names associated with RAT controllers. We plan to pursue the same dialogue with the VPN proxy services that provide support infrastructure for RAT operators.

5.6 Ethical Considerations

Our methodology was approved by our institution's Institutional Review Boards (IRB) and general legal counsel. The ethical framework that we operated under is that RAT-HOLE only completed the protocol handshake with peers that contacted us. We did not attempt to execute commands on infected peers. During the handshake there is some potentially Personal Identifiable Information (PII) that the peer sends to us, such as the PC name (often the name of the victim) or full website URLs a person is visiting if the active window is a browser. To mitigate the potential harm caused by our study, we immediately encrypted any fields that might contain PII and did not ever store an unencrypted version of these fields (PII listed at Table 2.2). Our IRB takes the position that IP addresses are not personally identifiable. In no cases did we attempt to tie our measurements to an actual identity.

During the experiment, our sinkhole infrastructure was repeatedly mistaken for and flagged by security vendors as a large-scale RAT controller hosting operation. It is likely that the IP range on which we ran our measurement sinkhole is permanently tainted. Such a possibility must be considered before impersonating network-accessible devices in the malware ecosystem, even if just for measurement purposes.

5.7 Limitations

5.7.1 RAT Family Scope

We limited our study to two RAT families, DarkComet and njRAT, since reverse engineering and implementing parts of their protocols in both RAT-HOLE and RAT-SCAN is laborious. DarkComet and njRAT were chosen for their popularity and the availability of prior work to assist in the reverse engineering process [28, 15, 46]. As an approximate measure of prevalence, we counted the number of unique binaries associated with major RAT families on VirusTotal; njRAT and DarkComet were the third and fifth most prevalent RATs when our study began. We plan to investigate methods for automating protocol reverse-engineering [18, 23, 126] as future work to make our system more extensible.

5.8 Conclusion

We presented the results of our study of attacker and victim populations of two major RAT families, njRAT and DarkComet. We modified HAMELIN to poach dynamic DNS domain names associated with RAT activity upon expiration, directing their traffic to a sinkhole designed by my collaborator. Using the data collected by this sinkhole, we then report on the population of victims and controllers, their geographic relationship, and periods of activity. Our results show that the RATs we studied are used primarily by operators and victims located in the same country, with the bulk of the population in Russia, Brazil, and Turkey. We also found that victims remain vulnerable long after the controller abandons the campaign, presenting an opportunity for third-party intervention by sinkholing the domains.

5.9 Acknowledgements

Chapter 5, in part, is a reprint of the material as it appears in Proceedings of the 27th USENIX Security Symposium 2018. Mohammad Rezaeirad, Brown Farinholt, Hitesh Dharm-

dasani, Paul Pearce, Kirill Levchenko, Damon McCoy, 2018. The dissertation author was the primary investigator and author of the components of the paper presented in this dissertation.

Chapter 6

Quantifying Victim Harm in the RAT Ecosystem

The second experiment we ran using HAMELIN, described previously in Chapter 5, was designed to measure the victim populations of RAT campaigns, a challenging endeavor. Unfortunately, the study was limited by the vantage point we had chosen; we could only claim and sinkhole domains that had already expired or been abandoned by RAT operators. As such, the study's results provided an incomplete view of the RAT ecosystem, at least concerning its victims. Following this study, however, we discovered prior work indicating that DarkComet's command-and-control protocol was vulnerable to blind file retrieval requests. This, combined with every DarkComet controller's maintaining an SQLite database of all its victims, offered us a new vantage point for victim measurement. Following much legal vetting, we modified HAMELIN to download this victim database from any DarkComet controller with which it made contact, finally allowing for a thorough investigation of the elusive victims of DarkComet malware and the harms they suffer.

6.1 Introduction

Traditional forms of malware generate revenue for a miscreant through large-scale illicit activity, be it spamming, click fraud, or ransom extortion. The direct victims of such malware experience the infection as a theft of CPU cycles, network bandwidth, or money. While costly in

the aggregate, each user's loss is ultimately limited by an attacker's ability to extract value from such victims at scale. Remote Access Trojans (RATs) change this arrangement to one where an attacker interacts with each victim *individually*, scouring through the victim's file system, spying on the victim through the webcam and microphone, or harassing the victim using the computer's speakers and user interface.

In contrast to traditional malware, whose operators have made millions of dollars through illicit activity [83], the financial gains of RAT operators are necessarily limited by the small number of victims they can control. From the victim's point of view, however, a RAT infection may incur not only financial loss but also significant emotional distress of blackmail and sextortion perpetrated by RAT operators [19, 31]. Thus, the apparent amateur nature of RAT operators and the negligible economic losses they cause belie the greater *individual* harm they beget. Unfortunately, aside from a handful of high-profile cases, little is known about the victims, as published studies of RATs have largely focused on the attackers, their behavior, practices, and business models [85].

As opposed to most studies of malware, the focus of this study are the *victims* of RATs. A considerable challenge of studying RAT victims is our limited visibility into this population. Victims of RATs are difficult to identify: computers infected with RATs do not, as a rule, commit click fraud, send spam, participate in DDoS attacks, or otherwise stand out to an external observer. Thus, unlike botnets, even measuring the population of such victims poses a special challenge.

In this study, we have created a framework for analyzing data collected from RAT operators that enables us to study the harms victims of RAT malware experience. It is commonplace for RAT controller software to maintain a database of each victim infected, along with data pertaining to that victim (e.g., logs of captured keystrokes). By treating the victim entries in these databases as a form of ancestry, we have developed techniques for tracking RAT controllers across hostname changes and for understanding their phylogeny with regards to the origin of their controller software. Further, we have developed techniques using anonymized victim metadata from these databases to remove spurious victim records resulting from network scans and the

execution of DarkComet stubs by security analysts. This allows us to determine, with high confidence, which records correspond to real victims.

A unique feature of one popular RAT called DarkComet provided us with an opportunity to collect these victim databases at scale. Whether added intentionally or by mistake, DarkComet makes it possible to download its victim database by issuing a specific command to the controller software over the command-and-control channel. We used this mechanism to download 3,988 victim databases from 590 distinct controllers, which we discovered while monitoring a set of 64,343 domains from samples from MalwareConfig, Shodan, VirusTotal, and ReversingLabs.

Using the techniques we developed to track controllers and filter spurious victim records, and following a strictly-controlled methodology of anonymizing private data about victims, we report on a population of 44,802 victims infected by DarkComet controllers over a span of five years. While this study is not comprehensive due to the limitations of our data collection techniques, the sample set we observed allows us to understand the victims of the DarkComet ecosystem and the harms they suffer. We found that the DarkComet operators we observed acquire **50 new victims per day**, or one every 29 minutes. In total, they have captured **186,748,463 keystrokes** over **16,397 hours** of victim surveillance, including emails, chat transcripts, and login credentials. Over **6,000 victim webcams** were accessed.

In summary, the major contributions of this work are:

- We describe a methodology for tracking controllers of DarkComet, a popular commodity RAT, across hostname changes based on a phylogenetic analysis of their victims.
- We describe a methodology for identifying real DarkComet victims in the presence of honeypots, scanners, and VM execution of malware by researchers.
- We detail the process by which we collect information about victims of DarkComet at scale and present the results of our analysis of the victims in the studied ecosystem, the harms they incur, and their relationship with their attackers.

The rest of this chapter is organized as follows. Section 6.2 provides necessary back-

ground. Section 6.3 describes our data collection methodology; importantly, Section 6.7 discusses our **ethical and legal considerations**. Section 6.4 describes how we processed the collected data. Section 6.5 presents our results. Section 6.6 discusses our findings. Section 6.9 concludes the chapter.

6.2 Background & Related Word

This work aims to report on the victims of DarkComet, a well-known RAT. In this section, we provide the necessary background for the rest of our study.

6.2.1 Downloading Victim Databases

DarkComet allows an operator to configure a stub to automatically download a file from the controller, for example, to download updates or secondary payloads from the controller. Denbow and Hertz [28] reverse-engineered the DarkComet network protocol and discovered that DarkComet allowed a stub connected to a controller to request and download *any* file from the controller, without operator notification. We explain how this mechanism in detail in Section 6.3.1. For this study, we use this feature of the DarkComet protocol to glean information about the victims of DarkComet operations.

DarkComet stores information about every victim ever infected in an SQLite database file. Researchers have previously investigated the possibility of obtaining this database from controllers; in particular, Kevin Breen [15, 16, 12] proposed using DarkComet's arbitrary file download functionality to collect DarkComet controller databases for research purposes. Breen's `dc-toolkit` [11] provides a set of working Python scripts for downloading DarkComet databases, which was later incorporated into Metasploit as a module [17, 59]. Breen [12] also examines the contents of a sample database he downloaded with the `dc-toolkit`, highlighting some of its sensitive contents, such as the keylog table.

6.2.2 Hack Pack Sharing

DarkComet was initially offered freely for download by its author, `DarkCoderSc`, from an official site [79]. However, its authors removed DarkComet from the official site following its widely publicized use by the Syrian government in a cyber-espionage campaign against dissidents at the onset of the Syrian Civil War [128, 92, 48]. The official site reads now states, "DarkComet-RAT development ceased indefinitely in July 2012. Since the [sic], we do not offer downloads, copies or support."

Despite this, DarkComet is available for download, packaged as in what is known as *hacking packs* or *hack packs*, collections of RATs and other malware that are sold or freely distributed in hacking forums online. Many RAT hack packs are bundled and distributed by RAT operators hoping to improve their reputation in a hacking forum. These operators package the very RAT software they use personally for distribution. RAT controller executables, including `DarkComet.exe`, run from a directory that contains its supporting DLLs (e.g., `SQLite.dll`) hack pack distributors simply compress and ship this entire directory. The same directory also contains the victim SQLite database, stored in a file called `comet.db`. Most hack packs *also* include this database file, which contains records of victims infected by the hack pack creator. We exploit this phenomenon in Section 6.4.1 to understand the ancestry of victim databases we obtain from live controllers.

6.2.3 Protocol Vulnerabilities in RATs & Other Malware Controllers

Denbow and Hertz [28] reverse engineered the network protocols for several popular, freely available RATs, including DarkComet, Bandook, CyberGate, and Xtreme RAT. In this analysis, they first discovered that DarkComet's controller allows arbitrary file reads. Grange [52] expanded on this, discovering and describing corollary file download functionality in XtremeRAT and Gh0st RAT, as well as other network protocol vulnerabilities in PlugX RAT. Grange also posited that blind file retrieval API implementations have propagated through other RAT families

via the leak of XtremeRAT's source code, which RATs like Spynet, CyberGate, and Cerberus copied nearly wholesale.

Recently, Miller [94] showed that modifying authenticated, outbound traffic to PlugX RAT controllers can be used defensively to interfere with RAT operations. Similarly, Watkins *et al.* [137, 138] performed successful penetration testing on banking and denial-of-service malware command-and-control server network protocols, proposing the usage of discovered vulnerabilities for measurement and defensive operations.

6.2.4 Estimating Infected Population

The accuracy of malware infection size measurements and estimation has long been an issue broached by botnet-focused measurement studies. Ramachandran *et al.* [115] proposed a method for estimating botnet infection size based on the frequency of DNS lookups to C&C domains. A subsequent pair of botnet studies used DNS lookups [25] and IRC channel monitoring [1] as measurement vectors but arrived at different estimates due to churn [114]. Recently, Antonakakis *et al.* [7] used a variety of techniques to gauge the size and scope of the Mirai botnet, including active scanning and running a so-called *milker* to obtain attack commands.

6.2.5 Data Set Pollution

Part of our data processing methodology entails pre-processing our data to remove records introduced by interfering measurement and counter-offensive operations. In malware infection size estimation, this is a particularly significant obstacle due to the prevalence of such operations by security researchers and anti-malware vendors alike. A number of botnet measurement studies have expounded on the issue of data set pollution [53, 108, 38, 130, 96]. Particularly, Kanich *et al.* [65] showed that data set pollution caused by interfering measurement operations and other active participants in the network could magnify the measured size of the Storm botnet by 10 to 20 times when using a naive estimation approach. Another common source of measurement pollution is the sandboxed execution of malware. We attempt to counter

this by submitting our own malware samples to Internet-connected sandbox services to obtain measurement artifacts about said sandboxes, mimicking part of the methodology showcased by Yokoyama *et al.* [143]. Specifically related to RAT malware, Chapter 5 touched on my collaborator Mohammad Rezaeirad's and my efforts to investigate and enumerate the "stakeholders" in the RAT ecosystem via active scanning and sinkholing, tailoring their methodology to identify active participants polluting the RAT ecosystem. We adopt the techniques described in that chapter (and more thoroughly in [121]) to identify and exclude such pollution in our data, as described in Section 6.4.2.

6.3 Data Collection

Figure 6.1 illustrates our methodology for collecting DarkComet victim databases, which we describe in this section. We used HAMELIN's RAT-HUNTER to acquire target DarkComet hosts and RAT-SCAN to probe them for the presence of DarkComet controller software, as described in Section 3.5.5. Upon controller discovery, we attempt to download the DarkComet victim database as described below.

6.3.1 Victim Database Acquisition

The central focus of this study is the DarkComet database. In this section, we describe how we acquire a data set of DarkComet databases, and what information they contain.

File Download Procedure

DarkComet implements functionality by which a controller can instruct a stub to download a file from the controller's machine. Figure 6.2 provides a network diagram depicting this process. As indicated in the diagram, the controller issuing the `QUICKUPC` download command is actually unnecessary. A stub can connect to a controller, authenticate with it, and request *any file* from the controller machine; the controller will oblige, provided the file exists. This is, in fact, likely a feature rather than a bug. DarkComet's plugin system includes modules that

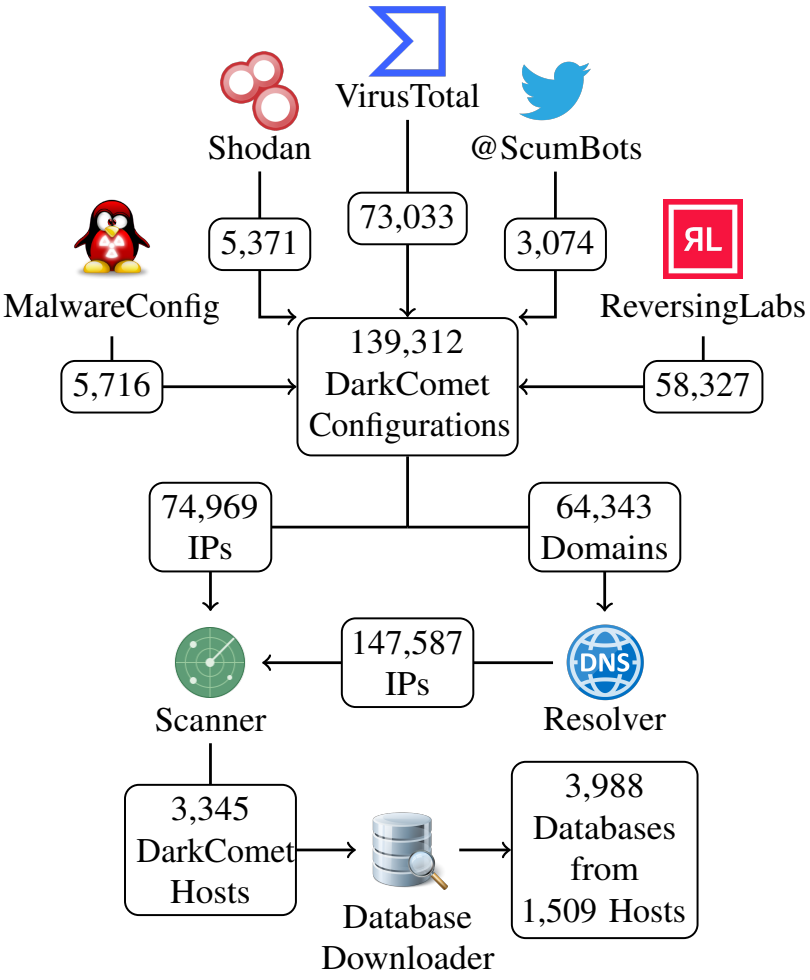


Figure 6.1. Illustration of our data collection methodology, from the sourcing of DarkComet configurations from IoC feeds to the downloading of databases from detected hosts.

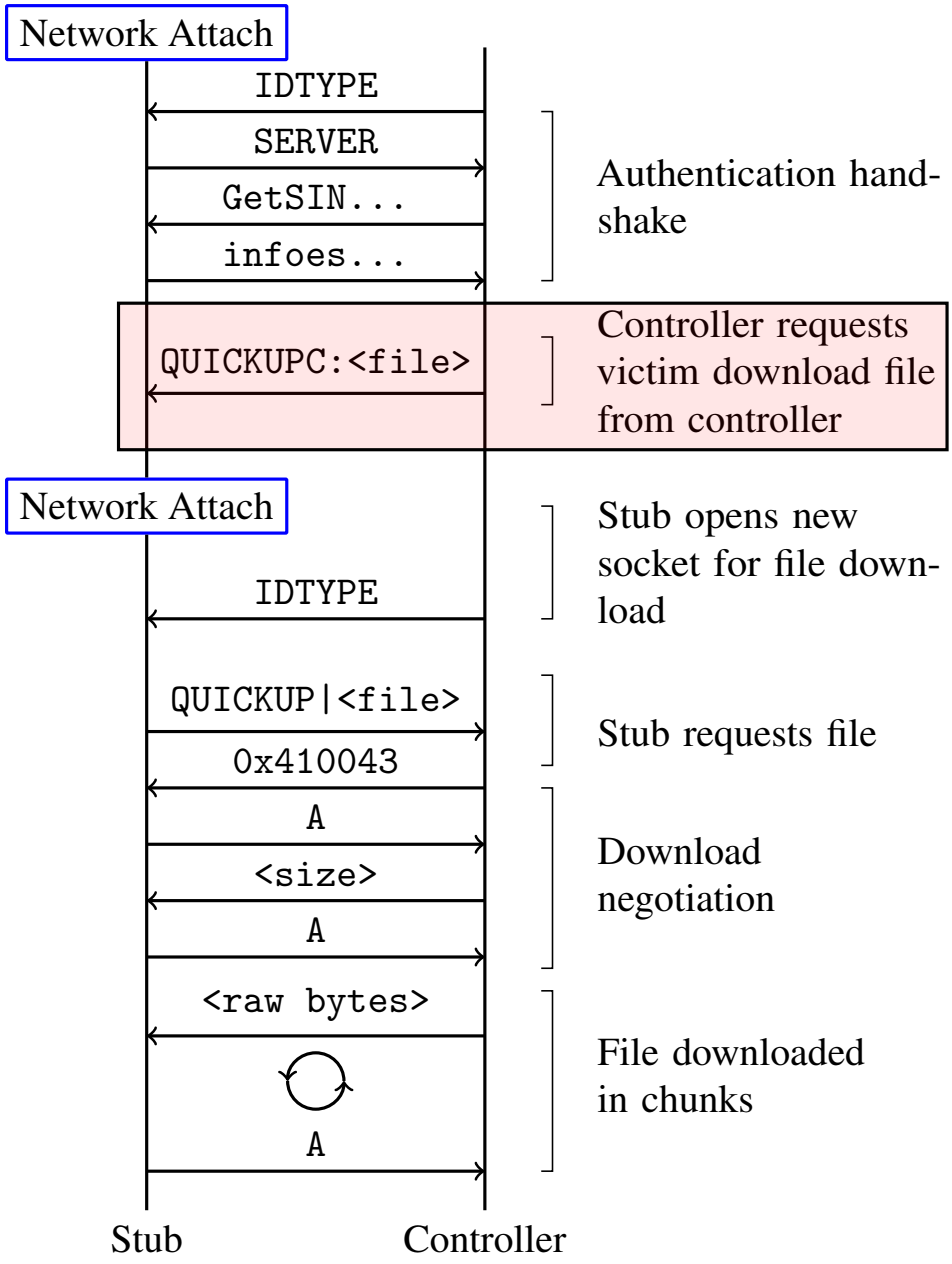


Figure 6.2. Network diagram of the DarkComet file download procedure. The highlighted command is omittable, allowing for arbitrary file downloads. Descriptions of the packets in this diagram are available in prior work [28, 11].

allow an attacker to script stubs to automatically download (and install or run) files from the controller upon connection, without requiring explicit intervention from the attacker. However, this "feature" also allows for blind file retrieval, wherein any stub can request arbitrary files from the controller. This has been well-documented, as we described in Section 6.2.1. We use DarkComet's blind file retrieval functionality to collect *victim databases* and *DarkComet configuration files* from DarkComet controllers.

Upon execution, `DarkComet.exe` creates or loads a file in its working directory named `comet.db`. This SQLite database manages victim connections, and is described thoroughly in the following Section 6.3.1. We collect this file by emulating a DarkComet victim and simply issuing the command `QUICKUP|comet.db`, as it is located in `DarkComet.exe`'s working directory. From December 5, 2018 to April 4, 2019, we downloaded 3,988 databases from 1,509 unique IP addresses. DarkComet also uses an `INI` file named `config.ini`to manage configuration information internally. This file encodes additional, valuable information about operator interactions with victims; as such, we updated our methodology on March 25, 2019 to collect this file as well using the command `QUICKUP|config.ini`. From March 25, 2019 to April 4, 2019, we downloaded 388 configuration files from 336 unique IP addresses.

Tainting Databases on Download

Each time we download a DarkComet database as described above, we append a unique, tainted victim record (a *taint*) to its `dc_users` table (continue to Section 6.3.1 for more details on this table). This tainting happens automatically, as per the handshake shown in Figure 6.2, our downloader registers with the controller as a victim each time it downloads a database; we simply taint the victim information we transmit in the `infoes` packet such that we can identify our downloader's records uniquely in the `dc_users` table.

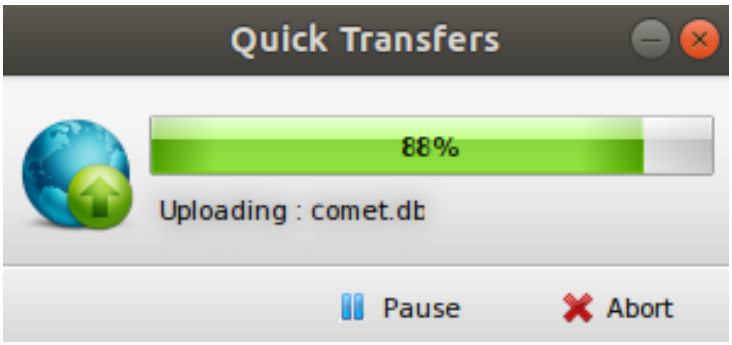


Figure 6.3. Window displayed during file download.

Table 6.1. Database download failure reasons and occurrence.

Failure Condition	Pct.
Download canceled by operator	37.3%
SOCKS5 proxy connection error	32.0%
Connection timed out	26.6%
File not available	4.1%

Download Failures

Within the first month of operation, our downloader was disabled by a series of denial-of-service attacks. Since then, we have used `SOCKS5` proxying to anonymize our download requests. This, as well as some other factors, have impacted our ability to successfully download databases consistently. In the course of the experiment, we attempted 5,484 database downloads, but 1,496 downloads failed. The specific reasons for download failure are listed in Table 6.1 in the Appendix; network connectivity problems were the main cause of failure. We believe this is due to `SOCKS5` proxying for large file downloads. Indeed, some database downloads took hours to complete. The other issue affecting our download capability is that DarkComet allows the operator to cancel downloads while they are occuring. The DarkComet controller displays

Table 6.2. Download metrics.

Metric	Max.	Min.	Avg.	Med.
Time	3 hours	2 sec	251 sec	31 sec
Size	340 MB	7168 B	6 MB	369 KB

Table 6.3. The schemas of the tables of importance in the DarkComet database.

Table	Column	Format	Example
dc_users	UUID	¡HWID¿-¡Unique suffix¿	{846ee340-7039-11de-9d20-806e6f6e6963-12345678}
	userIP	¡WAN IP¿ / [¡LAN IP¿] : ¡Port¿	8.8.8.8 / [10.0.0.5] : 1604
	userName	¡Hostname¿ / ¡Username¿	DESKTOP-432AHT11 / Administrator
	userOS	¡OS¿ [¡Build¿] ¡Arch¿ (¡Drive¿)	Windows 7 Service Pack 1 [7601] 32 bit (C:\\ )
	userGroup	¡dc_groups:groupId¿	0
dc_keyloggers	UUID	¡dc_users:UUID¿	{846ee340-7039-11de-9d20-806e6f6e6963-12345678}
	name	¡Date¿-¡Random integer¿.dc	2015-12-10-5.dc
	content	<hexified victim keystrokes>	..
dc_groups	groupId	¡Sequential integer¿	0
	groupTitle	¡Operator-created title¿	Webcam
	groupSubTitle	¡Operator-created subtitle¿	Webcams are here.
	groupFooter	¡Operator-created footer¿	acc4hak

a pop-up window during a file transfer, shown in Figure 6.3 in the Appendix. This window offers the operator the ability to abort a file transfer while it is in progress, a feature operators sometimes used to prevent us from downloading databases. Overall, we failed to extract a single database from 385 controllers; for another 231 controllers, only some downloads succeeded.

Of the 3,345 DarkComet hosts detected by our scanner, we only attempted to download databases from 1,894; the remaining 1,451 were never probed due to two factors. First, per the legal and ethical framework on which we based on data collection methodology (see Section 6.7), we do not attempt downloads from hosts which have active web or email servers running, an aggressive measure to avoid probing unaware, compromised hosts being used as intermediary infrastructure by DarkComet campaigns. Second, our downloader is network-constrained; there are some short-lived DarkComet hosts from which it never has a chance to download a database.

Victim Database Schema

DarkComet uses a SQLite database, stored in a file named `comet.db`, to manage victim connections and metadata. Table 6.3 depicts the schemas of each of its tables of importance, as well as provides examples of each.

dc_users. This table contains a single row for every unique victim that has connected to the controller. In Table 6.3, we observe the contents of a sample row in the dc_users table. As this table is append-only, the order of its contents indicates the order in which victims first connected;

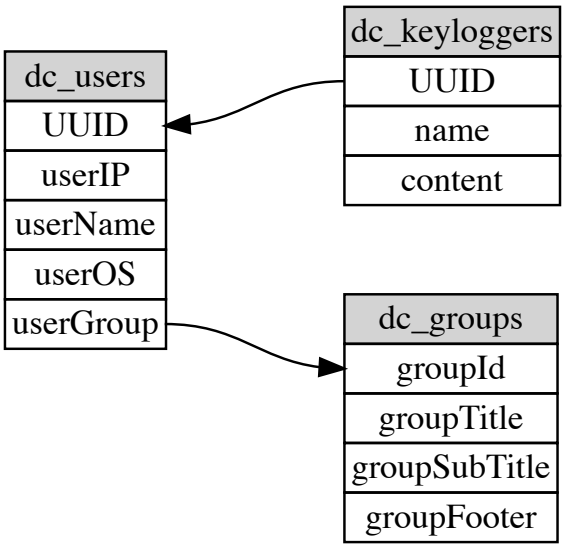


Figure 6.4. The schemas of the tables of importance in the DarkComet database.

users whose IP addresses or operating systems change maintain their original row. Most items in this row are self-explanatory. `userGroup` references the `groupId` field in `dc_groups`. `UUID` is the victim machine's hardward profile ID, returned by the function `GetCurrentHwProfile`, sometimes appended with a random identifier. Since this table is likely to contain victims' personally identifiable information (PII), we hash the `userIP` and `userName` fields before storing them. Prior to hashing victim IP addresses, we resolve their geolocations against a local MaxMind GeoLite2 City database [91].

dc_keyloggers. This table stores victim keystrokes. Each row contains the keystokes logged from a victim, denoted by a `UUID` that references `dc_users`, on a given day. The `name` field, shown in the example below, refers to the file (rolled daily) on the victim machine in which keystrokes are logged. DarkComet caches victim keystrokes until connection to a controller, at which point all stored daily logs are uploaded at once. The `contents` field stores all captured victim keystrokes, delimited by the victim's active window as it changes. As this table is highly likely to contain PII, we do not store it. Instead, we count the number of keystrokes captured, and, as of our methodology update on March 25, 2019, store letter distributions and victim

active window matches against 141 regular expressions for common applications and websites, including the Alexa Top 100. We group these regexes into 29 categories, listed in Table 6.8 in the Appendix.

dc_groups. This table allows for attackers to sort and annotate victims into groups. Each row is an attacker-created group, complete with a title, subtitle, and footer. These groups tend to reveal an attacker's language and motivations.

DarkComet Configuration File Format

DarkComet uses an `INI` file, stored as `config.ini`, to manage configuration information internally. Among other things, this file encodes whether an operator has interacted with a specific victim via the inclusion of a configuration section header with the victim's `UUID` (corresponding to `comet.db`). Further, these victim-specific sections contains keys indicating whether the operator has accessed or recorded the victim's webcam or screen.

`config.ini` also contains automation information. It lists the tasks the operator has configured stubs to execute upon connection, from controller hostname updates to DDoS targets. It includes the operator's No-IP login information and hostname(s), as the DarkComet controller can automatically handle DDNS IP address updates. Table 6.4 in the Appendix details the numerous configurations stored in this file.

Hack Pack Sourcing

To supplement our data set of downloaded DarkComet databases, we downloaded DarkComet hack packs from a combination of hacking forums and VirusTotal. Recipients of hack packs often upload them to malware scanning sites like VirusTotal, as the software in hack packs is (ironically) frequently infected by the packager of said hack packs. From this source, we collected an additional 29 distinct DarkComet victim databases. We use these databases in Section 6.4.1 to describe the phylogeny of DarkComet controller software.

Table 6.4. Sample of information in the DarkComet `INI` file.

Header	Key	Description
`<Victim UUID>`	`CAMFREQ`	Frequency of webcam still capture
	`CAMQUALITY`	Quality of webcam stream
	`CAMREC`	Webcam set to record constantly
	`CAMSTRETCH`	Widescreen webcam
	`SC2OP<N>`	Screen capture options
	`SC2QUALITY`	Screen capture image quality
	`SC2SIZE`	Screen capture image size
`FUNC`	`AUTOCAM`	Record webcam on connect
	`AUTOMIC`	Record microphone on connect
	`AUTOSCREEN`	Capture screen on connect
	`CUSTOMFUNCS`	Execute functions from `SIN:Tasks` on connect
`LSTSIN`	`col<N>`	Victim information column dimensions
`NOIP`	`AUTOUPDATE`	Update DNS record with current IP address
	`HOST`	No-IP DDNS hostname
	`PASS`	No-IP account password
	`USER`	No-IP account username
`NOTIFY`	`BALOON`	Alert window on victim connect
	`SND`	Push notification on victim connect
	`TOAST`	Toast message on victim connect
`PUSHME`	`active`	Push notifications enabled
	`api`	Operator's pushme.to (defunct) URL
	`c<N>`	Custom push configuration options
	`sig`	Notification message to push
`SEARCHUSERS`	`CASEMASK`	Search case-sensitivity
	`SEARCHBOX`	String for which to search victims
	`WHOLELINE`	Search all victim metadata vs. victim name
`SECURITY`	`PASSWD`	RC4 password for DarkComet communications
	`SHOWFTPPWD`	Show FTP password (stored in `comet.db`)
`SIN`	`AUTOREFRESHSIN`	Refresh victim information periodically
	`GEOIPFLAG`	If victim geolocation enabled
	`GROUPS`	If `dc_groups` exists in database
	`Ports`	List of ports to listen for victims
	`REFRESHSINRATIO`	How often to refresh victim information
	`THUMBX`	Thumbnail width
	`THUMBY`	Thumbnail height
	`Tasks`	List of functions to execute on victim connect
	`USERSTHUMB`	Show thumbnails of victim screen captures
	`disclamer`	If operator has agreed to DarkComet EULA
	`help`	If operator has enabled help
`SINPOS`	`height`	Control panel height
	`left`	Left position relative to screen
	`top`	Top position relative to screen
	`width`	Width
`SYSTEM`	`AUTOLISTEN`	Listen for victims on program start
	`CLOSETRAY`	Hide panel from taskbar
	`KEEPALIVE`	Frequency to issue victim keepalives
	`SKIN`	If operator is using custom DarkComet skin
	`SKNFN`	Custom DarkComet skin

6.4 Post-Processing

We downloaded 3,988 DarkComet databases from 1,509 unique IP addresses from December 5, 2018 to April 4, 2019. Whenever possible, we downloaded the database from a given controller multiple times – no more than once every 24 hours – allowing us to observe the acquisition of new victims over the course of the 120-day measurement period. We know that a single IP address is not synonymous with a single controller; indeed, most controllers use one or more domain names for addressing rather than hard-coded IP addresses. So in Section 6.4.1, we describe how we identify databases from the same controller (despite potentially being downloaded from different IP addresses), and how we construct a family tree of database inheritance. We also know that some of the records in a given database may not be real victims. Chapter 5 demonstrated that numerous entities are running DarkComet malware samples in sandboxes or operating high-fidelity DarkComet network scanners. These operations pollute the databases we download with fake victim records. Because we are interested in *real* victims, we take several steps to remove this pollution from our data set. We describe this methodology in Section 6.4.2.

6.4.1 Database Attribution

When we download a database from a controller, we record the hostname used to contact the controller, which may be either a domain name or a raw IP address. This hostname is used to uniquely identify a particular controller over the course of the experiment, allowing us to track controllers identified by domain names across multiple IP addresses. Based on our corpus of DarkComet sample configurations, we know that some controllers use more than one hostname. We consider any domain names and IP addresses that appear in the same DarkComet sample configuration to belong to the same controller.

Using this initial technique of hostname-based consolidation, we condense the 1,509 DarkComet IP addresses from which we downloaded databases to **664** controllers. 421 of these

controllers are identified by domain names, encompassing 84% of the 1,509 IP addresses (1,266) and 63% of the 3,988 downloaded databases (2,576). The remaining 243 IP addresses identify controllers with hard-coded IPs, to which the other 1,412 databases belong. Interestingly, only 15% of our DarkComet samples contain hard-coded IP addresses, compared to the 37% of active controllers identified by them.

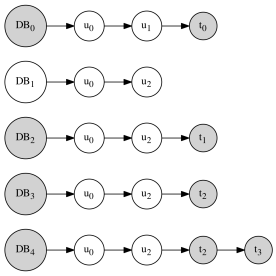
Because a controller may produce stubs with multiple, disjoint configurations, there may be more than one controller hostname for each database in our data set; therefore, 664 is an *overestimate* of the number of unique controllers we observed. To identify cases where the same controller's databases were contacted under different hostname, we use the records in `dc_users` to construct an inheritance tree of DarkComet databases.

DarkComet Database Ancestry

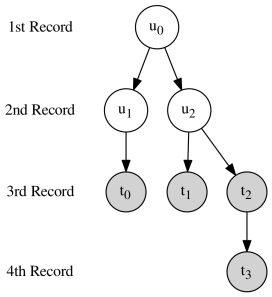
The `dc_users` table in a DarkComet database is append-only, meaning that when a controller infects a new victim, the victim's metadata is appended to the `dc_users` table. Returning users are identified by their `UUIDs`, so duplicate records are never created for the same victim. Thus, the order of the records in `dc_users` describes the order in which the corresponding victims were infected. Each time we download a `dc_users` table from a controller, we expect it to have new victims appended to the end, so that the previously downloaded `dc_users` table is a prefix of the new one.

Furthermore, recall that we add a unique victim record, or *taint*, to the `dc_users` table each time we download it because the process of connecting to the controller generates a victim record. A controller's `dc_users` table should, therefore, not only contain a history of the victims the controller has infected in the order they were added, but also a special victim record corresponding to each time we downloaded the database.

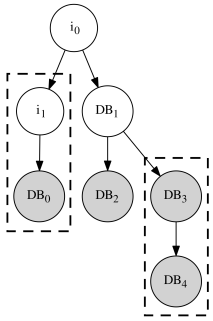
Using the monotonic growth property of the `dc_users` table described above allows us to identify a controller by its database, even if we contact it at a different hostname and IP address. Applying this technique identified 41 controllers using 115 hostnames or hard-coded IP



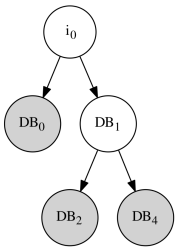
(a) Example Data Set. Suppose we have five databases belonging to five different controllers (as determined by hostname consolidation), the `dc_users` tables of which contain the records shown here. A victim record, denoted u_n, may be present in more than one database. Per our methodology, the last record in every database is a taint record, denoted t_n. DB_1 is a known hack pack database, so it is unshaded; the rest were downloaded by our scanner.



(b) Victim Tree. We generate a inheritance tree of victim records based on their positions in the databases' `dc_users` tables. Each node represents a record that directly follows its parent node's record in at least one `dc_users` table, the children of which follow the record in some `dc_users` table. The rank of a node indicates the record's index (row) in its corresponding `dc_users` table.



(c) Database Tree. We construct a tree shadowing the user tree, wherein we replace a given user tree nodes with a database node if that node's record is the final record in the databases's `dc_users` table. Any user tree node not replaced is considered an intermediate node, denoted i_n. Note that i_1 has one child. Also note that DB_3 and DB_4 share a taint, t_2, with DB_4 extending DB_3.



(d) Condensed Database Tree. We recursively merge any only-child leaf node with its parent. As such, intermediate nodes belonging to a single database (i_1) are squashed, and consecutive database downloads from the same controller (DB_3 and DB_4) are combined. Taints (t_2) ensure that only databases from the same controller operation are merged. i_0 and DB_1 indicate hack pack sharing, controller reversion behavior, or both. See Figure 6.7 for a fragment of this actual tree.

Figure 6.5. Illustration of the database inheritance tree reconstruction algorithm.

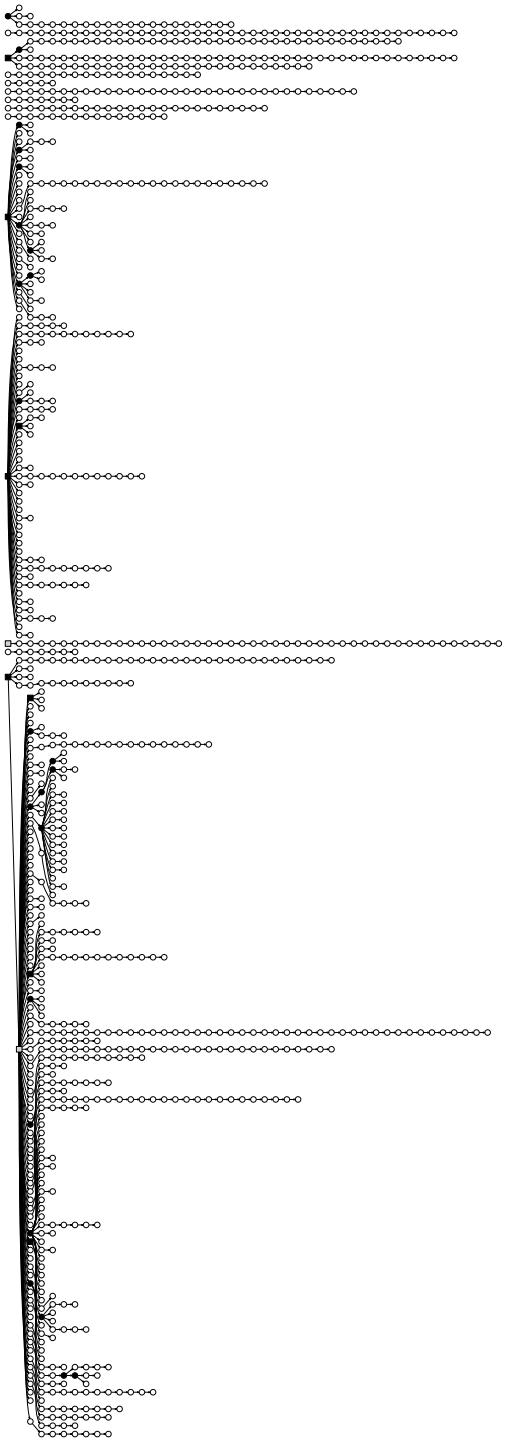


Figure 6.6. Fragment of the reconstructed DarkComet database phylogenetic tree, after the tree creation stage in Figure 6.5b. Open circles are individual, downloaded databases; shaded rectangles are known hack packs; opaque rectangles are inferred hack packs not part of our corpus of hack packs; opaque circles are single-controller reversion points.

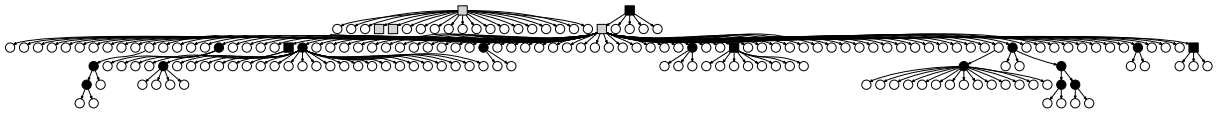


Figure 6.7. Fragment of the reconstructed DarkComet database inheritance tree. Open circles are sequences of databases downloaded from a single controller; grey rectangles are known hack packs; black rectangles are inferred hack packs not part of our corpus of hack packs; black circles are single-controller reversion points.

addresses, reducing the number of distinct controllers from 664 (identified by hostname only) to **590**. Having fully consolidated the controllers in our data set, we find that 72% of controllers used just one IP address; the remainder traversed multiple IP addresses during the window of observation. Further, 48% of controllers actively switched domain names during observation. In these cases, our methodology for controller tracking is necessary to accurately report on the number of controllers present.

Database Divergence

If two controllers start with the same initial database and then go on to acquire distinct victims, the two databases will share a common prefix of user records from the initial database, followed by distinct sequences of victims acquired by each controller. This is precisely what happens when two or more operators start from a common hack pack (Sections 6.2.2 and 6.3.1): their `dc_users` tables will each contain the set of victims inherited from the hack pack, followed by each operator's own victims. We use the term *divergence* to describe cases where two or more databases have a common non-empty prefix of victim records and different non-empty suffixes of victim records in their `dc_users` tables.

If, for two divergent databases in our corpus, there is no database containing their common prefix, we *infer* such an ancestor database and add it to our data set. The collected and inferred databases can now be arranged into a forest of trees representing database inheritance. The nodes of the inheritance tree represent databases, with an edge from a parent to child if the `dc_users` table of the parent is a prefix of the `dc_users` table of the child, that is if the child is derived from the parent. (Note that there are never points of *convergence* in the DarkComet

inheritance tree because there is no mechanism to combine the records from two databases into a new one, so the inheritance tree is indeed a well-formed tree.)

In addition to hack packs, databases may diverge when a controller *reverts* to an earlier version of the database. This happens when an operator runs the RAT controller software in a virtual machine and periodically restores the virtual machine state to an earlier snapshot. Unlike cases of database sharing (e.g., via hack packs), at most one database derived from a common ancestor by reversion will exist at any given point in time, while there may be multiple databases derived from the same hack pack active at a given point in time. In addition, databases related by reversion may be downloaded from a controller identified by the same hostname, while two different databases related by sharing should never appear on the same controller. Only 10% of controllers (61) exhibit this behavior; they reverted their databases 259 times in total during the observation period.

Figure 6.7 shows a fragment with two inheritance trees from our data set. Open circles represent databases downloaded in the course of the study. Inferred ancestral databases are shown shaded black: black circles denote inferred reversion databases and black squares denote inferred shared databases. Grey squares denote known hack packs (publicly shared databases). In all, the set of inheritance trees consists of 3,988 downloaded databases, 86 inferred ancestral databases related by reversion, 21 inferred shared databases that are not known hack packs and 17 known hack packs.

Hack Pack Prevalence

Of particular note is that 74% of controllers' databases are derived from an inferred hack pack, while 51% are based on one of the 17 hack packs we possess. This indicates both the prevalence of hack pack sharing in the DarkComet community, as well as the relatively few points of origin for DarkComet controller software downloads. We find that using a hack pack corresponds to both longer operational duration, as well as a higher number of victims; the median hack pack user accumulates five times as many victims and operates for eight times as

many days. All outlier attackers (covered in Section 6.5.1) acquired DarkComet from a hack pack.

Controller Attrition

We managed to download just a single database from about 40% of all controllers. We only downloaded 10 or more databases from 20% of controllers, and were able to download a database on each day of the measurement period from just 2%. Similarly, 55% of all controllers were observed for fewer than 5 days, calculated as the difference between the first and last download from a given controller, while just 22% were observed 30 or more days. (Recall that 40% of controllers were only seen once.)

Such high attrition led to sparse data collection from a large portion of controllers. We hypothesize that this phenomenon is a result of several factors. First, per Section 6.3.1, our downloader is visible to operators; we suspect that indication of discovery drives operators to abandon either DarkComet or their current infrastructure (e.g., domain names). Second, we believe that many DarkComet operations are inherently short-lived or even experimental, and that they thus expire quickly, regardless of our intervention; we explore this further in Section 6.5.1.

6.4.2 Identifying Victim Pollution

Correctly identifying and enumerating victims is essential to understanding the scope and severity of the DarkComet campaigns under observation. However, in Chapter 5 we found that there are "active participants" in the DarkComet ecosystem that impersonate victims, including malware sandboxes and network scanners (like ourselves). We fully expect these entities to be present in the victim records in our DarkComet databases' `dc_users` tables.

Fortunately, DarkComet victim impersonators are often identifiable. For instance, network scanners tend to violate subtle aspects of the DarkComet handshake protocol, particularly when crafting the `infoes` packet (recall Figure 6.2). As such, here we present a set of anomaly- and behavior-based rules for identifying and removing pollution from the `dc_users` table, so as

to accurately report on the *actual* DarkComet victim population.

The DarkComet Victim UUID

As stated in Section 6.3.1, DarkComet assigns each victim a universally unique identifier, or UUID. This UUID allows the attacker to connect keylogs and other stored metadata to a victim even if their IP address were to change or their operating system were to upgrade. The DarkComet stub generates a victim's UUID on installation. Specifically, the UUID is comprised of the victim machine's hardware ID [93], concatenated with 6 to 10 digits randomly selected upon generation. DarkComet uses the Windows registry and file system to persist a victim's UUID across system restarts, reinfection, and even infection by other DarkComet stubs. During our analysis in Section 6.5, we will consider a UUID to be equivalent to a victim; however, since imposter victims can fabricate their UUIDs to the controller, during pollution identification we will describe our reduction efforts in terms of *records*, that is, rows in `dc_users` tables.

Anomaly Detection

My collaborator, Mohammad Rezaeirad, developed following logic to detect anomalous victim entries based on the contents of their DarkComet handshakes, the contents of which make up the `dc_users` table. It is based on his work discussed in Chapter 5. Here, I present the results of that logic in order to provide context to the overall victim filtering procedure.

In the 3,988 databases' `dc_users` tables, there are 6,359,335 total records corresponding to 435,360 distinct UUIDs (although, as above, we do not consider UUID a useful field initially). Table 6.5 describes the anomalies by which we first detect and filter imposter victims, as well as the number of records filtered by each rule. Here we explain the intricacies of the denser rules.

Invalid UUID field

Over 96% of anomalous records' UUID fields were malformed in some way, likely because the DarkComet UUID is the most subtly complicated field crafted by the stub. Below is

Table 6.5. Records and UUIDs filtered by our anomaly detection logic. Many records exhibit more than one anomaly.

Description	Records		UUIDs	
Invalid UUID field	5,573,469	87.6%	329,895	75.8%
Matches known scanner (dc_toolkit)	4,839,445	76.1%	272,709	62.6%
Invalid userOS field	803,850	12.6%	64,831	14.9%
In hack pack	192,844	3.0%	8,916	2.0%
Matches suspected scanner	135,515	2.1%	31,262	7.2%
Missing expected keystrokes	128,065	2.0%	15,790	3.6%
Matches known scanner (ours)	39,578	0.6%	4,262	1.0%
Matches known sandbox	33,395	0.5%	2,961	0.7%
Invalid userIP field	10,638	0.2%	966	0.2%
Empty UUID field	761	<0.1%	1	<0.1%
Anomalous keystrokes	455	0.0%	38	0.0%
Empty userOS field	135	<0.1%	2	<0.1%
Invalid userName field	51	<0.1%	22	<0.1%
Empty userIP field	25	<0.1%	1	<0.1%
Empty userName field	25	<0.1%	1	<0.1%
Total anomalous victims	5,924,024	93.2%	385,096	88.5%
Total victims (hack pack)	124,445	2.0%	5,462	1.3%
Total victims (unique)	310,866	4.9%	**44,802**	10.3%
Total records	6,359,335	100.0%	435,360	100.0%

an example of a valid UUID, taken from Table 6.3.

$$\{\underbrace{\texttt{846ee340-7039-11de-9d20-806e6f6e6963}}_{\text{Hardware ID}}\texttt{-}\underbrace{\texttt{12345678}}_{\text{Suffix}}\}$$

The UUID field, as transmitted in the DarkComet handshake, should be bookended by parentheses. It should consist of the victim machine's hardware ID, often but not always concatenated with a unique suffix, joined by a dash if so. The hardware ID itself should be five, dash-joined fields of lengths 8, 4, 4, 4, and 12 characters, all lowercase hexadecimal. The hardware ID *must* include some non-digit characters. The unique suffix is only present when the hardware ID is non-random (as is typically the case in Windows 7 and above), and consists of six to ten random digits. Over 87% of all records in all downloaded databases violate these rules.

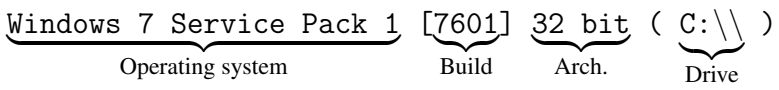
Scanners and sandboxes

The original DarkComet scanner (and database downloader) is Kevin Breen's `dc_toolkit` [11]. Between drawing from a fixed set of usernames and having protocol implementation errors like UUID malformation, victim records created by the `dc_toolkit` are trivial to detect. We find that a full 76% of records in our corpus of databases belong to the `dc_toolkit`, a testament to its popularity. Additionally, we test the `userIP` and `userName` fields for indicators of other known and suspected scanners that have contacted a DarkComet sinkhole we deployed independently. 2.1% of records belong to an apparent scanner that we suspect is a Russian anti-malware vendor based on naming convention; 0.6% belong to our scanner (our taints); and 0.5% belong to known malware sandboxes. We identify malware sandboxes by crafting custom DarkComet stubs that resolve to our sinkhole and submitting them to malware scanning services.[1]

[1]These services include Avira, Comodo, F-Secure, Fortiguard, Hybrid Analysis, Kaspersky, ViCheck, and VirusTotal.

Invalid userOS field

Here we show an example of the information transmitted in the `userOS` field.



Imposter victims tend to mismatch the operating system and build number or to use an operating system name not compatible with DarkComet. For instance, DarkComet does not have a valid name for Windows 10, so it substitutes `Unknow` (sic). Over 12% of all records have an invalid `userOS` field. Since the `userOS` field appears on the DarkComet controller's GUI, it is used by braggart scanners to communicate with operators. We even came across an `userOS` value: "`Thanks for your comet.db ( C:\\ )`"

Hack Pack Victim Detection

In Section 6.4.1 we described the process by which we inferred the existence of hack packs in addition to those we possess. We will consider victims shared in hack packs separately from victims belonging to individual campaigns. The hack packs we possess contain 373 victim records, 311 of which do not have any anomalies. Our inferred hack packs contain 8,743 victim records, 5,338 of which are not anomalous. As there is some overlap, the total number of UUIDs across all hack packs is 8,916 with 5,462 apparent real victims.

Keylog Validation

Using metadata from the keylog table, we attempt to filter short-lived, sandboxed executions of DarkComet and provide a conservative bound on the victim population. The `dc_keyloggers` table, as described in Section 6.3.1, contains a file per victim per day that keystrokes were logged. DarkComet's keylogging functionality cached keystrokes locally on the victim machine and dumps them, a file per day, to the controller's `dc_keyloggers` table when both machines are online simultaneously. Keylogging is enabled on all victims by default, so

we should expect that a real victim would have numerous days of keylogs in the table, while a sandboxed execution would have one or few. However, this is complicated by the fact that the operator can configure keylogs to be sent to an offsite FTP server (eliminating the need for the operator to be online to receive keylogs). However, if configured for offsite FTP, the database will contain a table for storing FTP credentials, which is generated on configuration. While we do not collect this table, we do record its presence in the database schema. Therefore, we bound our victim population based on the following conditions:

1. If a victim record has anomalous dates in the keylog table (e.g. year 2077), it is excluded (38 records).

2. If the FTP table exists, the database's victim records are included in analysis (37,641 records).

3. If the victim record has more than two days of keylogs, it is included in analysis (7,161 records).

4. Otherwise, the victim record is excluded. (15,790 records).

Applying all of the validations steps (Sections 6.4.2, 6.4.2, and 6.4.2) leaves a conservative estimate of **44,802 victims**.

6.5 Analysis

After eliminating records that we suspect may not be real victims as described above, our data set consists of 44,802 victims controlled by 590 controllers. Here, we report on the more interesting and troubling aspects of their relationship.

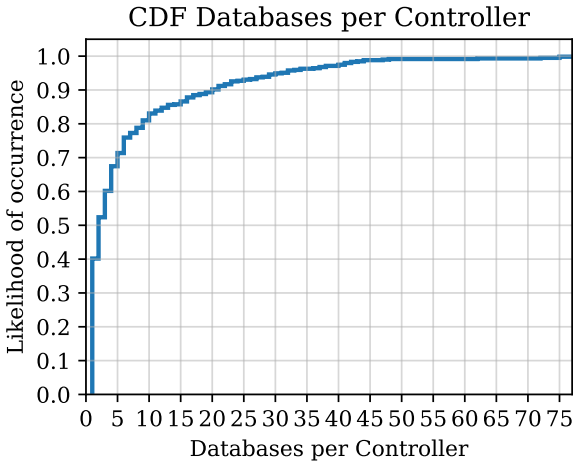


Figure 6.8. Total number of databases downloaded per controller.

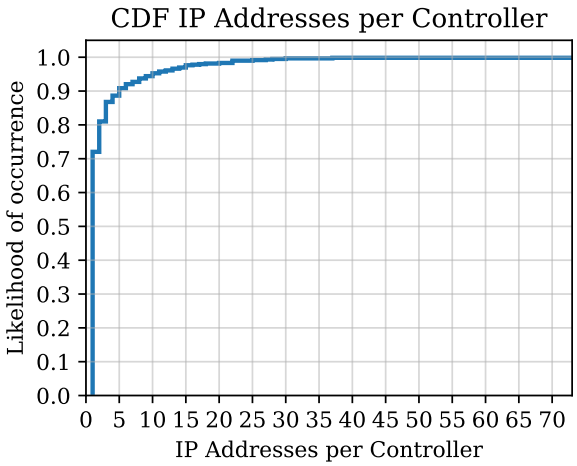


Figure 6.9. Total number of unique IP addresses per controller.

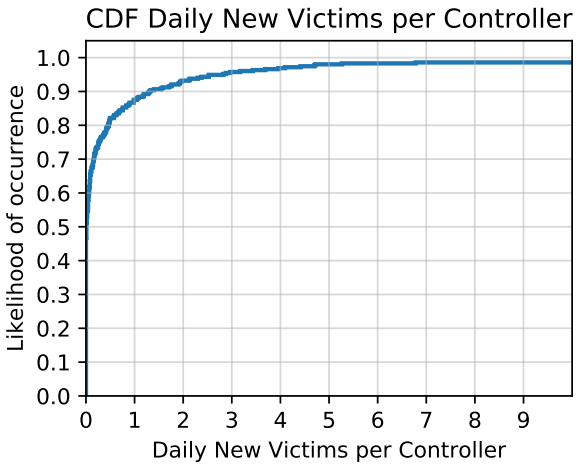


Figure 6.10. Daily new victim infection rate per controller.

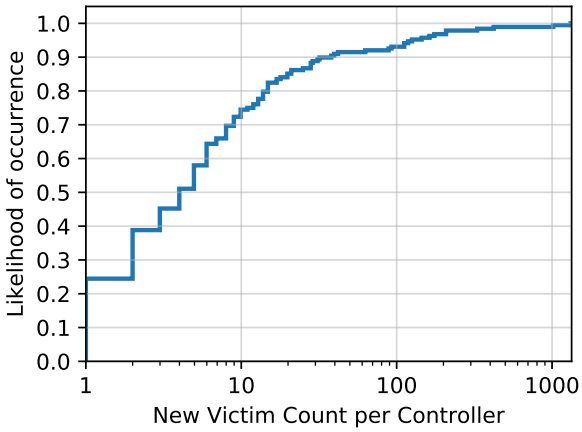


Figure 6.11. Total new victims per controller during observation period.

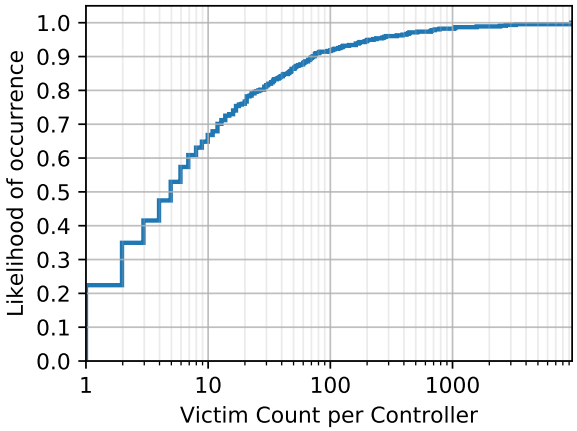


Figure 6.12. Total number of victims per controller.

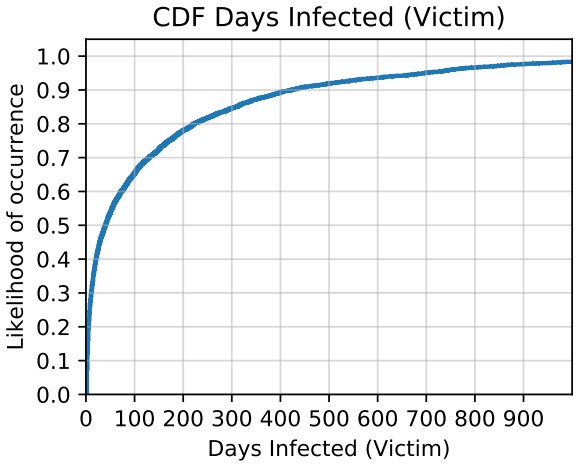


Figure 6.13. Days from victim first keylog to last keylog.

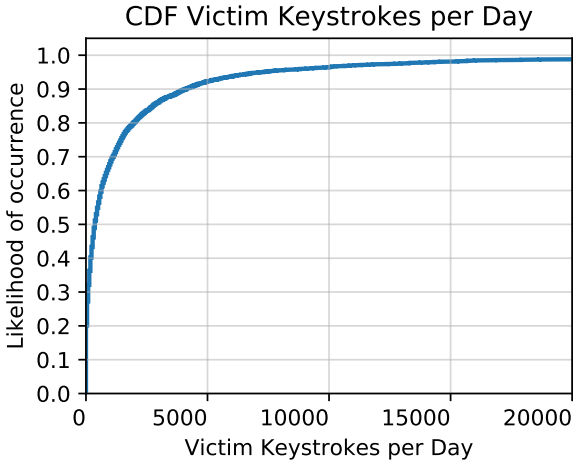


Figure 6.14. Victim keystrokes per day.

6.5.1 The DarkComet RAT Ecosystem at Scale

Victim Infection Rate

Over our 120-day data collection period, controllers added 5,869 new victims between the first database downloaded from each controller and the last, an aggregate rate of **50 new victims per day**, or a new victim every 29 minutes. Infection rates vary tremendously by controller. The average daily infection rate across all controllers is less than one victim a day; however, two controllers independently amassed more than 1,000 victims each during this same period, one of which infected 1,018 victims in just 12 days at a rate of *86 new victims per day*. Most controllers infected just a single victim over the entire period of observation.

Total Victims Infected

The total number of victims infected per controller over all time, excluding victims from known or inferred hack packs, is also dominated by a handful of outliers. In fact, two controllers in our data set have *over 9,000* victims each; the next closest has 3,328. Just 8 controllers have more than 1,000 victims, and only 38 have more than 100. A full 135 controllers have no real victims at all (at least, per our conservative filtering), and the median number of victims per controller is just 3. These findings are consistent with Chapter 5, where we found that a small number of outlier campaigns controlled hundreds of victims each, while a scant 14% of the domains they sinkholed even yielded a single victim.

Victim Infection Duration

We consider the time between a victim's first and last keylogs in a given database to be its infection duration, excluding those victims without keylogs. Figure 6.15 plots the cumulative distribution of victim infection duration, showing the probability of a victim being infected after a given number of days. At the time of our observation, half of all victims had been infected for over a month, with a median infection duration of 40 days. However, there is a small population of long-lasting infections, the longest of which is over 5 years old. Table 6.6 shows

Table 6.6. Victim operating systems.

OS	Victims	
Windows 7	24,008	53.6%
Windows 8	11,066	24.7%
Windows XP	5,609	12.5%
Windows 10	3,353	7.5%
Windows Vista	371	0.8%
Windows 8.1	366	0.8%
Other (2 systems)	29	0.1%
Total	44,802	100.0%

the percentage of victims broken down by operating system.

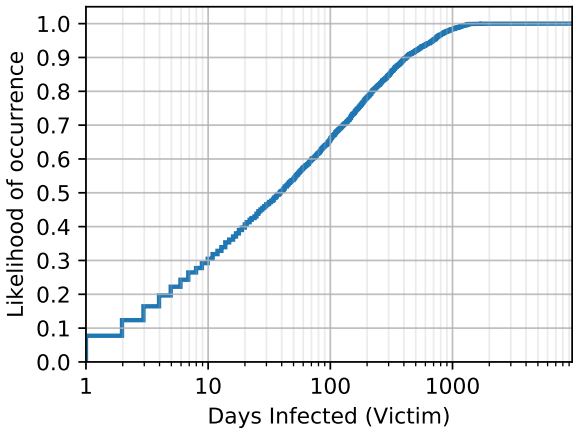


Figure 6.15. CDF of victim infection duration (n=4,687).

Campaign Longevity

We consider the operational age of a controller to be the time between when the first victim keylog was received by a controller and when its most recent database was download by our scanner. We only report on the 57% of controllers (334) that have keylogs, which allows us to effectively decouple our understanding of campaign longevity from our relatively limited window of observation. Figure 6.16 shows the cumulative distribution of controller operational longevity for controllers whose database was derived from a known hack pack and those whose

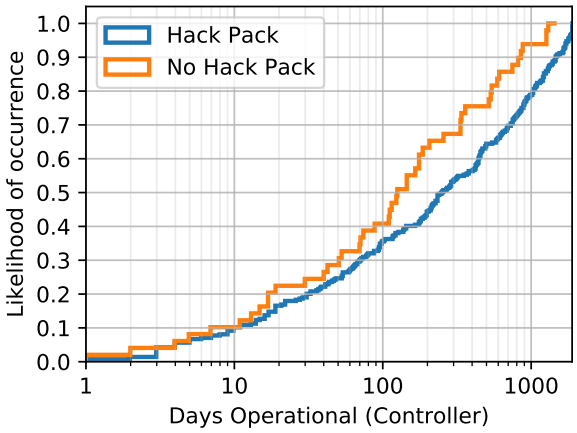


Figure 6.16. CDF of controller age (n=334).

database was not. The median longevity of controllers derived from hack packs, 252 days, is nearly twice that of controllers not using a hack pack (127 days). Over 40% of controllers in our data set have been operational for at least a year, and almost 10% have been functional for over 3 years. The longest lived controller in our data set has been infecting new victims and collecting keylogs for well over 5 years. The average operational duration was roughly 489 days, and the median was 231 days.

These results are biased towards controllers that have victims and have thus collected keystrokes (recall that 23% have no victims). Recall that in Section 6.4.1, we remarked on the high rate of controller attrition, that is, controllers disappearing after a small number of downloads. It may be that our downloader scares off some operators, or that some observed operations are inherently short-lived or experimental, but we cannot determine this with a right- and left-censored window of observation. Regardless, we observed two populations of controllers, transient and long-lived.

6.5.2 Quantifying Observed Harm to Victims

We quantify the harm incurred by the victims in our data set through several vectors: captured keystrokes, time monitored by attackers, non-consensual webcam accesses, and direct interaction or communication with attackers. Additionally, we glean operator motivations from

the labels they apply to victims and use these to further understand victim predicaments.

Captured Keystrokes

By default, DarkComet installs a keylogger on each victim's machine. As some RAT operators seek access to victim accounts and credentials (recall Chapter 4), and many are plainly opportunistic [97], victim keylogs can offer insight into the potential harms victims incur from DarkComet campaigns. In total, the DarkComet operators we observed have captured **186,748,463 keystrokes** from their victims. The median number of keystrokes captured per day was 382, while some active victims drove the average up to 1,896 keystrokes per day.

Table 6.7. Keystrokes captured and hours monitored, shown in aggregate and per victim. *Clipboard* refers to capturing text copied to the clipboard.

		Keystrokes		Hours	
Application	**Victims**	**Total**	**Rel.**	**Total**	**Rel.**
Clipboard	257	5,974,705	23,248	560.5	2.2
Web Browser	356	4,720,447	13,260	8,902.6	25.0
- E-Commerce	95	329,853	3,472	190.1	2.0
- Email	195	148,480	761	846.3	4.3
- Gaming	65	361,395	5,560	65.9	1.0
- Instant Messaging	65	155,286	2,389	193.1	3.0
- Social Networking	149	654,228	4,391	995.4	6.7
- Torrenting	13	1,924	148	44.4	3.4
Gaming	66	943,096	14,289	217.5	3.3
Email	88	213,870	2,430	420.5	4.8
System Tools	188	62,771	334	293.2	1.6
Instant Messaging	24	47,372	1,974	24.0	1.0
Productivity	63	42,632	677	206.2	3.3
DarkComet Chat	23	27,113	1,179	39.7	1.7
Malware	11	3,876	352	3.0	0.3
Antivirus	21	1,773	84	18.5	0.9
Entertainment	20	1,072	54	27.6	1.4
Torrenting	22	232	11	34.0	1.5
Unknown	367	5,993,378	16,331	5,650.1	15.4
Total	399	18,032,337	45,194	16,397.4	41.1

As described in Section 6.3.1, on March 25, 2019 we began collecting additional metadata

Table 6.8. Regular expressions and categories against which we compare victim active windows from `dc_keyloggers`. **DarkComet chat is handled as a special case, lest `chat` match all chat windows.

Regex	Categories	Regex	Categories
163	E-Commerce	naver	Search Engine
360	Technology	netflix	Entertainment
adaware	Antivirus	nicovideo	Video Sharing
adobe	Productivity	no-ip	Malware
aliexpress	E-Commerce	norton	Antivirus
alipay	E-Commerce	notepad	Productivity
amazon	E-Commerce	office	Productivity
antivirus	Antivirus	ok.ru	Social Networking
apex	Gaming	okezone	News
apple	E-Commerce	onenote	Productivity
avast	Antivirus	onlinesbi	Banking
avg	Antivirus	opera	Opera Browser
avira	Antivirus	panda	Antivirus
baidu	Search Engine	pandora	Entertainment
baidu browser	Baidu Browser	paypal	E-Commerce
bbc	News	photoshop	Productivity
bilibili	Video Sharing	pinterest	Social Networking
bing	Search Engine	pixnet	Social Networking
bitdefender	Antivirus	playerunknown	Gaming
blogspot	Social Networking	popads	Advertising
bongacams	Pornography	porn555	Pornography
booking	E-Commerce	pornhub	Pornography
chat**	DarkComet Chat**	program manager	System Tools
cisco	Productivity	pugb	Gaming
clipboard change	Clipboard	qq	Social Networking
cmd.exe	System Tools	quora	Social Networking
comodo	Antivirus	rakuten	E-Commerce
cs:go	Gaming	reddit	Forum
csdn	Social Networking	roblox	Gaming
darkcomet	DarkComet	sina	Email, Search Engine
ddos	Malware	skype	Instant Messaging
dolohen	Advertising	skyrim	Gaming
dropbox	Productivity	slack	Productivity
ebay	E-Commerce	so.com	Search Engine
espn	Entertainment	sohu	Email, Search Engine
ettoday	Entertainment	sophos	Antivirus
exosrv	Advertising	soso	Email, Search Engine
facebook	Social Networking	spam	Malware
fandom	Encyclopedia	spotify	Entertainment
fbcdn	Social Networking	stackoverflow	Productivity
fifa	Gaming	steam	Gaming
fortnite	Gaming	t.co	Social Networking
github	Productivity	taobao	E-Commerce
gmail	Email	teamspeak	Gaming
gmw	Technology	teamviewer	Productivity
google	Email, Search Engine	tianya	Forum
google chrome	Chrome Browser	tmall	E-Commerce
grand theft auto	Gaming	torrent	Torrenting
gta	Gaming	tribunNews	News
hao123	Video Sharing	tumblr	Social Networking
hotstar	Entertainment	twitch	Gaming
hulu	Entertainment	twitter	Social Networking
imdb	Entertainment	udp flood	Malware
imgur	Image Sharing	udp unicorn	Malware
instagram	Social Networking	vk	Social Networking
internet explorer	Internet Explorer Browser	vlc	Entertainment
iqiyi	Video Sharing	vpn	Productivity
itunes	Entertainment	warcraft	Gaming
jd	E-Commerce	weibo	Social Networking
kaspersky	Antivirus	whatsapp	Instant Messaging
keylog	Malware	wikipedia	Encyclopedia
league of legends	Gaming	wordpress	Social Networking
linkedin	Social Networking	xhamster	Pornography
live	Email	xinhuanet	News
livejasmin	Pornography	xnxx	Pornography
mail.ru	Email	xvideos	Pornography
messenger	Instant Messaging	yahoo	Email, Search Engine
microsoft	Email, Productivity	yandex	Email, Search Engine
minecraft	Gaming	youtube	Video Sharing
mozilla firefox	Firefox Browser	zonealarm	Antivirus
msn	News		

Table 6.9. Victim keystrokes for specific online accounts.

		Keystrokes		Hours	
Account	**Vic.**	**Total**	**Vic.**	**Total**	**Vic.**
Alibaba	15	1,949	130	8.4	0.6
Amazon	25	3,906	156	20.9	0.8
Apple	16	1,003	63	7.8	0.5
Dropbox	4	985	246	2.1	0.5
Ebay	23	4,334	188	98.2	4.3
Facebook	122	610,722	5,006	865.3	7.1
Github	4	108	27	0.4	0.1
Gmail	74	53,629	725	179.5	2.4
Google	100	50,330	503	497.9	5.0
Instagram	19	24,052	1,266	24.2	1.3
Linkedin	6	237	40	2.4	0.4
Mail.ru	9	3,295	366	8.0	0.9
Microsoft	110	38,508	350	187.3	1.7
Paypal	26	3,147	121	7.0	0.3
Skype	5	843	169	0.5	0.1
Slack	2	430	215	0.8	0.4
Steam	16	864	54	2.3	0.1
Twitter	13	2,670	205	4.3	0.3
VK.com	22	1,005	46	11.1	0.5
Whatsapp	22	41,560	1,889	63.2	2.9
Yahoo	43	14,482	337	53.7	1.2
Yandex	12	2,004	167	6.8	0.6
YouTube	217	186,417	859	1,514.2	7.0
Total	399	1,046,480	2,623	3,566.3	8.9

from the keylog table; we were able to do so for 399 recent victims from 73 controllers. DarkComet demarcates and timestamps stored victim keystrokes by the victim's active window. For each active window in a victim's keylogs, we record the specific number of keystrokes collected while the victim interacted with said window, as well as the time the victim spent using it. Additionally, we compare the active window name against a fixed list of application names (listed in Table 6.8 in the Appendix) designed to catch common applications and websites. We do not record the name of the active window, which may contain sensitive information, such as the title of a video the victim is watching on YouTube or the victim's email address shown in a

GMail window.

The 399 victims in this sample set had **18,032,337 keystrokes** captured over 2,592 days, amounting to over **16,394 hours** of keystroke monitoring. On average, DarkComet collected 45,194 keystrokes and recorded 41.1 hours of activity over 6.5 days from each victim. Table 6.7 provides a breakdown of the types of applications and websites active while keystrokes were collected, whereas Table 6.9 in the Appendix shows the keystrokes captured from some specific Internet services. These window names indicate that the keystrokes exfiltrated by the observed DarkComet campaigns contain sensitive information like emails, transcripts of private conversations, login credentials, and credit card numbers, putting victims at risk of blackmail or financial compromise. This is particularly concerning given the opportunistic and exploitative nature of DarkComet operators; in Chapter 4 we found that 43% of operators actively searched for victim passwords, and another 31% periodically checked victim keylogs.

Of note in Table 6.7 is that DarkComet's keylogger captures text copied to the clipboard, comprising the majority of keystrokes captures. *Malware* indicates that 11 victims were also users of malware, likely infected by backdoored tools they had downloaded. *Antivirus* is particularly interesting, as we could not find a correlation between supposed antivirus activity and infection cessation; we suspect that a few victims ran fake antivirus applications that were bundled with DarkComet, as they were the first active window recorded. *DarkComet Chat* is discussed next.

Direct Communication with Victims

An artifact of the keylogs we analyzed is that they capture when operators communicate with their victims. DarkComet offers the ability for an operator to spawn a chat window on the victim's desktop and communicate with them bi-directionally. Though the sample for which we recorded fine-grained keylog metadata only includes 399 victims from 73 operators, we find that on average operators opened a chat window with over 5% of their victims, and averaged nearly two hours per victim with whom they successfully engaged. This does not mean two hours of continued chat; rather, two hours of time with the chat window as the active window on

the desktop, during which an average 1,179 keystrokes were typed. The typical person types 200 keys per minute, so we estimate that roughly 6% of this time was spent actively engaging with the operator.

Only 22% of operators engaged in this behavior, but because we do not record chat keystrokes of either side (they may contain sensitive information), we cannot determine the purpose of this communication. In Chapter 4 we proposed that victim emotional harassment and extortion is part of the operators' value calculus and found that 18% of observed operators attempted to harass or extort their victims, which agrees with our finding of 22%.

Webcam Access

Described in Section 6.3.1, DarkComet's `config.ini` file encodes whether an operator has issued commands to a specific victim and whether the operator has accessed the victim's webcam. It also includes the automated tasks the operator has configured stubs to perform upon connection. We were able to download the `config.ini` file from 204 of the 590 controllers in our data set, encompassing 23,722 total victims. We find that operators issued commands to 18,625 (or 79%) of their victims and accessed the webcams of 6,271 (26%) victims they actively controlled. Though webcam access suggests a voyeuristic motive, we do not know how much time the controller spent accessing the webcam and cannot differentiate between webcam access for machine vetting versus voyeurism.

In Chapter 4, we found that 61% of attackers in their study attempted to access the victim's webcam, which is more than twice the frequency we observed. We note, however, that that chapter reported the number of *attempts* to access the webcam, while our data shows the number of victim's whose webcam was accessed *successfully*. In Chapter 5, we found that only 53% of victims had accessible webcams, which would put the number of successful accesses at about 32% using their data, compared to the 26% observed in our data.

Victim Automation

Just 13 of the 204 controllers (6%) assigned automated tasks to their victims, the most popular of which was to launch a denial-of-service attack (specifically, a UDP flood) against a specified target. Similarly, in Chapter 4 we observed only 1% of operators using DarkComet to launch DDoS attacks.

Table 6.10. Controller group suspected motivations.

Group Category	Ctrl.	Vic.
Webcams or Voyeurism	31	358
Targeting Individuals or Groups	12	2,108
Mining, Hacking, or Malware	9	1,123
Account Credentials	4	20

Victim Labels

80 of the 590 controllers' operators annotate their victims using group names, which are recorded in the `dc_groups` database table. 17,917 victim records have one such label, including 8,055 of the 44,802 victims we consider legitimate. (Note that operators can, and do, label victims our filtering considers illegitimate.) The use of group names is dominated by a single, French-speaking operator who manually assigned 15,544 victims (6,580 of which we consider legitimate) to 32 unique groups. Manual interpretation of group names and their underlying

Table 6.11. Controller group suspected languages.

Language(s)	Controllers
English	45
Turkish	5
English or Turkish	3
English or French	1
English, French, or Russian	1
Unknown	25
Total	80

Table 6.12. Example group names and translations from one French-speaking operator.

Group Name	Translation	Victims
Partition cryptée à surveiller	Encrypted partition to watch	1
disque externe manquant	missing external disk	4
grosse base de données	big database	3
Disque externe à surveiller!	External disk to watch!	7
Win pas Français	Win[dows] not French	163
pourrave étranger	may be foreign	33

motivations provides interesting anecdotes. For instance, the French operator had labeled 9,725 victims "virus total," presumably indicating that they are suspected sandboxes. (Our filtering, described in Section 6.4.2, identified over 88% of these as non-legitimate.) Another operator had a group titled "Internet Cafe Pc," possibly revealing a particular target. The French-speaking operator with 6,580 meticulously-labeled victims appeared to be monitoring victims with external drives and encrypted partitions, based on labels such as *Partition cryptée à surveiller*" (French: "Encrypted partition to watch").

Group labels often reveal operator language and intent, at least for the 14% of operators that use them. Most operators appear to speak English, with some also using Turkish, French, or Russian vocabulary. 39% of labels indicated voyeuristic intentions (e.g. webcam access). 15% suggest targeting specific individuals; 11% the deployment of other malware or hacking tools; and 5% credential theft. The voyeuristic motivations and propensity to target individuals (often by name) align closely with anecdotes of RAT usage for perverse acts such as sextortion [29, 31]. Specifically, in Chapter 4 we found that 45% of operators exhibited voyeuristic tendencies, while 39% of our operators had at least one label suggesting the same. Though less common, the use of victim machines to deploy offensive malware or cryptocurrency mining clients also aligns with our previous finding that 16% of operators used the victim machine for its network vantage point of some kind, while 11% of our operators had groups indicating the deployment of secondary payloads like miners or DDoS tools.

6.6 Discussion

6.6.1 Understanding Victim Harm

Understanding the harms incurred by victims of low-volume malware infections is challenging. Our system's data collection and automated analysis methods allow for quantifying these harms at scale, in terms of disquieting metrics like keystrokes stolen and hours monitored per application, webcam accesses made, and amount of direct communication initiated.

6.6.2 Campaign Tracking

Our method of obtaining victim databases from RAT controllers combined with our lineage analysis technique enables us to identify distinct RAT campaigns across any number of IP address and domain name changes, assuming a controller represents a RAT campaign. This allows us to better understand the size and dynamics of RAT campaigns, including two campaigns with over 9,000 real victims. This type of information can help security researchers perform attack attribution, or law enforcement prioritize investigations.

6.6.3 Real Victim Determination

The pollution reduction heuristics we developed enable us to reduce our initial set of 435,360 potential victims by around 95%, leaving us with 44,802 likely real victims. This allows us to more accurately understand the longitudinal relationship between attackers and victims at a large scale, something that we could not analyze in the previous chapters due to limited data sources. Our analysis confirms prior results indicating that many RAT attackers and corresponding victims are located in the same country or a bordering one, as we will discuss in Chapter 7. In addition, it shows the rate at which RAT operators tend to add victims and gives us other insights into active RAT campaigns. As our data processing methodology is not dependent on our form of data collection, it could be used in other scenarios. For instance, law enforcement acting on search warrants could use this technique to expedite victim notification.

6.7 Ethical & Legal Considerations

Part of our data collection methodology emulates DarkComet victims and uses the well-documented DarkComet file download API to retrieve a copy of the victim database from controllers. Before employing this methodology, we consulted with the general legal counsel at our institution, who confirmed that our methodology was legal based partly on the fact that we were using existing functionality that is accessible to any DarkComet stub, and that we were thus not "exceeding authorized access," part of the Computer Fraud and Abuse Act (CFAA) U.S. legal statute. In addition to consulting with our legal general counsel, we also submitted our protocol to our Institutional Review Board (IRB). As part of our protocol, we remove or hash all fields from the database that might contain personally identifiable information (PII), such as the keystroke logs. We also hash other sensitive fields like the victim IP address or user name, which is often the computer owner's name. These redacted copies of the databases were stored on a server with strict access controls and an encrypted file system. Our IRB exempted our study since we neither store nor analyze PII.

Although acceptable from a legal perspective and exempted by our IRB, one might still argue the ethics of our data collection methodology. Here, we present our framework for data collection and analysis within the ethical guidelines described in the Menlo Report [32], which is in turn based on the 1979 Belmont Report [101], and is a cornerstone for computer and information security research. This framework is based on four principles: respect for persons, beneficence, justice, and respect for law and public interest. Our framework addresses each of these principles as follows.

Respect for persons. Since "participation" in this study is not voluntary and cannot be based on informed consent, we take great care not to analyze victim PII, as they form the most vulnerable party involved. We only compile aggregate statistics about victims.

Beneficence. We believe that our analysis does not create further harm. The method we use to collect our data has been well-publicized in prior public reports and talks [28, 15, 16, 11, 52]. We

feel the benefits of a better understanding of RAT operators and victims outweigh the potential harms of publishing aggregate statistics.

Justice. The benefits of this work are distributed to the wider public, in terms of helping to reduce crime. The study particularly helps protect persons who are vulnerable to being victimized by RATs. We see no impact to persons from being included in the study.

Respect for law and public interest. We describe the legal framework for data collection and argue that it is in full compliance with U.S. laws. In addition, the researchers that participated in this project have obtained an exemption from their IRB. It is important to note that, while captured information may point to certain illegal conduct, establishing legal proof of criminal conduct is *not* the purpose of this study.

6.8 Limitations

Our data collection methodology is currently limited to DarkComet; however, prior work indicates that a number of other RAT families expose the same arbitrary file read functionality [52], suggesting that our data collection methodology could scale to many other RATs. Further, like DarkComet, most RATs maintain files or databases of victim metadata. While they may not expose the same download capabilities, they are still often shared in hack packs. If we could scale our collection of hack packs (e.g., through more access to VirusTotal and other malware upload repositories), this could enable our analysis methodology to expand to additional RAT families. Further, as the data processing techniques we debuted in Section 6.4 are independent of our data collection technique, they can be applied to data obtained otherwise (e.g., by legal seizure).

6.9 Conclusion

In this work, we presented a broad study on the victims of RAT malware. To carry out the study, we used a feature of the DarkComet RAT controller software that allows anyone to

download the database of the victims of a controller. Using his capability, we collected 3,988 databases from 590 unique controllers. To arrive at our data set for analysis, we developed new methods for tracking controllers and identifying real victims in the presence of honeypots, scanners, and VM executions of malware by researchers. Using this data, we presented the results of our analysis of the controllers, victims, and the relationship between them. Among our most troubling findings is that there are at least 44,802 victims of DarkComet, and DarkComet operators acquire 50 new victims per day.

6.10 Acknowledgements

Chapter 6, in part, has been submitted for publication of the material as it may appear in Proceedings of the 26th ACM Conference on Computer and Communications Security, 2019. Brown Farinholt, Mohammad Rezaeirad, Damon McCoy, Kirill Levchenko, 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 7

Examining the Geographic Relationship between Attackers & Their Victims

Each of our measurement experiments (described in Chapters 4, 5, and 6) contributed information revelatory of the relationship between RAT operators and their victims. In this chapter, we examine a prominent aspect of that relationship, geographic location.

7.1 VirusTotal Sample Submissions vs. Operators

In Chapter 4, VirusTotal allowed us to retrieve the geolocation of the IP addresses that were used to upload the DarkComet samples we executed. In Figure 7.1, we show the correlation between the countries of sample uploaders (who we suspect may be potential victims) and the countries of controllers that connected to our honeypots. For clarity, we discard the DarkComet samples uploaded or operated from multiple countries, comprising 5 and 260 samples, respectively. We see in Figure 7.1 that the two most popular countries for controllers are Russia and Turkey, with a clear diagonal indicating that DarkComet samples tend to be uploaded and controlled from the same countries. For example, 34% and 52% of all samples uploaded from Russia and Turkey, respectively, were controlled from the same country. The vertical lines for the US, France, and Canada are indicative of users uploading DarkComet samples in bulk. As these samples were likely acquired from users residing in different countries than these uploaders, the correlation between uploader and controller countries is weaker in those cases.

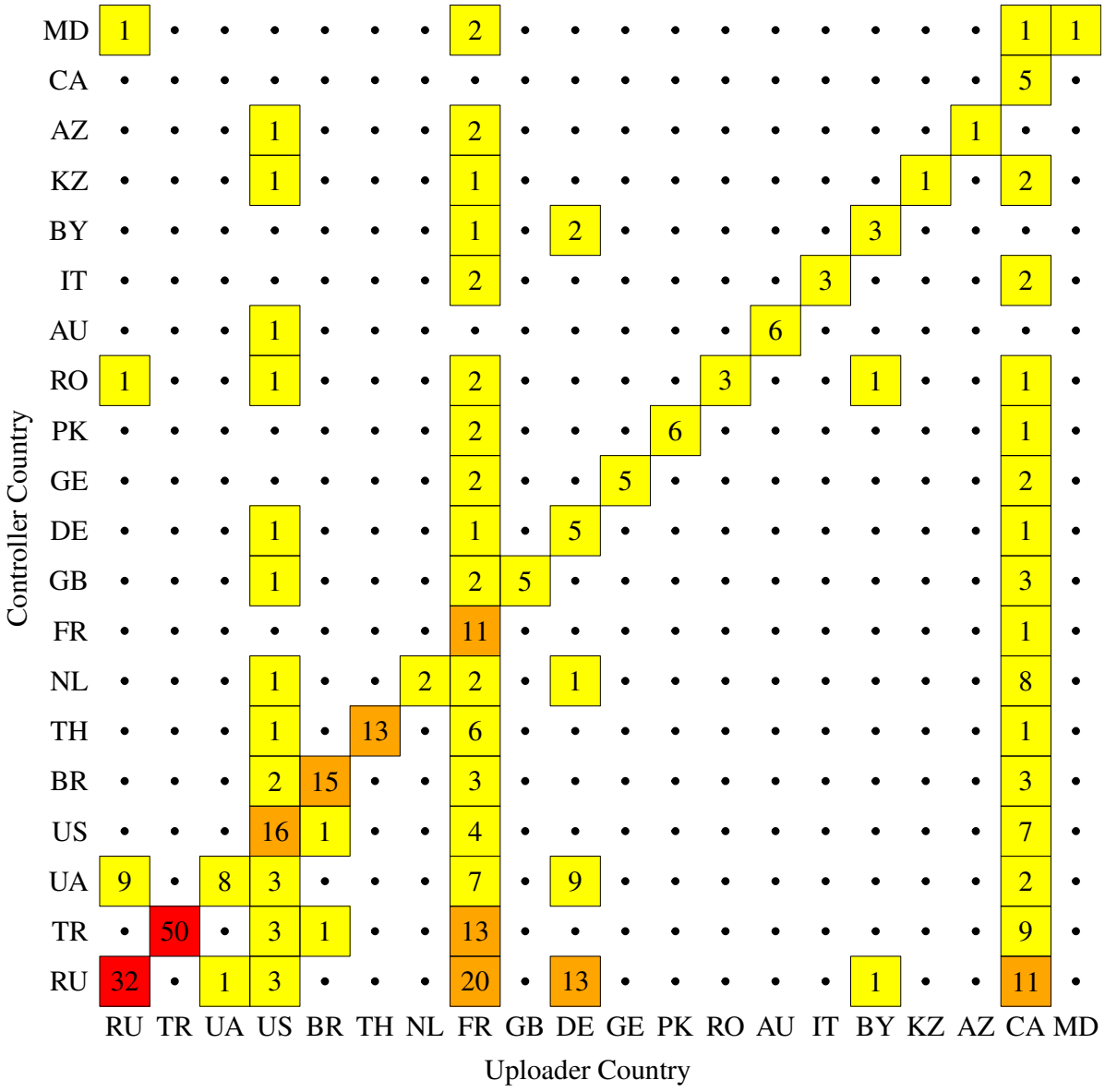


Figure 7.1. Correlation between the geolocated countries of the VirusTotal uploaders and those of the controllers accessing our honeypots. Countries are sorted by decreasing number of controllers.

7.2 DDNS Domain Resolutions vs. Victims

In Chapter 5, we geolocate victims based on the source IP addresses of their connections. Their presumed operators, however, are slightly harder to locate. We were able to detect some active controllers with RAT-SCAN before claiming their domains; for these, we can perform an accurate location comparison. However, for many of the domains we sinkholed (even those that provided victims), we never detected their controller. For these situations, we instead use passive DNS to look up previous IP addresses to which the domain name resolved, and use their locations for the comparison. All IP-based geolocations were performed using MaxMind's GeoIP2 Precision Insights service.

First, we filter controllers using proxies. MaxMind provides information regarding the likelihood that an IP address is a proxy, as well as IP ownership (which can be used to manually determine proxies). We use this information to separate proxies from non-proxies, as in Tables 7.1 and 7.2. A large portion of the controllers in our data set appear to be utilizing proxies from certain countries like France, Sweden, and the U.S. We manually investigated the largest in Section 3.5.4. In short, we find two VPN providers (IPjetable [63] and Relakks [117]) account for 40% and 3% of all actively-probed controllers, respectively, while prominent VPS services like Amazon AWS, Microsoft Azure, and Digital Ocean are also frequently abused. As the geolocation results of the proxies only serve to muddle the geospatial relationships between victims and attackers, we filter them from the following analyses. We report only on those results in the *Other* columns of the geolocation tables.

Tables 7.1 and 7.2 show the geolocations of historic and actively-probed controller IP addresses, respectively. We find both to have heavy presences in North Africa and the Middle East. Outliers include Brazil and Russia, both of which tend to correspond with victims in bordering nations.

Exploring Table 7.3, we find that virtually every country has some RAT victims with Brazil being the top location for victims of both DarkComet and njRAT, as shown in Table 7.3.

Table 7.1. Geolocations of historic controller IP addresses based on DNS history.

njRAT			**DarkComet**		
Country	Proxy	Other	Country	Proxy	Other
France (FR)	3,829	69	United States (US)	4,552	1,881
United States (US)	714	167	France (FR)	2,771	1,623
Sweden (SE)	433	19	Sweden (SE)	1,051	318
United Kingdom (GB)	160	63	Netherlands (NL)	706	256
Canada (CA)	152	12	Germany (DE)	511	3,077
Netherlands (NL)	96	9	United Kingdom (GB)	487	1,494
...			...		
Algeria (DZ)	22	7,820	Turkey (TR)	130	21,913
Brazil (BR)	42	7,206	Russia (RU)	233	17,020
Egypt (EG)	27	5,655	Algeria (DZ)	13	13,202
Morocco (MA)	3	4,293	Morocco (MA)	2	6,693
Iraq (IQ)	5	2,001	Egypt (EG)	4	4,872
Tunisia (TN)	0	1,504	Saudi Arabia (SA)	0	4,491
Saudi Arabia (SA)	0	1,297	Ukraine (UA)	75	3,971
Indonesia (ID)	8	732	Brazil (BR)	78	3,257
Libya (LY)	0	682	Pakistan (PK)	28	2,935
Other	524	6,113		1,921	36,919
Total	6,015	37,642		12,562	123,922

Table 7.2. Geolocations of probed controller IP addresses.

njRAT			DarkComet		
Country	Proxy	Other	Country	Proxy	Other
France (FR)	2,625	4	France (FR)	258	41
Sweden (SE)	184	0	Sweden (SE)	16	0
United States (US)	16	2	United States (US)	12	6
...			...		
Brazil (BR)	2	441	Turkey (TR)	0	594
Morocco (MA)	0	382	Ivory Coast (CI)	0	207
Algeria (DZ)	0	281	Russia (RU)	11	201
Egypt (EG)	0	178	India (IN)	1	128
Korea (KR)	0	80	Thailand (TH)	0	102
Tunisia (TN)	0	65	Vietnam (VN)	0	88
Iraq (IQ)	0	58	Ukraine (UA)	8	63
Saudi Arabia (SA)	0	52	Egypt (EG)	1	41
Thailand (TH)	0	39	Azerbaijan (AZ)	0	37
Turkey (TR)	0	37	Malaysia (MY)	0	33
Other	17	121		35	156
Total	2,844	1,740		342	1,697

Table 7.3. Geolocations of victim IP addresses.

njRAT			DarkComet		
Country	#Src-IP	#FP	Country	#Src-IP	#FP
Brazil (BR)	2,416	1,070	Brazil (BR)	318	178
Egypt (EG)	331	94	Turkey (TR)	188	130
Iraq (IQ)	207	82	Russia (RU)	184	127
Argentina (AR)	138	62	Ukraine (UA)	44	38
Algeria (DZ)	149	60	Egypt (EG)	74	36
Peru (PE)	131	55	Poland (PL)	28	26
Vietnam (VN)	117	53	Philippines (PH)	22	21
United States (US)	54	47	Thailand (TH)	35	17
Venezuela (VE)	105	47	Vietnam (VN)	16	14
India (IN)	88	46	Algeria (DZ)	21	13
Turkey (TR)	93	40	Bosnia (BA)	17	13
Thailand (TH)	189	38	Indonesia (ID)	12	11
Mexico (MX)	66	37	India (IN)	11	10
Other	1,401	659		265	207
Total	5,485	2,390		1,235	841

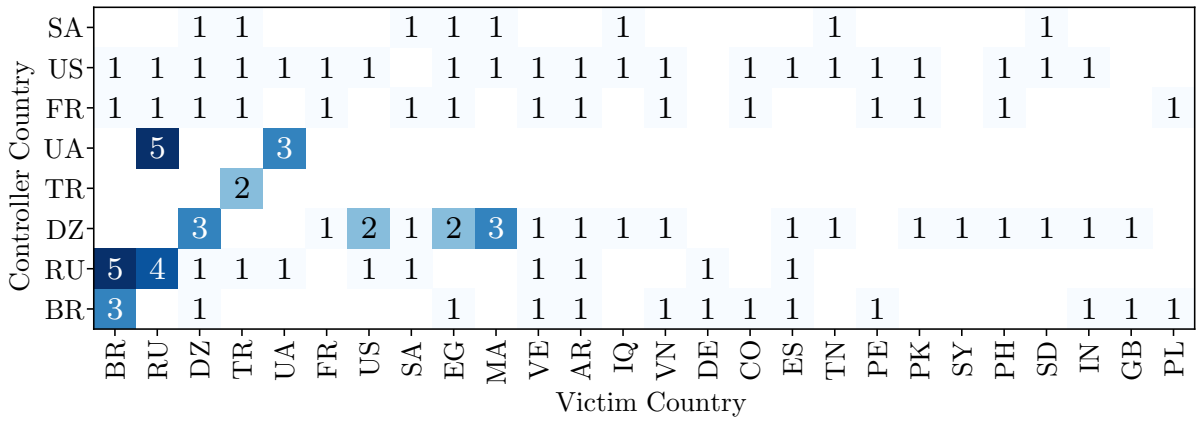


Figure 7.2. Relational matrix comparing geolocations of actively-probed controller IP addresses to received victim IP addresses, per sinkholed domain. Proxy IP addresses are filtered.

We find what appears to be geographic concentrations of DarkComet and njRAT victims in South America and North Africa / Middle East, including some bordering countries. We also find that DarkComet is used to infect a larger percentage of victims in Russia and bordering countries. Note that these measurements might be biased by our methodology of acquiring RAT samples and sinkholing DDNS domains.

Recall that during the sinkholing portion of the experiment, we registered the command-and-control domain, directing all potential victims to our server. During this period, we were able to observe all victims that attempted to connect to the controller. Prior to the sinkholing period, controller domains may have been held by the original controller or may have been sinkholed by researchers or vigilantes. In addition, for four and a half months prior to the sinkholing experiment, we resolved all controller domains to determine whether they were registered, and, if registered whether they had an associated A record, and whether the corresponding hosts behaved correctly (as a controller). Thus, for each domain, we have the IP addresses of all controllers that held the domain, as well as of all victims that attempted to connect to the domain during the sinkholding period. (Note that two periods are necessarily disjoint: both we and the original controller cannot hold the same domain at the same time.) Figure 7.2 shows the geographic relationship between responsive controllers and the victims, using the geolocation

methodology above. Each cell of the matrix shows the number of distinct campaigns (domains) associated with the given country pair. In cases where a domain resolved to more than one country or where victims where located in more than one country, the domain contributed a fractional weight to each cell in proportion to the number of controller-victim pairs of the domain from the country pair, so that the total contribution of each domain was 1. Figure 7.2 shows only the top 25 countries, ordered by the greater of the number of victims and controllers in the country. The dominant feature of the data is the controller and victim being located in the same country, visible as a concentration around the diagonal in the matrix. In addition, there were 5 campaigns with a controller in Ukraine (UA) and victims in Russia (RU). This may be due to a common infection vector, as Ukraine has a large Russian-speaking population and its users may frequent the same Russian-language sites. The incidence of controllers and Russia and victims in Brazil (BR) is more puzzling; although both Russia and Brazil have large victim and controller populations, there is no obvious reason why controllers in Russia might target victims in Brazil specifically. Another possibility is that the controllers were using a proxy in Russia that was missed by our filtering.

Figure 7.3 shows the same type of data, but for all controllers using the historic controller dataset. Note that this data spans the period 2010 to 2017 and includes name resolution from passive DNS sources, where we did not verify the correct behavior of the controller. As such, this data should be interpreted with caution. Figure 7.3 exhibits the same concentration around the diagonal as Figure 7.2, indicating campaigns where both controller and victim are in the same country. As the results of Table 7.3 suggest, Brazil has by far the largest concentration of victims across both RATs. Moreover, Brazil appears to be victimized indiscriminately. We also note some language clustering, where countries that speak the same language or are geographically proximate are more likely to be paired; e.g., Russia on Ukraine (13), Ukraine on Russia (18), Ukraine on Kazakhstan (4), Ukraine on Belarus (5), Morocco (MA) on Algeria (DZ) (9), Algeria on Morocco (8).

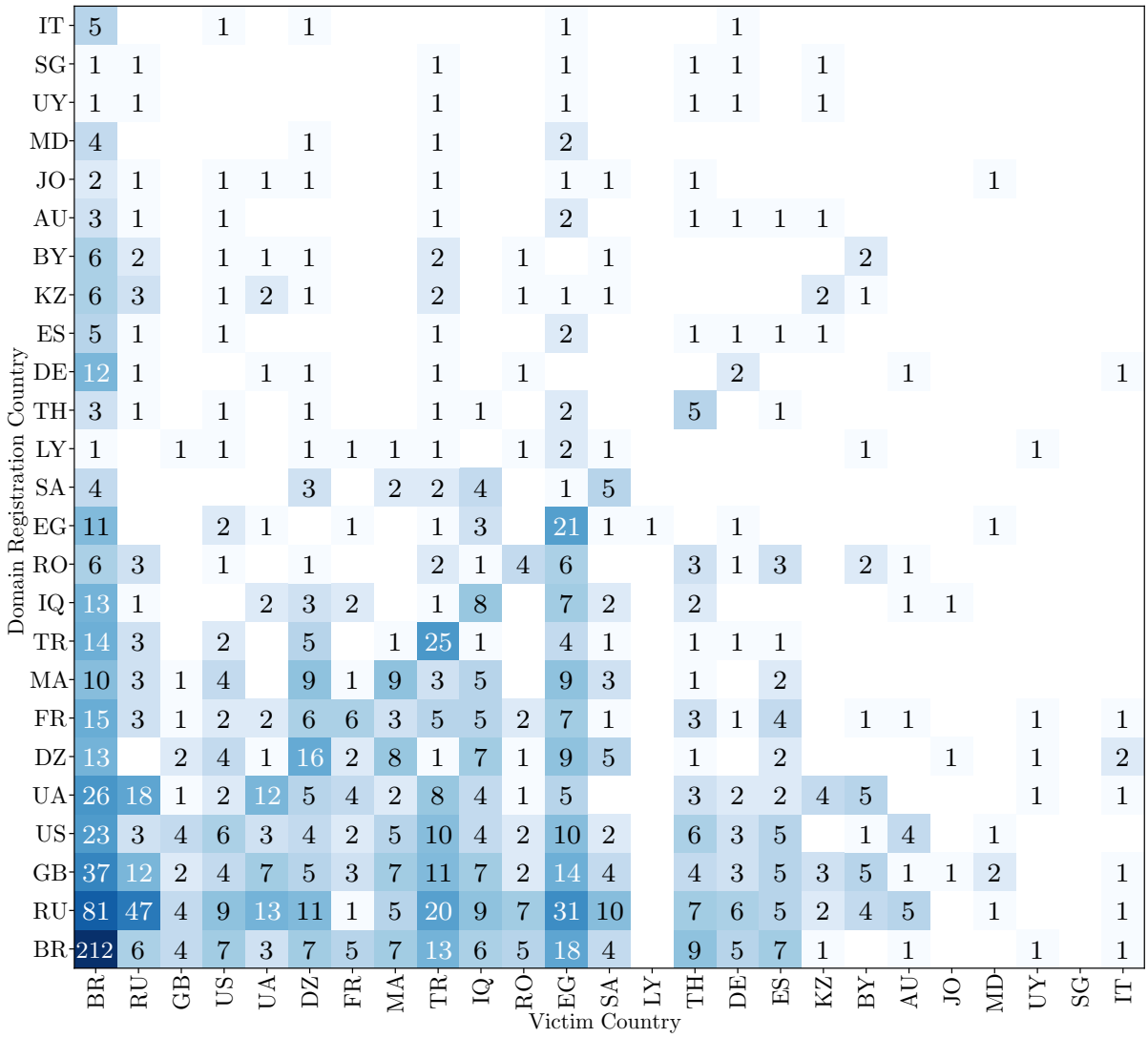


Figure 7.3. Relational matrix comparing geolocations of historic controller IP addresses to received victim IP addresses, per sinkholed domain. Proxy IP addresses are filtered.

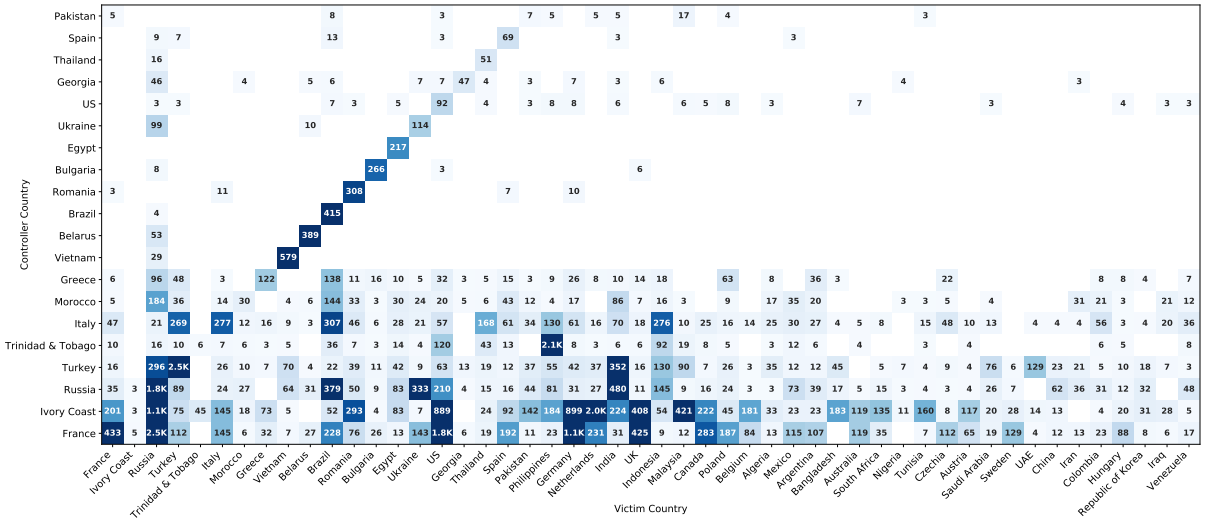


Figure 7.4. Number of victims for each combination of controller and victim country. Rows denote controller countries and columns victim countries, based on geo-located IP address (excluding VPN providers).

7.3 Operators vs. Victims

In Chapter 6, our methodology finally gives us a uniquely accurate dataset that links victims and operators, allowing us to understand the geographic relationship between them. In order to preserve victim anonymity, victim IP address geo-locations are resolved against a locally-stored MaxMind GeoLite2 City database, which requires no queries to an external geo-location service. We geo-locate attacker IP address using MaxMind's Precision Insights service, which is advertised as being more accurate [89]. Table 7.4 shows the number of controllers in each country together with their footprint, which is the number of victims whose controller is in that country. Countries are listed in decreasing order of the number of controllers. Nearly 15% of operators used a known anonymizing service like a VPN or VPS, predominantly IPjetable (VPN) and Amazon AWS (VPS), making true geo-location impossible for these controllers; such controllers are counted separately in the *Anonymous VPN* row. Controllers whose location we could not determine are counted in the *Other* row.

Turkey has the largest controller population, with 23.9% of all controller IPs geo-located

Table 7.4. Sample of controller locations and victim counts. Controller footprint is the total number of victims controlled by all controllers from a given country.

	Controller					
Country	**Count**		**Footprint**		**Victims**	
Turkey	141	23.9%	4,877	10.9%	3,223	7.2%
Russia	116	19.7%	5,024	11.2%	6,632	14.8%
Ukraine	47	8.0%	250	0.6%	726	1.6%
France	17	2.9%	9,915	22.1%	1,023	2.3%
Italy	13	2.2%	2,613	5.8%	694	1.5%
Brazil	12	2.0%	438	1.0%	1,918	4.3%
United States	11	1.9%	242	0.5%	3,477	7.8%
Germany	11	1.9%	38	0.1%	2,527	5.6%
Ivory Coast	4	0.7%	9,602	21.4%	41	0.1%
Netherlands	2	0.3%	10	¡0.1%	2,522	5.6%
India	1	0.2%	63	0.1%	1,701	3.8%
Trinidad and Tobago	1	0.2%	2,738	6.1%	60	0.1%
Philippines	1	0.2%	1	¡0.1%	2,747	6.1%
Anonymous VPN	78	13.2%	3,693	8.2%	—	—
Other	135	22.9%	5,298	11.8%	17,511	39.1%
Total	590	100.0%	44,802	100.0%	44,802	100.0%

there, followed by Russia (19.7% of controllers) and Ukraine (8.0% of all controllers). These findings agree with the findings in Section 7.1, where we found that 37% of their controllers were geo-located in Turkey and 15% in Russia. While these two countries account for 43.6% of all controllers, their operators control only 22.1% of all victims. A handful of controllers in France and Ivory Coast control 43.5% of all victims. Most of these victims are controlled by a *single* operator; indeed, 3 controllers control half of all victims.

Returning to Table 7.4, the *Victims* column shows the number of victims in each of the countries listed. Although the table is ordered by number of *controllers*, the country with the largest number of victims not shown in the table is the United Kingdom, with 1,040 (2.3%) victims. The largest number of victims is location in Russia, which also has the second-largest number of operators. On the other hand, countries such as Trinidad and Tobago and Ivory Coast have very few victims (60 and 41, respectively), but a large footprint (victims controlled by an operator in the country): 2,738 and 9,602, respectively.

Figure 7.4 shows the number of victims for each combination of controller country and victim country. The vertical axis enumerates controller countries (ordered by the controller footprint), while the horizontal axis enumerates victim countries, in the same ordered as controller countries. The horizontal axis extends to include additional countries in order of decreasing number of victims. Figure 7.4 bears evidence of cross-border heavy-hitters as vertical bands: operators in France, Ivory Coast, Russia, Turkey, and Italy control large victim populations spanning many countries.

In Section 7.2, we evince that attackers and victims in the commodity-grade RAT ecosystem are often co-located, that is, in the same country or region. The propensity for some RAT operators to target people they know likely influences this, as do shared language and culture. As Figure 7.4 shows, there is also a tendency for controllers and be in the same country as their victims, although the majority of victim hosts (82%) are *not* in the same country as their controller. The converse, however, is not true. Because Figure 7.4 is weighted by victims (each victim contributes 1 count to the numbers shown), it emphasizes the heavy-hitters with a global

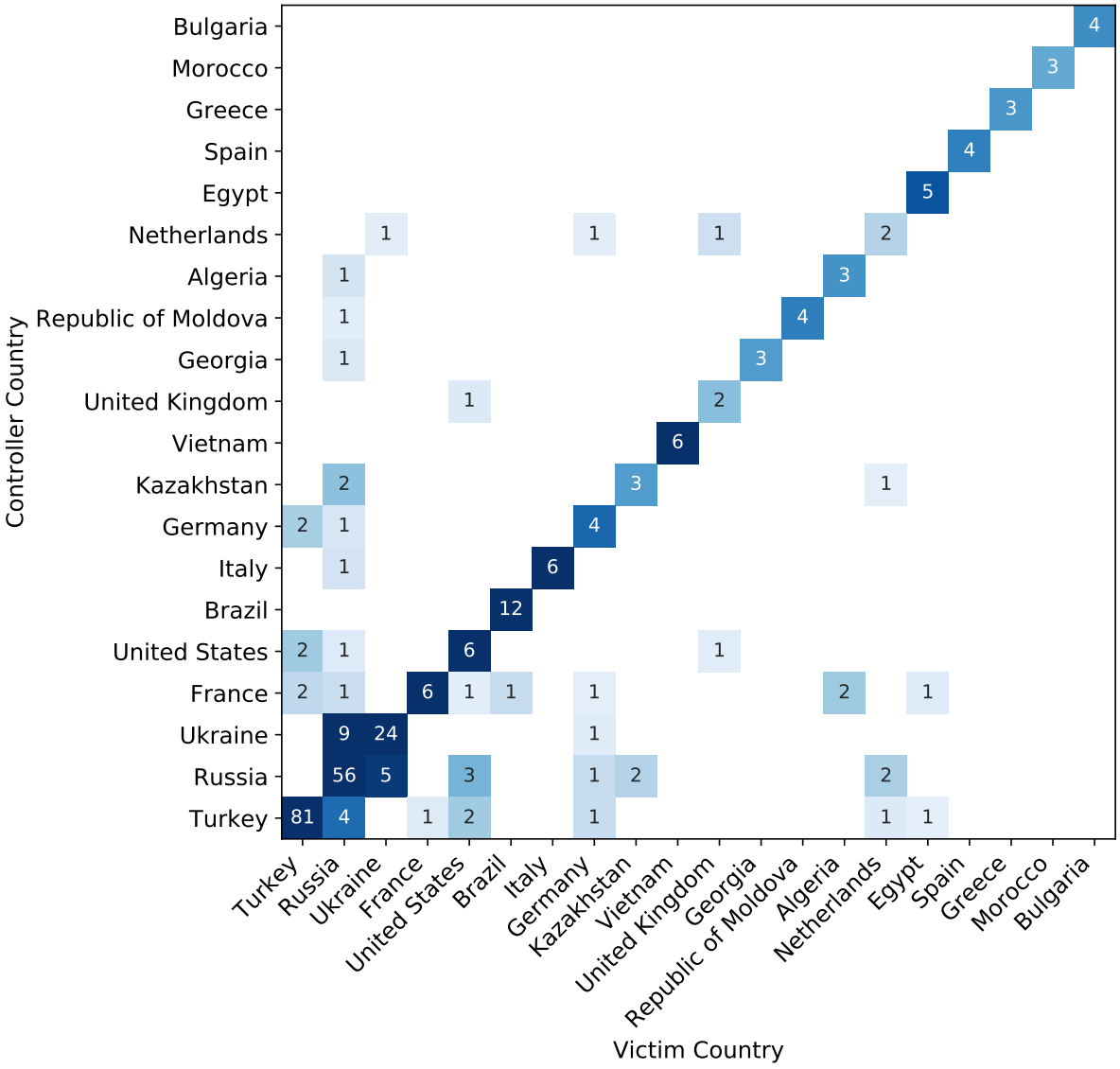


Figure 7.5. Number of controllers for each combination of controller and victim country, with each controller contributing 1 count. Rows denote controller countries and columns victim countries, based on geo-located IP address (excluding VPN providers).

victim population. Viewed from the controller side, a full 70% of attackers are in the same country as the *majority* of their victims. Colocation of operator and victim *is* the norm for operators with fewer victims. Figure 7.5 shows the same data as Figure 7.4, but normalized by the total number of victims a controller has. Each controller contributed 1 count distributed equally among its victims. Thus, we find in our data set evidence of two categories of DarkComet operator; the prolific spreader with many victims across the world, and the typical operator with one or a handful of victims, most of whom are geographically co-located.

7.4 Discussion

Throughout our experiments, we found there to be a propensity for operators and their victims to be colocated. We suspect this phenomenon is driven by a number of factors. RATs are infamous for their use in targeted, interpersonal attacks [5], for which they are well suited. We believe that RAT usage in targeting victims known to the attacker is a leading cause of this relationship. Further, for those operators attempting to spread stubs widely and infect many victims, shared language and culture likely narrow the population of individuals susceptible to the attacker's social engineering abilities to those from the same region or culture.

7.5 Acknowledgements

Chapter 7, in part, is a reprint of the material as it appears in Proceedings of the 38th IEEE Symposium on Security and Privacy 2017. Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blond, Damon McCoy, Kirill Levchenko, 2017. The dissertation author was the primary investigator and author of this paper.

Chapter 7, in part, is a reprint of the material as it appears in Proceedings of the 27th USENIX Security Symposium 2018. Mohammad Rezaeirad, Brown Farinholt, Hitesh Dharmdasani, Paul Pearce, Kirill Levchenko, Damon McCoy, 2018. The dissertation author was the primary investigator and author of the components of the paper presented in this dissertation.

Chapter 7, in part, has been submitted for publication of the material as it may appear in Proceedings of the 26th ACM Conference on Computer and Communications Security, 2019. Brown Farinholt, Mohammad Rezaeirad, Damon McCoy, Kirill Levchenko, 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 8

Conclusion

In this chapter, we summarize the dissertation, and then we discuss possible avenues for future work.

8.1 Dissertation Summary

In this disseration, we presented a system designed for measuring the ecosystem of remote access trojan malware from various vantage points. This system dramatically outperformed its closest competitor systems from industry, likely due to its ability to scour an extensible set of RAT threat intelligence sources. We then described the challenges inherent to measuring low-volume, low-visibility malware operations, and detailed three active measurement studies wherein we utilized a unique vantage point to overcome these challenges collect information about the participants in this ecosystem; particularly, RAT operators and their victims. We demonstrated the viability of using a system of honeypots to ellicit the behavior of DarkComet RAT operators, and assessed its potential as a defensive measure against RAT operators. Next, we uncovered the lingering threat of residual RAT infections by poaching and sinkholing dynamic DNS domain names formerly used by RAT campaigns. We showed that DarkComet and njRAT infections often persist for months after their attacker has ceased operation, with victims remaining vulnerable to control by new or returning operators. Then, we used a known, unique feature of the DarkComet controller software's command-and-control protocol to download

databases of its victims, developing data processing tools to assemble a high-fidelity data set of DarkComet victims and attackers. Using this data set, we quantified the harms done to victims of the DarkComet operators under observation. Finally, we provided a geographic analysis of the relationship between RAT operators and their victims based on information collected during each of the three active measurement studies, finding a distinct propensity for colocation. Ultimately, we demonstrated the challenges, viability, and value of collecting measurement data from low-volume malware operations.

8.2 Future Directions

Much of the work we presented in this dissertation was designed to help uncover and understand the RAT ecosystem — its scale, its attackers' motivations, the harms its victims suffer — through a series of measurement studies. Comparatively little of this work examined the actual viability of possible defenses against RAT operators, with the exception of a cursory investigation of using honeypots as tarpit defenses against manual attacks. Given the knowledge and data we have collected using our system, we believe a fruitful direction for future work is the evaluation of **deterrence efforts** against RAT operators.

A 2013 United Nations report on cybercriminality [83] indicated that most criminal users of malware, including remote access trojans, begin in adolescence and are indoctrinated in online forums. In 2017, the UK's National Crime Agency published a series of interviews [97] of individuals whom they arrested using Blackshades, a common RAT. These interviews confirmed the United Nations report's findings, and offered a number of particularly enlightening insights:

- Criminal users of Blackshades were largely unaware of the legal severity of their actions.
- They often were introduced to RATs through online forums dedicated to hacking; their peers on these forums dismissed any legal risk of using these tools.
- They almost unanimously indicated that, had they thought there was a chance of getting

caught, or had they known how severe the punishment could be, they would not have used RATs.

These findings lead us to believe that a study designed to deter RAT usage could find success, given the right methodology. This hypothesis resonated with work by Maimon *et al.* [82], who found that the simple act of showing a banner to SSH hijackers warning them of the legality of their actions led to a demonstrable drop in attacker activity. Our system, HAMELIN, could be modified to conduct two such studies: first, measuring the effect of displaying a warning banner on a RAT controller's GUI; and second, measuring the effect of showing warnings to honeypot intruders specifically crafted to trigger the responses found in the Blackshades arrest interviews.

Additionally, in our studies we found a relatively small number of services providing much of the supporting infrastructure to our operators. For instance, just two DDNS providers accounted for nearly all domain names found in DarkComet samples. IPjetable provided VPN service for almost half of all njRAT controllers. Many of the interviewed Blackshades users professed to learning their skills and acquiring their controller software on HackForums. This leads us to believe that there may be a small set of **principal enablers** of RAT users, the disruption or monitoring of which could significantly impact their operations. For instance, if No-IP and other DDNS providers were to implement and enforce more Draconian terms of service, a great many RAT operations could be impacted at once. A conclusive study enumerating the *choke points* in RAT operator communities could have a significant deterring impact on this ecosystem of abuse.

Bibliography

[1] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *ACM Internet Measurement Conference (IMC)*, 2006.

[2] Victor Manuel Alvarez. YARA: The pattern matching swiss knife for malware researchers (and everyone else). http://virustotal.github.io/yara/.

[3] AM523. How to create vpn for rat 2017. https://www.youtube.com/watch?v=0KQQ0pM3dDU.

[4] Nate Anderson. How an omniscient internet "sextortionist" ruined the lives of teen girls. http://arstechnica.com/tech-policy/2011/09/how-an-omniscient-internet-sextortionist-ruined-lives/, September 2011.

[5] Nate Anderson. How the fbi found miss teen usa's webcam spy. http://arstechnica.com/tech-policy/2013/09/miss-teen-usas-webcam-spy-called-himself-cutefuzzypuppy/, September 2013.

[6] Nate Anderson. Digital voyeur spied on women's webcams 5-12 hours a day. http://arstechnica.com/tech-policy/2015/10/digital-voyeur-spied-on-womens-webcams-5-12-hours-a-day/, October 2015.

[7] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the Mirai Botnet. In *USENIX Security Symposium (USENIX)*, 2017.

[8] Laura Aylward. Malware analysis - dark comet rat. http://www.contextis.com/resources/blog/malware-analysis-dark-comet-rat/, November 2011.

[9] U. Bayer, C. Kruegel, and E. Kirda. TTAnalyze: A tool for analyzing malware. In *European Institute for Computer Antivirus Research (EICAR)*, 2006.

[10] BBC. Miss teen usa hacker pleads guilty to 'sextortion' threats. http://www.bbc.com/news/technology-24929916, November 2013.

[11] Kevin Breen. dc-toolkit. https://github.com/kevthehermit/dc-toolkit.

[12] Kevin Breen. Look inside a dark comet campaign. Technical report, 2014.

[13] Kevin Breen. RAT Decoders. https://techanarchy.net/2014/04/rat-decoders/, April 2014.

[14] Kevin Breen. RATDecoders. https://github.com/kevthehermit/RATDecoders, April 2014.

[15] Kevin Breen. DarkComet - Hacking The Hacker. https://techanarchy.net/2015/11/darkcomet-hacking-the-hacker/, November 2015.

[16] Kevin Breen. DarkComet: From Defense To Offense - Identify your Attacker. In *Security BSides London*, 2015.

[17] BuddhaLabs. Packetstorm-exploits. https://github.com/BuddhaLabs/PacketStorm-Exploits/blob/master/1606-exploits/darkcomet-download.rb.txt, 2016.

[18] Juan Caballero, Heng Yin, Zhenkai Liang, and Dawn Song. Polyglot: Automatic Extraction of Protocol Message Format Using Dynamic Binary Analysis. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.

[19] Rahul Chatterjee, Periwinkle Doerfler, Hadas Orgad, Sam Havron, Jackeline Palmer, Diana Freed, Karen Levy, Nicola Dell, Damon McCoy, and Thomas Ristenpart. The spyware used in intimate partner violence. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 441–458. IEEE, 2018.

[20] Catalin Cimpanu. Ukrainian police arrest hacker who infected over 2,000 users with darkcomet rat. Technical report, ZDNet, 2018.

[21] Citizen Lab. Packrat: Seven Years of a South American Threat Actor. https://citizenlab.org/2015/12/packrat-report, December 2015.

[22] Nicolas Correia and Adrien Chevalier. zer0m0n driver for cuckoo sandbox. https://github.com/conix-security/zer0m0n/blob/master/bin/update_cuckoo.sh, 2015.

[23] Weidong Cui, Jayanthkumar Kannan, and Helen J. Wang. Discoverer: Automatic Protocol Reverse Engineering from Network Traces. In *USENIX Security Symposium (USENIX)*, 2007.

[24] Cynthia A. Brewer, Geography, Pennsylvania State University. ColorBrewer2. http://colorbrewer2.org/.

[25] David Dagon, Cliff Zou, and Wenke Lee. Modeling Botnet Propagation Using Time Zones. In *Networked and Distributed System Security Symposium (NDSS)*, 2006.

[26] DarkComet YARA rules. https://github.com/Yara-Rules/rules/blob/master/malware/RAT_DarkComet.yar.

[27] DEF CON 6. https://www.defcon.org/html/defcon-6/defcon-6.html, 2000.

[28] Shawn Denbow and Jesse Hertz. Pest control: taming the rats. Technical report, 2012.

[29] Department of Justice, U.S. Attorney's Office, Central District of California. Temecula student sentenced to federal prison in 'sextortion' case. https://www.justice.gov/usao-cdca/pr/temecula-student-sentenced-federal-prison-sextortion-case, March 2014.

[30] Rachna Dhamija, J Doug Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590. ACM, 2006.

[31] Digital Citizens Alliance. Selling "slaving" - outing the principal enablers that profit from pushing malware and put your privacy at risk. https://media.gractions.com/314A5A5A9ABBBBC5E3BD824CF47C46EF4B9D3A76/07027202-8151-4903-9c40-b6a8503743aa.pdf, July 2015.

[32] D. Dittrich and E. Kenneally. The menlo report: Ethical principles guiding information and communication technology research, 2012.

[33] David Dittrich, Felix Leder, and Tillmann Werner. A case study in ethical decision making regarding remote mitigation of botnets. In *International Conference on Financial Cryptography and Data Security*, 2010.

[34] DtDNS. https://dtdns.com.

[35] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. A Search Engine Backed by Internet-Wide Scanning. In *ACM Conference on Computer and Communications Security (CCS)*, October 2015.

[36] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Zmap: The internet scanner. https://zmap.io/.

[37] Marc Eisenbarth and Jason Jones. BladeRunner: Adventures in Tracking Botnets. In *Botnet Fighting Conference (Botconf)*, 2013.

[38] Brandon Enright, Geoffrey M Voelker, Stefan Savage, Chris Kanich, and Kirill Levchenko. Storm: When researchers collide. *USENIX ;login:*, 2008.

[39] Shayan Eskandari, Andreas Leoutsarakos, Troy Mursch, and Jeremy Clark. A first look at browser-based cryptojacking. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 58–66. IEEE, 2018.

[40] Najla Etaher, George RS Weir, and Mamoun Alazab. From zeus to zitmo: Trends in banking malware. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 1386–1391. IEEE, 2015.

[41] EUROJUST. International operation hits blackshades users. 2014.

[42] Robert Falcone and Simon Conant. Projectm: Link found between pakistani actor and operation transparent tribe. http://researchcenter.paloaltonetworks.com/2016/03/unit42-projectm-link-found-between-pakistani-actor-and-operation-transparent-tribe/, 2016.

[43] Cyrus Farivar. Sextortionist who hacked Miss Teen USA's computer sentenced to 18 months. http://arstechnica.com/tech-policy/2014/03/sextortionist-who-hacked-miss-teen-usas-computer-sentenced-to-18-months, March 2014.

[44] Farsight Security. https://farsightsecurity.com/.

[45] FBI. International blackshades malware takedown: Coordinated law enforcement actions announced. *FBI News*, 2014. https://www.fbi.gov/news/stories/international-blackshades-malware-takedown-1.

[46] Fidelis. Fidelis Threat Advisory 1009: "njRAT" Uncovered, June 2013.

[47] Fidelis Cybersecurity. Looking at the sky for a darkcomet. https://www.fidelissecurity.com/sites/default/files/FTA_1018_looking_at_the_sky_for_a_dark_comet.pdf, August 2015.

[48] Eva Galperin and Morgan Marquis-Boiremay. Fake skype encryption tool targeted at syrian activists promises security, delivers spyware. *Electronic Frontier Foundation*, 2012. https://www.eff.org/deeplinks/2012/05/fake-skype-encryption-tool-targeted-syrian-activists-promises-security-delivers.

[49] Dan Goodin. Teamviewer users are being hacked in bulk, and we still don't know how. https://arstechnica.com/security/2016/06/teamviewer-users-are-being-hacked-in-bulk-and-we-still-dont-know-how/, June 2016.

[50] Google. Google translation api. https://cloud.google.com/translate/v2/detecting-language-with-rest.

[51] Google. Virustotal intelligence. https://www.virustotal.com/intelligence.

[52] Waylon Grange. Digital vengeance: Exploiting the most notorious c&c toolkits. In *Black Hat USA*, 2017.

[53] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-peer botnets: Overview and case study. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)*, 2007.

[54] Claudio Guarnieri, Alessandro Tanasi, Jurriaan Bremer, and Mark Schloesser. Cuckoo sandbox. https://cuckoosandbox.org/, 2010.

[55] Levi Gundert. Proactive threat identification neutralizes remote access trojan efficacy. Technical report, Recorded Future, 2015.

[56] HackForums. https://hackforums.net/.

[57] Seth Hardy, Masashi Crete-Nishihata, Katherine Kleemola, Adam Senft, Byron Sonne, Greg Wiseman, and Phillipa Gill. Targeted threat index: Characterizing and quantifying politically-motivated targeted malware. In *USENIX Security Symposium*. USENIX, August 2014.

[58] Cormac Herley and Dinei Florêncio. A profitless endeavor: phishing as tragedy of the commons. In *Proceedings of the 2008 New Security Paradigms Workshop*, pages 59–70. ACM, 2009.

[59] Jesse Hertz, Shawn Denbow, and Jos Wetzels. Darkcomet server 3.2 remote file download. Technical report, 2016.

[60] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C Snoeren, and Damon McCoy. Tracking ransomware end-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 618–631. IEEE, 2018.

[61] Insikt Group. Talking to RATs - RAT IPs Dec2 Jan8 only. https://github.com/Insikt-Group/Research/blob/master/Talking%20to%20RATs/RAT_IPs_Dec2_Jan8_only, 2019.

[62] Insikt Group. Talking to rats: Assessing corporate risk by analyzing remote access trojan infections. Technical report, Recorded Future, 2019.

[63] IPjetable. https://ipjetable.net/.

[64] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M Voelker, Vern Paxson, and Stefan Savage. Spamalytics: An empirical analysis of spam marketing conversion. In *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2008.

[65] Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M Voelker, and Stefan Savage. The heisenbot uncertainty problem: Challenges in separating bots from chaff. In *USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)*, 2008.

[66] Mohammad Karami and Damon McCoy. Understanding the emerging threat of ddos-as-a-service. In *Presented as part of the 6th {USENIX} Workshop on Large-Scale Exploits and Emergent Threats*, 2013.

[67] Mohammad Karami, Youngsam Park, and Damon McCoy. Stress testing the booters: Understanding and undermining the business of ddos services. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1033–1043. International World Wide Web Conferences Steering Committee, 2016.

[68] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer, 2015.

[69] Kent Knudsen. Tracking the back orifice trojan on a university network. https://pen-testing.sans.org/resources/papers/gcih/tracking-orifice-trojan-university-network-101743, 2002.

[70] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[71] Radhesh Krishnan Konoth, Emanuele Vineti, Veelasha Moonsamy, Martina Lindorfer, Christopher Kruegel, Herbert Bos, and Giovanni Vigna. Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1714–1730, New York, NY, USA, 2018. ACM.

[72] Brian Krebs. 'luminositylink rat' author pleads guilty. *Krebs on Security*, 2018. https://krebsonsecurity.com/2018/07/luminositylink-rat-author-pleads-guilty/.

[73] Brian Krebs. Canadian police raid 'orcus rat' author. *Krebs on Security*, 2019. https://krebsonsecurity.com/2019/04/canadian-police-raid-orcus-rat-author/.

[74] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M Voelker, Vern Paxson, and Stefan Savage. Spamcraft: an inside look at spam campaign orchestration. In *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, pages 4–4, 2009.

[75] Christian Kreibich, Nicholas Weaver, Chris Kanich, Weidong Cui, and Vern Paxson. Gq: Practical containment for measuring modern malware systems. In *ACM SIGCOMM Conference on Internet Measurement Conference (IMC)*. ACM, 2011.

[76] Adam Kujawa. You dirty rat! part 1 – darkcomet. https://blog.malwarebytes.org/threat-analysis/2012/06/you-dirty-rat-part-1-darkcomet/, June 2012.

[77] Stevens Le Blond, Cédric Gilbert, Utkarsh Upadhyay, Manuel Gomez Rodriguez, and David Choffness. A broad view of the ecosystem of socially engineered exploit documents. In *Network and Distributed System Security Symposium (NDSS)*, 2017.

[78] Stevens Le Blond, Adina Uritesc, Cédric Gilbert, Zheng Leong Chua, Prateek Saxena, and Engin Kirda. A look at targeted attacks through the lense of an NGO. In *USENIX Security Symposium*. USENIX, 2014.

[79] Jean-Pierre Lesueur. Darkcomet remote administration tool. http://darkcomet-rat.com/.

[80] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Márk Félegyházi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, et al. Click trajectories: End-to-end analysis of the spam value chain. In *2011 ieee symposium on security and privacy*, pages 431–446. IEEE, 2011.

[81] Rainer Link and David Sancho. Lessons learned while sinkholing botnets - not as easy as it looks! In *Virus Bulletin International Conference,(Barcelona)*, 2001.

[82] David Maimon, Mariel Alper, Bertrand Sobesto, and Michel Cukier. Restrictive deterrent effects of a warning banner in an attacked computer system. *Criminology*, 52(1):33–59, 2014.

[83] Steven Malby, Robyn Mace, Anika Holterhof, Cameron Brown, Stefan Kascherus, and Eva Ignatuschtschenko. United Nations Office on Drugs and Crime: Comprehensive study on cybercrime. https://www.unodc.org/documents/commissions/CCPCJ/CCPCJ_Sessions/CCPCJ_22/_E-CN15-2013-CRP05/Comprehensive_study_on_cybercrime.pdf, 2013.

[84] MalwareConfig. https://malwareconfig.com.

[85] William R Marczak, John Scott-Railton, Morgan Marquis-Boire, and Vern Paxson. When governments hack opponents: A look at actors and technology. In *USENIX Security Symposium*. USENIX, 2014.

[86] Morgan Marquis-Boire and Eva Galperin. A brief history of governments hacking human rights organizations. https://www.amnesty.org/en/latest/campaigns/2016/01/brief-history-of-government-hacking-human-rights-organizations, January 2016.

[87] John Matherly. Shodan - Malware Hunter. https://malware-hunter.shodan.io/.

[88] John Matherly. Shodan - the search engine for the internet of things. https://www.shodan.io/.

[89] MaxMind. Geoip2 precision insights service. https://www.maxmind.com/en/geoip2-precision-insights, 2012.

[90] MaxMind. Geolite legacy downloadable databases. https://dev.maxmind.com/geoip/legacy/geolite/, 2012.

[91] MaxMind. Geolite2 downloadable databases. https://dev.maxmind.com/geoip/geoip2/geolite2/, 2012.

[92] Robert McMillan. How the boy next door accidentally built a syrian spy tool. *Wired*, 2012. https://www.wired.com/2012/07/dark-comet-syrian-spy-tool/.

[93] Microsoft TechNet. Specifying hardware ids for a computer. https://technet.microsoft.com/en-us/ff552325(v=vs.96).

[94] Steve Miller. Absolutely positively not 'hacking back' with pcap, 2019.

[95] MPRESS. https://autohotkey.com/mpress/mpress_web.htm, 1997.

[96] Yacin Nadji, Manos Antonakakis, Roberto Perdisci, David Dagon, and Wenke Lee. Beheading hydras: performing effective botnet takedowns. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[97] National Cyber Crime Unit / Prevent Team. Pathways into cyber crime. http://www.nationalcrimeagency.gov.uk/publications/791-pathways-into-cyber-crime/, 2017.

[98] NetBus. https://www.f-secure.com/v-descs/netbus.shtml.

[99] No-IP. https://www.noip.com/.

[100] Markus Oberhumer, Laszlo Molnar, and John Reiser. Upx: Ultimate packer for executables. http://upx.sourceforge.net/, 1996.

[101] Office of the Secretary, Department of Health, Education, and Welfare. The Belmont Report: Ethical principles and guidelines for the protection of human subjects of research, 1979.

[102] Micke Ölander. Offer för porrkupp. https://www.expressen.se/nyheter/offer-for-porrkupp/, 2004.

[103] Victor Oppleman. Network Defense Applications using IP Sinkholes. hakin9.org.

[104] Stig Øyvann. Rat trap: Norway police nab five in remote-access trojan europol swoop. Technical report, ZDNet, 2018.

[105] PasteHunter: Scanning pastebin with yara rules. https://github.com/kevthehermit/PasteHunter.

[106] Paul Pearce, Vacha Dave, Chris Grier, Kirill Levchenko, Saikat Guha, Damon McCoy, Vern Paxson, Stefan Savage, and Geoffrey M. Voelker. Characterizing large-scale click fraud in zeroaccess. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 141–152, New York, NY, USA, 2014. ACM.

[107] Daniel Plohmann and Elmar Gerhards-Padilla. Case study of the miner botnet. In *2012 4th International Conference on Cyber Conflict (CYCON 2012)*, pages 1–16. IEEE, 2012.

[108] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. A multi-perspective analysis of the storm (peacomm) worm. Technical report, Computer Science Laboratory, SRI International, 2007.

[109] Kevin Poulsen. Fbi arrests hacker who hacked no one. 2017.

[110] Niels Provos. A virtual honeypot framework. In *USENIX Security Symposium*. USENIX, 2004.

[111] QuasarRAT: Remote Administration Tool for Windows. https://github.com/quasar/QuasarRAT.

[112] Quequero. Darkcomet analysis – understanding the trojan used in syrian uprising. http://resources.infosecinstitute.com/darkcomet-analysis-syria/, March 2012.

[113] Babak Rahbarinia, Roberto Perdisci, Manos Antonakakis, and David Dagon. SinkMiner: Mining Botnet Sinkholes for Fun and Profit. In *USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)*, 2013.

[114] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. My Botnet is Bigger Than Yours (Maybe, Better Than Yours): Why Size Estimates Remain Challenging. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)*, 2007.

[115] Anirudh Ramachandran, Nick Feamster, and David Dagon. Revealing Botnet Membership Using DNSBL Counter-intelligence. In *Steps to Reducing Unwanted Traffic on the Internet - Volume 2*, 2006.

[116] David Reguera Garcia. Anti cuckoo. https://github.com/David-Reguera-Garcia-Dreg/anticuckoo, 2015.

[117] Relakks VPN. https://www.relakks.com/.

[118] Symantec Security Response. W32.shadesrat (blackshades) author arrested? Technical report, Symantec, 2012.

[119] Symantec Security Response. Blackshades rat usage on the rise despite author's alleged arrest. Technical report, Symantec, 2013.

[120] Symantec Security Response. Blackshades - coordinated takedown leads to multiple arrests. Technical report, Symantec, 2014.

[121] Mohammad Rezaeirad, Brown Farinholt, Hitesh Dharmdasani, Paul Pearce, Kirill Levchenko, and Damon McCoy. Schrödingers {RAT}: Profiling the stakeholders in the remote access trojan ecosystem. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1043–1060, 2018.

[122] Matt Richtel. Hacker group says program can exploit microsoft security hole. https://archive.nytimes.com/www.nytimes.com/library/tech/98/08/cyber/articles/04hacker.html, 1998.

[123] RiskIQ PassiveTotal. https://www.riskiq.com/.

[124] Neil C Rowe. Measuring the effectiveness of honeypot counter-counterdeception. In *Hawaii International Conference on System Sciences (HICSS)*, volume 6. IEEE, 2006.

[125] Jukka Ruohonen, Sami Hyrynsalmi, Igor Mishkovski, Tuomas Aura, Ville Leppänen, et al. The black mark beside my name server: Exploring the importance of name server ip addresses in malware dns graphs. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 264–269. IEEE, 2016.

[126] Dawn Song, David Brumley, Heng Yin, Juan Caballero, Ivan Jager, Min Gyung Kang, Zhenkai Liang, James Newsome, Pongsin Poosankam, and Prateek Saxena. BitBlaze: A New Approach to Computer Security via Binary Analysis. In *International Conference on Information Systems Security*, 2008.

[127] SSLBL: SSL blacklist. https://sslbl.abuse.ch/.

[128] Kevin Stevens and Nart Villeneuve. Darkcomet surfaced in the targeted attacks in syrian conflict. *Trend Micro: TrendLabs Security Intelligence Blog*, 2012. https://blog.trendmicro.com/trendlabs-security-intelligence/darkcomet-surfaced-in-the-targeted-attacks-in-syrian-conflict/.

[129] Cliff Stoll. *The cuckoo's egg: tracking a spy through the maze of computer espionage*. Simon and Schuster, 2005.

[130] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[131] Symantec Security Response. Creepware - who's watching you? http://www.symantec.com/connect/blogs/creepware-who-s-watching-you, December 2013.

[132] Tutorials On Hack. Example: How To open port windows 10 Free vpn {Tutorials android hack}. https://www.youtube.com/watch?v=N1mhRUCuyF4.

[133] Veronica Valeros. A study of rats: Third timeline iteration. Technical report, Veronica Valeros' Blog, 2018.

[134] Nart Villeneuve and Mike Scott. Crimeware or apt malware: Fifty shades of grey, 2014.

[135] VMware. vsphere platform. https://www.vmware.com/products/vsphere.

[136] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. Snoeren, G. Voelker, and S. Savage. Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm. In *ACM Symposium on Operating System Principles (SOSP)*, 2005.

[137] Lanier Watkins, Christina Kawka, Cherita Corbett, and William H Robinson. Fighting banking botnets by exploiting inherent command and control vulnerabilities. In *2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE)*, pages 93–100. IEEE, 2014.

[138] Lanier Watkins, Kurt Silberberg, Jose Andre Morales, and William H Robinson. Using inherent command and control vulnerabilities to halt ddos attacks. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 3–10. IEEE, 2015.

[139] Webmonitor enterprise pc. https://revcode.se/purchase-webmonitor/.

[140] WikiLeaks. SpyFiles 4. https://wikileaks.org/spyfiles4, September 2014.

[141] Carsten Willems, Thorsten Holz, and Felix Freiling. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security and Privacy (S&P)*, March 2007.

[142] Curt Wilson. Exterminating the rat part i: Dissecting dark comet campaigns. https://www.arbornetworks.com/blog/asert/exterminating-the-rat-part-i-dissecting-dark-comet-campaigns/, July 2012.

[143] Akira Yokoyama, Kou Ishii, Rui Tanabe, Yinmin Papa, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, Daisuke Inoue, Michael Brengel, Michael Backes, and Christian Rossow. SandPrint: Fingerprinting Malware Sandboxes to Provide Intelligence for Sandbox Evasion. In *Research in Attacks, Intrusions, and Defenses*, 2016.

[144] Andra Zaharia. Security alert: Infamous darkcomet rat used in spear phishing campaigns. Technical report, Heimdal Security, 2015.

[145] Lenny Zeltser. The evolution of malicious agents. https://zeltser.com/media/docs/malicious-agents.pdf, 2000.

[146] Guodong Zhao, Ke Xu, Lei Xu, and Bo Wu. Detecting apt malware infections based on malicious dns and traffic analysis. *IEEE access*, 3:1132–1142, 2015.