

Superworms and Cryptovirology: a Deadly Combination

Ivan Balepin

Department of Computer Science

University of California, Davis

ibalepin@ucdavis.edu

Abstract

Understanding the possible extent of the future attacks is the key to successfully protecting against them. Designers of protection mechanisms need to keep in mind the potential ferocity and sophistication of viruses that are just around the corner. That is why we think that the potential destructive capabilities of fast spreading worms like the Warhol worm, Flash worm and Curious Yellow need to be explored to the maximum extent possible. While re-visiting some techniques of viruses from the past, we can come across some that utilize cryptographic tools in their malicious activity. That alarming property, combined with the speed of the so-called “superworms”, is explored in the present work. Suggestions for countermeasures and future work are given. Keywords: computer viruses, worms, cryptography, cryptovirology

1. Introduction

The most distinctive and alarming trends in current computer attacks are high automation and speed, increasing sophistication of attack tools, vulnerability discovery rate that is hard to keep up with, increasing permeability of firewalls and highly asymmetric nature of threat [1]. Monitoring organizations name worms as one of the four most alarming types of today’s attacks.

The most notable incidents that caused such concern include the outbreaks of Code Red [10], Code Red II [11], Nimda [9], and, more recently, linux.slapper [12] worms. All four worms were noted for their extraordinary propagations speeds; however, damage-wise, they were rated as a low threat. Such a discrepancy between the levels of propagation techniques and destructive capabilities was immediately spotted, and several interesting works were produced ([2],[3],[4]) that (sometimes too emotionally) put the situation in perspective and explored the limits of destructive potential of fast-spreading, cooperating malicious entities.

However, this potential becomes even more overwhelming when one tries to combine the swiftness of the worms with the ferocity of some viruses from the past. Cryptography, as some point out [5], is usually thought of as a science that supplies us with tools to enforce integrity and confidentiality; however,

its undoubted strengths can be used to attack these same properties. Some of the studied viruses relied on cryptographic tools to cause damage that is quite hard to un-do.

This paper explores the combination of fast worms and cryptovirologic virus techniques. First, in Section 2.1, we give a survey of works describing the Warhol worm, Flash worm and Curious Yellow. Then, in Section 2.2 we describe Cryptovirology and potential damage that can be done by viruses with cryptographic capabilities. Section 3 is dedicated to further damage assessment and the countermeasures to the problem that we suggest. Finally, Section 4 is a summary of the ideas outlined in this paper.

2. Overview

2.1 Warhol Worm

The widely discussed [13] work on the Warhol worm begins by a quick analysis of the worms that plagued the internet in 2001. The famous Code Red virus was quite successful in its propagation. However, it performed random automatic scanning for the new victims, and utilized only one vulnerability in the Microsoft Internet Information Services (IIS). The worm did not use any local information to spread itself more efficiently. It did not have any communication or coordination capabilities.

Nonetheless, after a quick analysis, the authors come to a conclusion that the proportion of web servers infected grew exponentially with time. In the beginning, each infected server was able to find 1.8 other vulnerable servers per hour; in the final stages of the worm’s life, the rate was 0.7. Code Red turned itself off on July 19, 2001.

Damage-wise, Code Red had a distributed denial of service (DDOS) payload targeting the IP address of www.whitehouse.gov, and some web site defacement capabilities. Apart from that, it initiated an extraordinary amount of scanning traffic from the victim host. While somewhat bothersome, these actions cannot be considered a serious attack and indicate that the creator of the worm most likely pursued experimental goals.

A distinctive characteristic of Code Red is the very random nature of scanning it performed. According to the authors’ data, Code Red entities

scanned the same computers for the same vulnerabilities up to 500000 times per hour! The proportion of wasted scanning traffic becomes even more impressive if we consider the percentage of all possible IP addresses that actually map to active web servers running IIS with the targeted vulnerability. Such a random propagation strategy has several disadvantages: it wastes victim's resources, greatly reduces the propagation speed, reveals itself on the target system, and makes the worm world-famous in a matter of hours.

Code Red II targeted the same single IIS vulnerability as Code Red. As a scanning strategy improvement, it chose a random IP address from the victim's the class B address space with probability of 3/8, a random IP address from the victim's the class A address space with probability of 1/2, and an absolutely random IP address with a probability of 1/8. The authors note that such improved scanning strategy was successful, due to the fact that apparently hosts with similar vulnerabilities tend to be closer on the network, and also the quicker contamination of firewall-protected domains, once some Code Red II instance managed to get inside such network. The worm died by design on October 1, 2001.

Based on the new propagation strategy, we can conclude that the author of Code Red II, most likely, also pursued experimental goals, taking no time to address multiple vulnerabilities, or develop a more meaningful way to spread the virus.

The new virus had a potentially more damaging payload, which installed a root backdoor allowing unrestricted remote access to the infected host. However, Code Red II was quickly contained too, immediately revealing itself on the victim hosts.

The authors also argue that analysis of Code Red II behavior would be more involved than Code Red's, due to the fact that the two viruses overlapped and interfered with each other, and also to the local scanning strategy of the former.

Finally, the authors describe the Nimda worm, which contained a few obvious improvements. Nimda used five different ways to propagate itself, namely: an IIS vulnerability, bulk emails, open network shares, defaced web pages to infect visitors through their browsers and backdoors left by Code Red II and sadmind viruses. Such multi-vector approach also helped to penetrate the firewalls quicker, since most organizations leave incoming mail handling to the mail server or even users themselves. These improvements made Nimda another widely discussed worm; however, Nimda still appears to be a quick hack that lacks any solid design or purpose. The worm displayed the same characteristics; the authors cite their measurements on a Lawrence Berkeley National Laboratory computer that showed a peak hit rate of 140 Nimda HTTP connections per second. Despite the same inefficiency, system administrators report Nimda activity still, more than a year since the attack [13].

Nimda did not carry a communication or coordination payload. According to most sources ([9],[2]), the worm did not include any apparent destructive functions, apart from the ones that facilitated further propagation.

A large part of the paper is dedicated to considering possible worm improvements. The authors refer to the improved virus as a "Warhol worm". First, they look at so-called "hit-list scanning", which is collecting a list of vulnerable hosts prior to worm launch. After the pre-scanning stage, the worm would be unleashed on the hosts in the list. The authors argue that it took existing worm the longest to infect the first 10000 hosts and infection grew exponentially; therefore, a boost of 50000 would greatly speed up the propagation.

Permutation scanning is another improvement targeted at reducing the scanning overlap between worm entities. The new worm would generate an IP address space permutation using a 32-bit block cipher and a pre-selected key. It would encrypt an IP to get the corresponding permutation, and decrypt to get an IP. During the infection, it would work up the permutation starting from a random IP's hash, and restart at a random point in the hash every time it comes across an already infected system. Another improvement would be to stop completely after running into several infected hosts in row; that would indicate that the Internet is completely infected.

In a partitioned permutation scheme, worm instances get a hash range they are responsible for, and they halve their range every time they infect a new host, giving the other half to the new instance. When an instance completes its range scan, it restarts from a random point in the hash.

Topological scanning relies on the information and properties of the infected hosts, such as email addresses found on hard drives, a list of peers from a peer-to-peer networks a host might be participating in, etc. Some ([13]) note that a "spider" type of virus, which would operate similarly to web indexing and email collecting spiders, might also be efficient. That kind of a virus would be completely topology-dependent, traversing the network using popular protocols (HTTP, FTP, etc.) following the links it collects on its way. Such a possibility can also be considered in a separate work. Giving a Warhol worm spider-like capabilities appears to be another improvement in its propagation techniques.

The authors proceed to describe a so-called Flash worm. Such a worm, they argue, would require a somewhat more involved preparation stage, however, their simulations show that it would spread out in about 30 seconds as opposed to 15 minutes it takes the Warhol worm to subvert the Internet. In order to start a Flash worm, an individual (or a terrorist organization) would preferably have an access to an OC-12 type of network connection. In that case, according to the

authors' calculations, they would be able to pre-scan the whole web server space in a reasonable amount of time, and build a list of approximately 9 million web servers to start with. Such a list would cover the majority of the Internet, according to the recent Netcraft survey [14], and would require only 7.5 Mbytes to store in compressed form, according to authors' calculations. The first Flash worm instances would take the entire list, and would handle it similarly to the permutation scanning hash, halving the list for every new victim. Some redundancy would be required to prevent the first several instances from getting caught and not covering their part of the Internet.

Finally, the authors describe a stealthy slow-propagating contagion worm, which prefers concealing itself to fast propagation. Although it presents another interesting research topic, we would like to focus on the first two worms as the ones having a greater destructive potential.

Throughout the paper, the authors complement their arguments with descriptions of measurements and simulations they performed, and overall form an impression of a credible research work.

We observe that despite being a first serious analysis of worm design and suggesting a multitude of further research directions, the authors seldom mention a possibility of worm instances cooperating communicating with each other and the originator of the worm. We also note that the described worms do not rely on any cryptographic mechanisms except for trivial IP hashing for permutation scanning or, perhaps, encrypting itself to hide from static anti-virus scanners.

Nicholas Weaver published a follow-up work exploring how permutation scanning interacts with different ways the virus spreads itself ("multimode" or "multivector" worms). He also included a brief discussion of distributed control and update mechanisms; however, it still did not contain a solid coordination strategy.

2.2 Curious Yellow

The issues of worm communication and coordination were addressed in the design of a Curious Yellow worm [4].

Although the work is somewhat fictional in its nature and the author does not always provide a proof for his ideas, it presents another serious analysis of the worm potential and numerous directions for future research.

First, the author describes the benefits of worm coordination, which include the ability to easily assign an infection domain to each instance of the worm, easy control and update mechanisms, and less traffic which reveals the worm.

The difficulties of such coordination include problems with the truly gigantic scale of coordination, minimizing coordination costs, the need to take spoofed updates into account, etc. The author

concludes that some of these issues are similar to the ones observed in large scale peer-to-peer networks.

The author then proceeds to describing a peer-to-peer Chord strategy developed at MIT [15] in the context of a large-scale worm. The scheme, which is essentially a distributed hash table, is used to assign portions of task space to individual instances of the worm. In the improved version of Chord, Achord, all nodes act anonymously: a node cannot determine the identity of other nodes. The author argues that in a developing worm network, it would take a node $O(\log N)$ time to communicate with any other node, and a node would have to store information about $O(\log N)$ nodes for the most efficient communication. Therefore, in a network of 10 million nodes (which approximates the number of potential infected web servers) it would take correspondingly 23 node hops and 23 nodes to store. The instances use a hash (for example, SHA1) for identification, and once they find a new target, they pass it to the closest neighbor, or infect it themselves.

As an unsupported claim, the author argues that the Curious Yellow worm would form a fully connected network, and any messages such as code updates, etc., could be distributed network-wide in less than 15 seconds. Although that estimate remains to be verified, if we accept it, then we now have a malicious network that potentially can patch itself much quicker than a corresponding solution would be distributed network-wide.

In the section that explores potential uses for such a powerful network, the author notes that a DDOS against a few servers or disruption of the entire Internet would not utilize the worm to its full potential. Among the more creative uses the author names the possibility of defacing web pages uncontrollably, either at the host or at surrounding routers, isolating the unwanted servers, or the ones resisting the intrusion, by re-routing traffic around them, utilizing the CPU power of infected machines and stealing sensitive information.

A considerable part of the paper is dedicated to drawing an emotional picture of the subverted network. The author mostly focuses on fictional aspects of such an attack, as opposed to exploring the particular destructive directions an attacker might take.

The author briefly mentions that in order to safely use the updates, the worm instances would have to have the originator's public key, and authenticate each update to prevent unauthorized patches that might disrupt the operation of the worm.

A hypothetical concept of Curious Blue, a worm that cleans up after a Curious Yellow infection by using similar propagation strategies, or by exploiting a potential vulnerability in Curious Yellow itself, is also briefly mentioned. However, the author agrees with security experts on the fact that forcefully patching a large number of arbitrary servers is a very questionable action, both from the legal and technical point of view.

2.3 Cryptovirology

As an attempt to outline more concrete threat the rapidly propagating worms carry, we will briefly describe a 1996 study on cryptovirology [5]. It presents an interesting twist on cryptography, showing its possible malicious applications.

The authors start off by analyzing several viruses with cryptographic capabilities that have been observed during that time. LZR, AIDS Information Trojan and KOH were viruses briefly observed on some computers in 1994-1996 that exhibited some characteristics that the authors generalize to the main idea of the paper. The main goal is to make a victim host dependent upon the virus. They define a property of a high survivability of a virus, which can be summarized as “you kill the virus, you lose the data”. As a close approximation to a highly survivable virus, they suggest a scenario where a virus make the victim host depended upon the originator of the virus. Such virus would encrypt some sensitive data with some public key, but it would not contain a private key to decrypt it, therefore making any attempts to recover the data by analyzing its source code useless. The originator of such virus would hold the key to the data, therefore gaining control over the victim.

Cryptovirologic attacks exploit this dependency to the benefit of the virus originator. The authors consider two examples of such attacks, a reversible denial of service attack, and an information extortion attack.

In a reversible denial of service attack, the virus is equipped with a strong random number generator and a strong seeding procedure, and mounts an attack by generating a random session key K_s , and a random initialization vector IV . A simple cryptographic protocol forms the basis for the attack. The message $\{K_s, IV\}$ is encrypted with the public key of the virus' originator, resulting in cipher text C . Next, the virus encrypts the targeted data on the victim's system using K_s, IV , and a symmetric algorithm. After successful encryption, the virus overwrites the original data. Finally, the virus prompts the victim's operator to send cipher text C to the virus' originator, obtain a decrypted version of C , and regain access to their data by decrypting it with K_s , and IV . We note that this attack is more efficient with relatively small files, since encrypting a large file might reveal the virus and also it complicates the exchange process.

Another interesting kind of cryptovirologic attack we will describe here is the information extortion attack. This kind of attack is based on trading access to some target data in exchange for other data which is more valuable to the victim than the virus managed to get a hold of. The virus encrypts the sensitive data on victim's host as before, and then it calculates a checksum of a (possibly very large) file targeted by the attacker. The virus then prompts for the exchange of cipher text C from the previous attack that

now also contains the checksum, and the targeted data, for the key to the hijacked victim's data. Virus owner compares the checksum to the data received, and if really is the data desired, the key is released and the victim safely recovers the data.

The remainder of the work is dedicated to modifying a cryptovirus in such a way so it becomes highly survivable. The authors suggest distributing parts of the private key with virus instances, so that complete recovery is possible only if all victims cooperate, and explore the various arrangements and capabilities this approach carries. We leave out most of this discussion, since we feel that the arrangement in which the originator of the virus controls the data would be much more useful in the context of fast-spreading viruses.

The authors describe a rudimentary mechanism of supplying automatic feedback to the author of cryptoviruses. In order to steal the needed data without directly interacting with the victim, the author would have to intercept one of the victim's virus' offspring that would contain an encrypted copy of the data. However, they admit that such a scenario is highly unlikely and inefficient, especially considering the rates at which viruses propagated at the time the paper was written.

Some suggestions for countermeasures are actually included in the work. Traditional active virus detection and frequent backups are proposed. Another suggestion is strict control over cryptographic tools. Since including all necessary tools with the virus would make it large, inefficient and easy to detect, the virus actually has to rely on the ones built into the victim system, and the authors argue that by carefully controlling such accesses, the virus can be defeated. However, they do not supply any scenarios of such control; furthermore, they admit that this would be relatively hard to enforce.

3. Open Questions

As a relatively recent development, a linux.slapper worm appeared to be the first attempt to implement a coordinated malicious network [12]. The Linux-based worm created a peer-to-peer network of infected nodes. Communication was basic, allowing the network to learn its own topology, and launch DDOS attacks as a single unit when commanded from a single remote location. Slapper missed the ability to authenticate communication, and it was quickly contained, partly due to the prompt response by affected Red Hat Apache server administrators.

We note that in the Curious Yellow scheme, coordinated infection might not be very useful -- partitioned permutation seems a sufficient strategy to avoid overlapped scanning. However, coordinated control and update mechanisms, as we stated before,

open a multitude of opportunities for malicious activity.

3.1 Damage

Let us imagine a cryptovirologic superworm. It would combine the propagation speed of the Warhol worm, or a Flash worm, depending on the capabilities of the creator; communication capabilities of Curious Yellow, and cryptography-based malicious payload. Traditional active virus detection, proposed as one countermeasure, would be helpless against such worm, since the updates could be distributed much faster than the system administrators can clean their system. The virus stays afloat by constantly re-infecting the whole Internet using new zero-day vulnerabilities discovered by the worm owner. Despite the observation that as the worms get more complex, they become more vulnerable and easier to subvert themselves [13], a team of highly motivated experts with a solid destructive plan can easily produce a fault-free design and implementation of such a worm. Regular updates ensure invisibility even from the Curious Blue worm, which attempts to disinfect the victims. Worm instances observe schemes of access control to cryptographic tools on the victims' systems and trick them into allowing access to those tools. All attempts to analyze the traffic and track down the worm owner fail, since all traffic is minimized -- most of the times, it is not even apparent that a victim is infected; and finally, we can suggest periodic traffic exchanges to prevent traffic analysis. Even if some of these periodic exchange messages are observed, it would not be clear if the message, which is, of course, encrypted, actually contains some meaningful data (like an update), or simply is a placeholder message.

We note that the full scheme, in which all instances of the worm and its creator remain completely anonymous, and yet communication occurs on the regular basis without revealing the parties involved, is yet to be developed. However, it would be wise to assume that such a scheme can be implemented in the nearest future and prepare for the worst.

Frequent backups would be a somewhat effective measure against the cryptovirologic attacks of the worm; however, staying undetected for a long period of time and carefully analyzing the information flow on the victim system allows the worm to hijack the sensitive data between the backups. Therefore, we can conclude that none of the countermeasures presented in the covered works would be an adequate response to the worm.

Furthermore, we observe that this worm would threaten the existence of most of the digital payment schemes ([6],[7]), as well some certificate systems ([8]). E-cash and certificates can be instantly subverted, and either traded for real money, or same e-cash, or for some sensitive information.

3.2 Countermeasures

Apart from vague advice to perform the backups and patch the systems on the regular basis, there are a few things that we can suggest.

Specifically for certificates and e-cash schemes, we can suggest storing them in encrypted form, so that even in case of an infection, the worm would not be able to tell that encrypted data from regular files which present no interest to it. However, that appears to be a non-trivial implementation problem, since the victim needs to somehow obtain these, and the very request for them might lead the worm to the encrypted versions of certificates and e-cash. Even though they cannot be stolen in encrypted form, they still can be subverted once the worm finds out about the nature of that data.

One effective tool to combat the cryptovirologic superworm that we envision are automated response-enabled Intrusion Detection Systems (IDS). Although state-of-the-art is not at that point yet, a fruitful direction for research would be trying to develop coordinated response-enabled IDS's that quickly generate signatures of unknown attacks and communicate them to their peers before the worm. Specification-based IDS's that allow detection of unknown attack and automated response techniques are now being developed at several research sites, including the University of California, Davis Computer Security Lab.

4. Summary

By analyzing the successful worm implementations, we can conclude that only the lack of the clear damage strategy saved the Internet this time. The propagation strategies used in real attacks were not the most well-thought-out, either. Needless to say, a coordinated and well-planned attack can be much more devastating unless some countermeasures are taken.

In this work, we tried to combine the most notable recent works on the fast propagating malicious viruses with an interesting work on viruses with cryptographic capabilities to explore the extent of the possible damage that can be done by such a combination. We explored the questions that we felt these works left open. We also analyzed suggested countermeasures to such a worm, and proposed a few countermeasures of our own.

5. References

- [1] CERT Coordination Center, *2002 Overview of Attack Trends*, http://www.cert.org/archive/pdf/attack_trends.pdf, last accessed on December 4, 2002.

- [2] Staniford S., Paxson V., Weaver N., *How to Own the Internet in Your Spare Time*, Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, 2002.
- [3] Weaver, N., *Potential Strategies for High Speed Active Worms: A Worst Case Analysis*, <http://www.cs.berkeley.edu/~nweaver/worms.pdf>, last accessed on December 4, 2002.
- [4] Wiley, B., *Curious Yellow: The First Coordinated Worm Design*, http://blanu.net/curious_yellow.html, last accessed on December 4, 2002
- [5] Young A., Yung M., *Cryptovirology: Extortion-Based Security Threats and Countermeasures*, IEEE Symposium on Security and Privacy, Oakland, CA, 1996.
- [6] Chaum D., *Security without Identification: Card Computers to make Big Brother Obsolete*, *Communications of the ACM*, vol. 28 no. 10, October 1985 pp. 1030-1044.
- [7] Rivest R., Shamir A., *PayWord and MicroMint -- Two Simple Micropayment Schemes*, 4th Security Protocols International Workshop, Cambridge, UK, 1996.
- [8] International Telecommunications Union, *Recommendation X.509 – the Directory Authentication Framework*, 1993
- [9] CERT Coordination Center, *CERT® Advisory CA-2001-26 Nimda Worm*, <http://www.cert.org/advisories/CA-2001-26.html>, last accessed on December 4, 2002.
- [10] CERT Coordination Center, *CERT® Advisory CA-2001-19 "Code Red" Worm*, <http://www.cert.org/advisories/CA-2001-19.html>, last accessed on December 4, 2002
- [11] CERT Coordination Center, *CERT® Incident Note IN-2001-09*, http://www.cert.org/incident_notes/IN-2001-09.html, last accessed on December 4, 2002
- [12] CERT Coordination Center, *CERT® Advisory CA-2002-27 Apache/mod_ssl Worm*, <http://www.cert.org/advisories/CA-2002-27.html>, last accessed on December 4, 2002
- [13] Slashdot.org, *Malicious Distributed Computing*, <http://slashdot.org/article.pl?sid=02/10/25/1413220&mode=thread&tid=172>, last accessed on December 4, 2002
- [14] Netcraft, *Netcraft Web Server Survey*, <http://www.netcraft.com/survey>, last accessed on December 4, 2002
- [15] Dabek F., Brunskill E., Frans Kaashoek M., et. al. *Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service*, IEEE Eighth Workshop on Hot Topics in Operating Systems p. 81, Elmau, Germany, May 20 - 22, 2001