The EarlyBird System for Real-time Detection of Unknown Worms

Sumeet Singh, Cristian Estan, George Varghese, Stefan Savage
University of California, San Diego
{susingh,cestan,varghese,savage}@cs.ucsd.edu

***Abstract*—Network worms are a major threat to the security of today's Internet-connected hosts and networks. The combination of unmitigated connectivity and widespread software homogeneity allows worms to exploit tremendous parallelism in propagation. Modern worms spread so quickly that no human-mediated reaction to the outbreak of a new worm can hope to prevent a widespread epidemic. In this paper we propose an automated method for detecting new worms based on traffic characteristics common to most of them: highly repetitive packet content, an increasing population of sources generating infections and an increasing number of destinations being targeted. Our method generates content signatures for the worm without any human intervention. Preliminary results on a small network show promising results: we have identified three confirmed worms with a low percentage of false positives. This gives us reason to believe that our method could form the core of an effective network-level worm detection and countermeasure system capable of substantially slowing down the spread of new worms.**

I. Introduction

Current practice for worm response is *retroactive* and *manual*: only after a new worm is first detected and analyzed by a human who identifies a signature can the containment process be initiated. This approach is exemplified by the Code Red and Slammer outbreaks for which it took hours to days of human effort to identify the worms, followed by containment strategies in the form of turning off ports, applying patches, and doing signature based filtering in routers and intrusion detection systems [1], [2].

The difficulties with this current approach are:

1, Slow Response There is a proverb that talks about locking the stable door after the horse has escaped. Current practice fits this scenario because by the time the worm containment strategies are initiated, the worm has already infected much of the network. In fact, Moore et al. show that detection and containment must be initiated within minutes or seconds to prevent wide-spread infection in a 24 hour period [3].

2, Constant Effort Every new worm requires a major amount of human work to identify, post advisories, and finally take action to contain the worm. Unfortunately, all evidence seems to indicate that there is no shortage of new exploits and moreover simple binary modifications of existing attacks, can evade the signature-based detection employed in modern network intrusion detection systems (e.g., such as Snort).

There is a pressing need for a new worm detection and containment strategy that is real-time (and hence can contain the worm before it can infect a significant fraction of the network) and is able to deal with new worms with a minimum of human intervention (some human intervention is probably unavoidable to at least catalog detected worms, do forensics, and fine-tune automatic mechanisms).

More precisely, we list the goals of a automatic worm detection and containment system below (roughly in order of importance):

- *Real-time Detection:* The detection system should detect the worm in seconds, and potentially initiate some containment strategy in a similar time frame. Clearly, the reason for this is to prevent widespread infection before it is too late [3].
- *Content Agnosticism:* The detection system should not rely on external, manually supplied input of worm signatures. Instead, the system should *automatically* extract worm signatures even for new worms that may arise in the future. The reason for this is to reduce the constant human and other overhead involved in detecting a new worm.
- *Versatile Deployment:* The generic method should be deployable (with some modifications) at any point in the network including routers (edge and core), Intrusion Detection systems running on local area networks and even on end systems themselves. Wide deployment is necessary to ensure effective action against the spreading of a worm.
- *Scalability:* To support versatile deployment, we believe that worm detection and containment must be implementable in high-speed network devices – such as core routers. Ideally, a worm detection and containment method should be scalable: it should use moderate amounts of memory and in particular avoid any dependence on per-flow state. Moreover, any per-packet processing should be sufficiently small and parallizable to allow line-rate implementations at 1 and 10Gbps.

The following secondary goals may also be desirable:

- *Resilience to Simple Worm changes:* We can expect no system to provide absolute safety, but one should be able to argue that the worm detection strategy is resilient to simple countermeasures adopted by worm authors.
- *Ability to Handle Asymmetric Routing:* While in many places (e.g., end systems) one can observe both directions of a traffic flow, this is not the case at most core routers due to asymmetric routing. (50% of Internet routes in 1999 were asymmetrical.) Since it is be desirable to have a system that could be deployed at core routers we prefer methods that work even if they can see only one direction of certain flows.
- *No use of active probing:* Ideally, the containment system

should passively monitor the traffic (at least for detection purposes) as opposed to doing active probing of addresses. However, a small amount of traffic between cooperating agents of the IDS may be reasonable.

The rest of this paper describes a method that attempts to meet these goals, and a preliminary implementation of what we call the EarlyBird system. In 2 we describe the abstract characteristics of a worm our method relies on. In 3 we describe our generic method for detecting unknown worms that produces signatures, and 4 briefly describes possible actions the system might take in response. In 5, we discuss countermeasures that a worm author can riposte with. 6 describes the current EarlyBird prototype, and 7 describes the results of a preliminary evaluation. Finally in 8 we discuss related work and in 9 we conclude with future directions.

II. Abstract Characterization of a Worm

For the purposes of this paper, we define a worm based on two abstract features common to all current worms, from Code Red (multi-packet, TCP-based, targets widely-used HTTP port) to MS SQL Slammer (single-packet payload, UDP-based, targets rarely-used MS SQL port).[1]

F1, Substantial Volume of Identical Traffic: These worms have the property that at least at an intermediate stage (after an initial priming period but before full infection) the volume of traffic (aggregated across all sources and destinations) generated by attempts to infect further victims is a noticeable fraction of the network bandwidth.

F2, Rising Infection Levels: The number of sources and destinations involved steadily increases.

In addition, current worms also have the following property that can provide, along with the first two features, a stronger indication of guilt.

F3, Random Probing: An infected source spreads infection by attempting to communicate to random IP addresses at a fixed port to probe for vulnerable services, thereby often contacting unused portions of the address space.

This characteristic is commonly used in today's worm detection tools and analysis [1], [2]. Unfortunately, it seems possible for worm writers to modify their propagation strategies to use pre-generated hit lists [4] instead of random probing or permutation scanning [4] which contact unused addresses. Thus we use feature *F3* only as an additional sign of guilt.

Feature *F3* is also useful for distinguishing a worm from more benign traffic patterns such as Flash Crowds, Spam, and popular content in a P2P network that may exhibit features *F1* (high volume of repetitive content) and *F2* and (increasing number of sources and destinations). Unfortunately, if EarlyBird is to deal with simple alternative spreading strategies it may be best to use *F3* only to prioritize the list of suspicious traffic patterns. One can argue that these other "false positives" are indeed interesting traffic patterns that network operators would like to understand; thus flagging such patterns based on *F1* and *F2* only may provide benefits beyond the control of worm epidemics.

[1] We recognize that this characterization is vulnerable to opportunistic encryption and dynamic polymorphism, but we do not address those issues here.

III. A Method for Detecting Unknown Worms

We outline below our basic strategy that automatically detects each of these abstract features of worms with low memory, small amounts of processing, even at vantage points that see only one direction of certain flows, and without using active probing. The mechanisms[2] we use to detect each of these features are:

M1, Identifying large flows in real time with small amounts of memory : [5], [6] describe mechanisms to identify flows (aggregates) with large traffic volumes for any definition of a flow (e.g., source address, destination network). By modifying this definition to use the content of a packet (or more efficiently, a hash of the content) as a flow identifier, we can identify in real-time and with low memory the packets that repeat very often. We show later how we can make this method more robust to simple countermeasures by worm authors by detecting not entire packets that repeat often but frequently occurring fixed length strings that occur in many packets. An even more specific idea (that distinguished worms from valid traffic such as peer-to peer) is to compute a hash based on the content as well as the destination port (that remains invariant for a worm). While simple, we believe this notion (of using a content signature as a flow identifier or key on which to maintain counters) is an important contribution.

M2, Counting the number of sources and destinations: [7], [8] describe mechanisms using simple bitmaps of small size to estimate the number of sources and destinations on a link with small amounts of memory and processing. These mechanisms can be used easily to count the number of sources, destinations or source-destination pairs corresponding to signatures that account for high traffic volumes identified by the previous mechanism. While one could simply count source-destination pairs for a popular piece of content with a single counter, we believe that maintaining separate counters is more useful because it can reveal an increase in both sources and destinations, which is likely for a worm but less likely for Spam or a Flash Crowd.

Additionally, we can watch for random or permutation scanning as follows using a well known technique:

M3, Determining scanning by counting the number of connections attempts to unused portions of the IP address : We can use a small variant of Moore's "network telescope" idea [9], [10] by paying special attention to a portion of the address space we know to be unused. Worms spreading through UDP packets might not need a reply from the victim thus the exploit itself can be in the first packet they send to an address[11]. Thus assigning a larger "guilt score" to popular content that is sent to the telescope addresses will help us catch such worms. The exploit of TCP based worms will not be sent to these addresses because a connection cannot be initiated without an answer from the victim. However we can "blacklist" IP sources that send many packets to the telescope address space and assign a larger score to popular content if sent by these sources. Note that reserving such a large telescope address space for every IDS is clearly impractical. However, the telescope space can be shared by the IDS system within a network.

[2] Each of these mechanisms needs to be tuned to handle some special cases, but we prefer to present the main idea untarnished with extraneous details.

We note again that each of the mechanisms *M1, M2* and *M3* can be implemented at high speeds with low amounts of memory. Essentially this is because *M1* only keeps state for popular content signatures, and *M2* and *M3* require only a small amount of state for each piece of popular content. Notice also that the base method makes no assumptions about the point of deployment, whether at the endpoint, edge, or core.

IV. Acting on the Anomalous Packets

While our current system only reports the anomalous packets, the final system will also take adequate action. Because the system cannot always tell with high certainty whether a given packet belongs to a worm or not, it is appropriate to have more than one type of countermeasure in order to limit the amount of collateral damage caused by the system. First of all when the system suspects the appearance of a new worm it alarms the network administrator and provides forensic data to let human experts analyze and classify it. For traffic that looks like a worm but there is still a chance of misclassification the system takes less drastic action such as rate limiting it. For traffic identified with high confidence as worm traffic the system resorts to more drastic measures such as dropping the packets or resetting the TCP connections.

V. Worm Countermeasures

At this point the serious security enthusiast might wonder about the countermeasures worm authors might use to evade detection.

C1, Adding Random Filler: The worm could be simply modified to add a random filler before and after the actual payload that exploits the vulnerability, thus making a simple hash of the entire packet payload useless.

C2, Syntactic Polymorphism: The worm could spread across multiple links and the exploit could be fragmented differently across different links in the network.

C3, Semantic Polymorphism: The worm writer could use binary rewriting and other devices to completely rewrite the exploit at each infection point.

C4, Slow Contagion: Staniford et al [4] describe a technique called contagion where a worm spreads very slowly initially foiling mechanism *M1*.

To handle the limited polymorphism exhibited by *C1* and *C2* we use sampled Rabin fingerprints over portions of a packet (as was first used by Manber in spotting similarities in files [12]) instead of simpler hashes (SHA, MD5, CRC32) over entire packet contents. This is equivalent to detecting not frequently occurring packet contents but fixed length strings that occur frequently. *C3* and *C4* are effective against our worm detection method (and many others) but forcing worm authors to resort to such complex techniques raises the bar, thus helping us in the constant arms race with worm authors.

VI. EarlyBird System Description

We have built the system based on the method described in section 3. Our system is hardened against countermeasures *C1* and *C2* by using Rabin fingerprints of selected portions of a packet. The system can be deployed at a vantage point in the network to scrutinize every packet that passes through it, and provide various levels of alerts to the user.

First, a 64-bit signature (or a representative set of signatures) is computed by using a combination of the contents of the packet payload and the destination port number for every packet passing through the vantage point the system is deployed at. Section 6.1 describes in detail the signature computation process.

If the signature is found to occur more than $occurance Rate$ (threshold) number of times, then we instantiate counters to count the number of distinct sources, distinct destinations, and distinct source-destination pairs for the signature under consideration. The parameter $occurance Rate$ can be tuned to limit the number of signatures the counters are maintained for at any point of time.

As each packet may generate multiple signatures, we compute a ratio $matchPct$ which counts the percentage of matching signatures (substrings we saw before) for the particular packet. The decision to flag a packet as anomalous takes into account the $matchPct$ and the values of the counters across all the matching signatures for the packet.

The higher the $matchPct$ and the counts for the number of sources and destinations, for the packet under consideration, the more likely it is that the payload of the packet is anomalous.

As a general rule, the system generates alerts when:

1) Packets with similar content are being sent to a number of hosts (destination IP addresses).
2) Packets with similar content are being sent from a large number of hosts (source IP addresses).
3) Packets with similar content are being sent from a number of hosts to a large number of hosts (source-destination IP address pairs).

The number of alerts generated by the system can be throttled by requiring for the $matchPct$ and the number of hosts (sources and destinations) involved to be above minimum threshold levels which can be tuned by the user. We have done preliminary experiments by setting the value of the thresholds statically. We are now in the process of developing heuristics to adaptively determine these thresholds, we however elide the discussion from this paper due to space limitations.

A. Generating the Content based Signatures

For the purpose of generating signatures from the packet payload, we use a method similar to the one proposed by Manber [12] for finding similar files in a large file system. We compute Rabin fingerprints [13] for all possible sub-strings of a certain length. As these fingerprints are polynomials they can be computed incrementally while retaining the property that two equal sub-strings will generate the same fingerprint, no matter where they are in the string. Note that while Rabin signatures at all offsets may appear to greatly increase the amount of state, the use of sample-and-hold [6] will result only in maintaining state for popular signatures.

Let us consider that we have a string represented by a sequence of bytes $t_1, t_2 \ldots t_n$. The Rabin fingerprint F_1 for a sequence of bytes $t_1, t_2, t_3, \ldots, t_\beta$ of length β is given by:

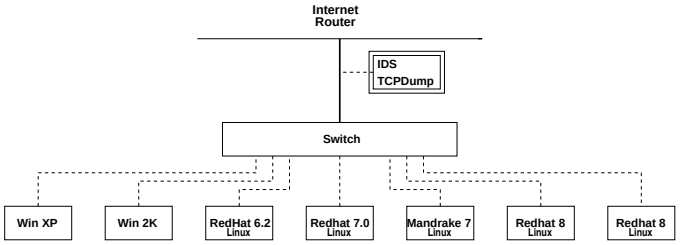

Fig. 1. Network configuration of the Local Area Network used for the evaluations in this paper

$F_1 = (t_1 \cdot p^{\beta-1} + t_2 \cdot p^{\beta-2} + \cdots + t_\beta)\ mod\ M$ where p and M are constants. If we now want to compute the next fingerprint F_2 for the sequence of bytes $t_2, t_3 \ldots, t_{\beta+1}$, then we need only to add the last coefficient and remove the first one: $F_2 = (p \cdot F_1 + t_{\beta+1} - t_1 \cdot p^{\beta-1})\ mod\ M$ For efficient computation we pre-compute a table of all possible values of $(t_i \cdot p^{\beta-1})$ for fast execution of the algorithm.

The length of the sub-string for which the fingerprints are computed has a direct impact on the frequency of occurrence of the fingerprint. The longer sub-strings we use the fewer repetitions we see. Based on a preliminary evaluation of the effectiveness of various lengths for the sub-string, we use 39 byte sub-strings for the experiments in this paper. Also we augment the incrementally computed Rabin fingerprints with the destination port number because the repetitive worm traffic always goes to the same ports while other repetitive content such as popular web pages or peer to peer traffic often goes to ports randomly chosen for each transfer.

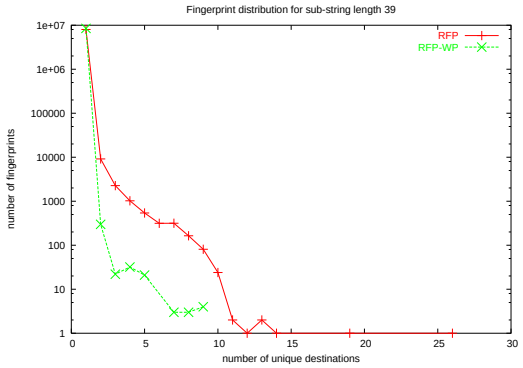

Fig. 2. Signature Distribution for signatures computed using Rabin Fingerprints of sub-string length $\beta = 39$ with and without the destination port number.

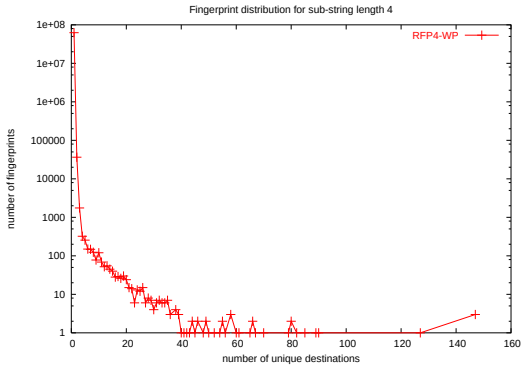

Fig. 3. Signature Distribution for signatures computed using Rabin Fingerprints of sub-string length $\beta = 4$ and the destination port number

VII. Preliminary Evaluation

A. Setup Description

The evaluation version of the system is positioned to look at all traffic flowing in and out of our Local Area Network (LAN) as shown in figure 1. The LAN is comprised of a total of 7 hosts (5 Linux, 1 Win XP, and 1 Win 2K). Because of the presence of the switch the IDS is unable to eavesdrop on any of the traffic between the hosts in the LAN. Alternatively the system can also analyze TCPDump traces.

In this paper we only present an evaluation of the mechanisms (discussed in section 3) and not a detailed evaluation of the complete system. We also did not implement sample-and-hold of Rabin signatures in the current prototype because memory was not an issue, and we also wanted to catch dormant worms (more likely to be seen in a random snapshot of traffic) whose traffic rates are too small to be caught by sample-and-hold.

B. Evaluation Results

The following evaluation results are based on a TCPDump trace collected at the vantage point shown in figure 1 over a 9 day period between May 2nd and May 10th 2003. The trace file has a total of 4 million packet (with payload) records.

Our primary goal in this evaluation was to detect anomalous packets (and the associated signatures) and understand why the false-positives and false-negatives were occurring (i.e. which applications or services if any, had an unusually high false reporting rate and what was causing it), and what impact the selection of the various threshold levels as well as the length of the sub-string (over which the Rabin fingerprints were computed) had on the false-positive and false-negative rates.

First consider the figures 2 and 3. In both figures the X-axis shows the number of unique destinations and the Y-axis shows the number of signatures, thus each point represents the total number of signatures (y-axis) destined for a number of destinations (x-axis).

Understanding the effects of using the port number in the signature: In Figure 2 the curve RFP represents the distribution of signatures based on the Rabin fingerprints and curve RFP-WP represents the distribution of the same signatures when augmented with the destination port number. As we need to maintain per signature counters (for signatures greater than threshold level) it is clear from this figure that utilizing the port number as a component of the signature significantly reduces the number of distinct signatures destined for more than one destination (thus we need to maintain counters for fewer signatures).

Understanding the effects of using small sub-strings for computing the signatures: Figure 3 shows the distribution of signatures based on Rabin fingerprints of sub-string length 4 augmented with the destination port number. It is clear that by reducing the length of the sub-string there is an order of magnitude increase in the total number of fingerprints.[3] Further in

[3]We emphasize again that the implementation does not use sample-and-hold

Source	Dest	SD Pairs	Prot/Port	Exploit (truncated)	Vulnerability	Name/Incident
45	5	51	TCP/80	GET /default.ida?XXXXXXX	IIS	CodeRed variant
4	3	4	TCP/80	GET / HTTP /1.1	pre-attack scan for ssl	Slapper
1	4	4	TCP/80	GET /scripts/.%252e/.%252e/	IIS	Unicode exploit
1	4	4	TCP/80	GET /scripts/..%c0%af../	IIS	Unicode exploit
1	4	4	TCP/80	GET /scripts/..%255c%255c../	IIS	Unicode exploit
1	9	9	TCP/443	-	OpenSSL-Linux	Slapper
1	3	3	TCP/80	GET http://www.s3.com HTTP/1.1	scan for open proxies	Not a Worm
498	4	742	TCP/139	-	NetBios	Out of band attack
17	3	23	TCP/445	-	NetBios	Out of band attack

TABLE I

Summary information of the packets marked as containing worm like traffic. Note, we do not only catch Worms, but also other intrusion attempts made which try to exploit vulnerabilities on the end hosts, many of these attempts follow the same characteristics as worms but for the fact that they are initiated by users.

comparison with figure 2 it is clear that reducing the length of the fingerprint increases the probability of finding a match for the small sub-string in a packet which has a direct impact on the resources needed by the system.

Recall that the IDS system does not require as seed (user input) any type of signatures (content based or port and byte count based) but instead the system should be able to automatically identify the anomalous signatures. We did not identify any new worms (as there were no new outbreaks during the evaluation period), but the results are nevertheless promising.

Our goal is to identify identical content that is prevalent on the network. Once identified (and the signatures gathered automatically) the system is required to determine the number of hosts the content has (and is) traversed. A content signature is determined to be anomalous if the count of source-destination pairs the content has traversed is above a threshold. For the results presented here we used the minimum threshold of 2 (i.e. the same piece of content is transferred between two or more hosts) for this counter. Table 1 summarizes some of the interesting findings.

All the exploits reported here (Code Red, Linux-Slapper etc.) have been out in the wild for some time. However, we were able to automatically identify their signatures, and verify that they were indeed worm attacks by comparing them with publicly available security advisories. The Unicode exploit is on the lines of the popular Nimda exploits. We were able to automatically capture 3 distinct attack patterns for the Unicode exploit.

The most unexpected thing we found was that packets originating from one of the host machines in our network showed behavior typical of the slapper worm (attempting to infect other hosts); this came as a complete surprise.

The Windows NetBios out of band attack (TCP/139) was generating an exceptionally high number of incidents. The number of incidents (in this trace and other traces) were high enough to warrant blocking all traffic destined for TCP Port 139 at the university gateway router.

We do not report the individual signatures or content strings that result in false positives but instead we list the following patterns which resulted in the packets being marked as anomalous. i) when the same piece of content is sent from one host to many different hosts; as in the case of mailing lists, or many clients being served the same image (hot object) from the same source (server), and ii) when the same piece of content is requested from many different clients (sources) from one server (destination).

In both these situations we would see the signatures repetitively along with a large number of source-destination pairs; however most of these false positive patterns reported by the system can be eliminated by simply requiring for the system to check that the same content propagate not just between multiple source-destination pairs, but also that there at least k distinct sources and k distinct destinations involved. By utilizing these additional counters we eliminated almost all the false positives, except for some small data-packets. In the case of the mailing-lists, the number of sources was one, while requests for the same piece of hot-content had one unique destination.

Besides these, other false positives the system could not automatically eliminate were, i) Requests for objects like ”robots.txt”, which are made from many different servers (by different bots, i.e. different user agents) to many different hosts. Note, requests for objects like index.html were almost always distinct for different hosts and are therefore not falsely reported as anomalous.ii) Single packet application identifier strings, for example in the case of ssh clients connecting to the server, the following string is passed during setup “SSH-1.99-3.1.1 SSH Secure Shell for Windows”. iii) Packets with small payload (3-4 bytes) belonging to applications like SSH and VNC. As the data payload is so small, creating signatures from the limited bytes increases the probability of finding the same signatures again and again.

All things considered, the particular signatures that generate false positives appear limited enough that a human can provide input to verify whether the signature under question is a piece of valid content or a worm (so that these signatures can be ignored in the future), just as spell-checkers use human input to avoid repeated flagging of valid words not present in a dictionary.

on the content signatures to reduce their size.

VIII. Related Work

Due to space limitations we only briefly discuss recent developments in worm detection mechanisms.

Zou et. al. [14] expand on the idea of Network Telescopes to monitor for port scan like behavior on ingress and egress routers as well as unused portions of the address space by setting up a black hole network. The major challenge they face is the prevalent noise on the unused address spaces.

Twycross and Williamson [15] take a host based approach and propose to rate limit the number of outgoing connections made by the host machine, as hosts infected with worms will be expected to connect to a larger than the usual number of hosts.

Snort [16] is publicly available open source software that attempts to prevent intrusions by filtering based on content signatures. However, *these signatures must be created by humans and entered into the system*. The system does not provide any kind of assistance in creation of the signatures. We note that the mechanisms discussed in this paper can be applied to Snort (and other existing tools) to further enhance their utility and effectiveness.

Spring and Wetherall [17] have previously applied Rabin Fingerprints [13] (in the context of web caching) to identify redundant network traffic in order to improve Web Performance. Duffield et al [18] sample packets based on a hash of the packet content for understanding routing packet patterns in a network.

IX. Conclusion and Future Directions

In this paper, we present an automated approach for real-time detection of unknown worms that uses two characteristics common to all worms — substantial volumes of identical traffic, and rising infection levels — to automatically derive signatures for new worms. More specifically, our method detects popular content, and measures other traffic characteristics (such as the number of distinct sources sending a particular piece of content and the number of destinations it is sent to) to ascertain whether the recurring content is a spreading worm. Given recently developed techniques for identifying popular flows and counting flows with small amounts of state, it appears that our method can be implemented at high speeds.

While we believe that EarlyBird can be a useful system in itself, we believe that the underlying method (maintaining state keyed by content signatures for signature extraction) raises a number of other interesting research questions. First, could adding content signatures as an extra field in traffic analysis tools such as FlowScan and AutoFocus yield extra insight for network managers? Second, can content signatures be used for better filtering at HoneyFarms, for speedier (maybe real-time) and efficient detection of intrusion attempts and their associated signatures? Third, the issue of content replication extends far beyond worms to other vexing questions such as Spam and Email worms. How should our ideas be modified for these very different contexts? These questions point to the more general notion of making content a first-class entity in traffic analysis.

Our preliminary results for EarlyBird on a small network show that our automated approach of identifying new worms is promising: using Rabin signatures, EarlyBird identified three confirmed worms with an encouragingly low percentage of false positives when configured with good parameters.

We plan to test EarlyBird at better vantage points such as the network edge and CAIDA monitors or even on the network of a Tier-1 ISP. These tests will allow us to further refine our method and better predict the effectiveness of a network-wide system based on it. A great challenge in testing the accuracy of worm detection methods is validating whether the pieces of content it flags are indeed worms. This requires examining content, an approach fraught with legal issues.

However, the ultimate question is whether EarlyBird will spot a genuinely new form of malicious traffic: or, more proverbially, will the EarlyBird indeed catch the worm?

X. Acknowledgements

This work was made possible by NSF Grant ANI-0137102 and the Sensilla project sponsored by NIST Grant 60NANB1D0118.

References

[1] D. Moore, C. Shannon, and J. Brown, "Code-red: A case study on the spread and victims of an internet worm," in *Proceedings of the Internet Measurement Workshop*, 2002.

[2] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The spread of the sapphire/slammer worm," July 2003.

[3] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Internet quarantine: Requirements for containing self-propagating code," in *Infocom*, Apr. 2003.

[4] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the internet in your spare time," in *Proceedings of the 11th USENIX Security Symposium*, 2002.

[5] P. B. Gibbons and Y. Matias, "New sampling-based summary statistics for improving approximate query answers," in *Proceedings of the ACM SIGMOD*, June 1998, pp. 331–342.

[6] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proceedings of the ACM SIGCOMM*, Aug. 2002.

[7] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Journal of Computer and System Sciences*, vol. 31, no. 2, pp. 182–209, Oct. 1985.

[8] C. Estan, G. Varghese, and M. Fisk, "Bitmap algorithms for counting active flows on high speed links," in *Internet Measurement Conference*, Oct. 2003.

[9] D. Moore, G. Voelker, and S. Savage, "Inferring internet denial-of-service activity," in *USENIX Security Symposium*, 2001.

[10] D. Moore, "Network telescopes: Observing small or distant security events," 2002.

[11] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The spread of the sapphire/slammer worm," Tech. Rep., Jan. 2003, `http://www.caida.org/ outreach/ papers/ 2003/ sapphire/ sapphire.html`.

[12] U. Manber, "Finding similar files in a large file system," in *Proceedings of the USENIX Winter 1994 Technical Conference*, San Fransisco, CA, USA, 17–21 1994, pp. 1–10.

[13] M. O. Rabin, "Fingerprinting by random polynomials," Center for Research in Computing Technology, Harvard University, Tech. Rep. 15-81, 1981.

[14] C. C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and early warning for internet worms," Department of Computer Science, Univ. of Massachusetts, Amherst, Tech. Rep. TR-CSE-03-01, 2003.

[15] J. Twycross and M. M. Williamson, "Implementing and testing a virus throttle," in *to apper in 12th USENIX Security Symposium*, Aug. 2003.

[16] "Snort: Open source network intrusion detection system," 2002. [Online]. Available: `www.snort.org`

[17] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in *ACM Sigcomm 2000*, Aug. 2000.

[18] N. Duffield and M. Grossglauster, "Trajectory sampling for direct traffic observation," in *ACM SIGCOMM*, 2000.