Cody Ickes, Ethan Letourneau

Dr. Jie Yang

CNT 5412 - Network Security

A Survey on Computer Worms: Their History and Methods of Operation, Detection, and Prevention

***Abstract* –** The aim of this research is to garner an in-depth understanding of the inner-workings of the category of malware known as computer worms. Additionally, we seek to understand the common techniques to slow the spread of and neutralize computer worms, as well as to uncover novel research into these matters. The methodology for this research was to take a survey of research papers, technical documents, and subject matter-oriented articles and compile a summation of the body of methodologies. The findings herein are that computer worms transfer over networks using variations of three key stages: target finding, propagation, and activation. The methods for detecting the presence of a computer worm on a network are broadly categorized as either anomaly or signature-based. For each category, there are a plethora of procedures. Finally, we deduced that the stoppage of worm spread is done by a combination of what we coin as education and gatekeeping. We conclude with a section reviewing novel research on the topics of modelling worm propagation and detection. We deduce that the research on the detection and mitigation of computer worms is ever-evolving in parallel with the ever-evolving nature of the malware.

Introduction

Network Security has had an exponentially increase in importance as our world has become more connected over the internet. With almost all sensitive information being exchanged over networks these days – from banking information to Social Security numbers – adversarial programs that attack over networks could be considered to be the greatest threat to Cybersecurity as we know it today. Computer worms, simply put, are pieces of malware that spread over a network without outside intervention, that is that they self-propagate. Such malware could be anything from ransomware to spyware, but regardless of the content it is inherently more dangerous because of its capability for autonomous spread. Thus, much research has been done in understanding the workings of how this distribution occurs, and what engineers can do to mitigate it.

We will explore a brief history of the evolution of the computer worm before surveying the different ways in which worms find victims, distribute themselves over different avenues, and activate. Furthermore, we will look into the practical methods and tools used by security experts to impede computer worms, as well as some academic research into novel methods of bettering these techniques of worm-stopping.

History of Worms

In the Beginning

Prior to its inception, the term "worm" was coined in John Brunner's book, *The Shockwave Rider*. Defined as bits being "added automatically as it works its way to places," and that "it can't be killed," and will continue to grow as it "will remain in store at some other station and the worm will automatically subdivide and send a duplicate head" (Schoch and Hupp). Shortly after this idea was penned, the first developments of what is known today as a "computer worm" began on the Arpanet – the early stages of the Internet. First was the program to transmit itself, this was known as the "Creeper." When executed, the program would search for another device, once found, its execution on the current computer would halt and it would transfer itself to the new system (Schoch and Hupp).

Transitioning from the "Creeper" to the first worm was a small gap to bridge – with similar functionality, it now needed to replicate and store a copy of itself before moving to another node. The first of its kind was an updated "Creeper" and its corresponding logging program, "Reaper." The new "Creeper" would traverse through the network while replicating itself while "Reaper" moved around searching and logging copies (Schoch and Hupp).

Unfortunately, the benign use of worms would quickly become malicious. In November 1988, the first worm was released on the Internet by a Cornell computer science graduate student named Robert Tappan Morris, this would later be called the Morris worm. On November 2, the worm had reached "at least one computer in 20," still propagating and finding new targets with every transmission (Orman). This malware grinded computer systems to a halt by spreading throughout the Internet using the Unix sendmail program while replicating itself with the C compiler. No one was certain of what other effects the virus had, but it was clear that service was being denied (Orman).

Major changes were made in response to the fallout of the Morris worm. Initially, computer science departments attempted to define acceptable usage, attempting to balance security with ease-of-use and simple access. Most notably, the Computer Emergency Response Team (CERT) was founded by DARPA to collect and distribute “security-vulnerability information.” The purpose of CERT served to inform users of security concerns and provide fixes for them. As one would expect, despite the formation of such a team, many still fail to make use of the information and, as such, leave themselves and others potentially exposed to similar attacks (Orman).

In the Modern Era

After the Morris worm, there was a period that such a malware was no longer common; however, in 1999, a new contender dubbed “Melissa” revived these types of attacks. Discovered on March 26, this worm exploited Microsoft Outlook and denied service in mail systems due to the infected emails it produced. Upon execution, the malware attempted to transmit to the first 50 addresses in the target’s Outlook address book, if this was not possible, it would instead store copies of itself to other documents and attempt to email itself with the data. This would ignite the rise of new worm developments for many years to come (Fosnock).

In later years, a new worm emerged that carried out an “unprecedented sophisticated attack that would have wide implications for future industrial systems” (Karnouskos). This worm was labeled Stuxnet and it made its mark in 2010. For the first time ever, a large scale attack took place on SCADA systems, commonly used for “specific industrial processes,” as put by Karnouskos. Given that the systems were all offline and had limited access, the point of intrusion would need to occur inside the network from a trusted source, an act considered nearly impossible. Dismantling the facade of security, this attack further exemplified the disastrous effects worms could have and how they change over time (Karnouskos).

Most recently, the ransomware labeled “WannaCry” appeared and revealed itself as a serious threat. Blocking access to information and systems, Mohurle and Patil state that this ransomware held “files or entire devices hostage using encryption until the victim pays a ransom in exchange for a decryption key.” This worm found its way inside of “many hospitals, companies, universities and government organizations,” forcing those in need of essential data to pay up or lose the information for good (Mohurle and Patil). Without the proper measures in place, all it took was a single user to be infected before propagating to other unsecure devices.

In modern times, worms take common viruses to a new level. Recent events have shown that the spread of malware is becoming more common over time. One single device is infected, then the number of impacted devices exponentially grows, the worm replicating and injecting itself to many new targets for every vector it acquires. The security world is seeing just how aggressive worms are becoming and it is most likely explained by the growing widespread usage of the Internet, the most common domain for worm propagation.

Used as a Tool for Good?

The dangerous consequences of worms have been thoroughly examined, but it is also important to cover the idea that worms can be beneficial. Simply put, worms are not dangerous by themselves, they are simply a tool that can be used for good or bad, it all depends on how it is defined to operate. One example of a “well-intended” worm is “Nachi.” In August 2003, a malicious worm called “Blaster” was released and propagated by exploiting a Windows Distributed Component Object Model (DCOM) Remote Procedure Call (RPC) vulnerability. Making use of the same vulnerability, Nachi would transmit itself and attempt to remove Blaster from infected devices while subsequently downloading a security patch from Microsoft to fix the RPC DCOM exploit (Chen).

While worms can be used with good intent, they are often used in malice. Some viruses exist that are meant to be useful in some way, but the truth in the matter is that these are still called “viruses” and are generally thought of as dangerous. Just to function, they tend to exploit a vulnerability, regardless of the intent. There can be specific cases in which worms benefit the user, but it is best to avoid getting any infection by keeping everything updated and secure.

Composition of Computer Worms

Finding Targets

Before carrying out an attack, computer worms need to know where to send themselves. There are many ways for this type of virus to determine valid targets, the main ones are scanning, use of target lists, and passive searching. To use target lists, there are three different generation methods: pre-generated, externally generated, and internal (Weaver, Paxson and Staniford). Once vulnerable devices are discovered, they can then be used with propagation schemes to seed the attack.

Scanning

Commonly used for internal networks, scanning probes for valid IP addresses to determine vulnerable points of attack (Boukerche and Zhang). Scanning is done in four possible ways: selective random scan, sequential scan, hit-list scan, and routable scan. Selective random scan is where the worm chooses a random address to attack, no logic is used in this and success is not guaranteed, making it one of the slower methods. Instead of a random scan, sequential scan searches for vulnerable devices in order rather than by chance, potentially increasing the hit rate. Next, moving away from choosing one address at a time, hit-list scan creates a list of possible targets and uses it to find vulnerable machines to attack. Lastly, routable scan searches a specific IP address space based on obtained routing information, this allows the worm to travel much more quickly and quietly as targets are easier to find with fewer failures (Pratama and Rafrastara).

In recent events, the worm “Hajime” makes use of random scanning by “repeatedly generating random IPv4 addresses” and attempting “to connect to them on port 23” and then trying to log in “by sequentially going through a table of username/password credential pairs” (Edwards and Profetis). Even more interesting, Hajime impacts IoT devices, a newer technology. As technology advances, the current state of connectivity and wireless networking creates easier to attack targets, as demonstrated by Hajime.

Pre-generated Target Lists

Prior to an attack, worms generate a list of targets that are kept with the worm itself. These are typically used in situations where specific addresses are to be attacked without generating significant traffic (Boukerche and Zhang).

Externally Generated Target Lists

Much like pre-generated target lists, this method of targeting stores the list outside of the worm. The data can either be kept as a separate entity that is sent with the worm or it can be stored on a server that the worm can later query during its propagation (Boukerche and Zhang). One considerable factor in this method is the increased traffic generated during an attack either through external requests or increased packets needed to operate.

Internal Target Lists

The most lightweight of target lists, internal target lists store the list locally on the attacked network(s) where all the traffic can be kept inside of the network with smaller worm sizes (Boukerche and Zhang).

Passive

Last of the targeting techniques, and perhaps one of the slowest, passive targeting is carried out by having the vulnerable devices access the worm itself (Boukerche and Zhang). Instead of the malware actively finding new machines, the users go to it by themselves, as such, it does not generate any suspicious traffic.

Propagation Schemes

Worms are unique in their characteristic property to spread to other devices on a network. Rather than being the target of a user-prompted attack, this malware can be transmitted to even the most security-savvy users. There is no single point of protection, instead, it requires security on the entire network as it can only be as strong as its weakest link; one compromised machine is all it takes to tear down a whole group of devices. While worms transmit themselves much like other viruses through communication channels such as email, propagation methods are also carried out through the network, including self-carried, second channel, and embedded schemes (Li, Salour and Su).

Self-Carried

Self-carried worms are transmitted by themselves in a packet, this can be done through UDP or TCP (Li, Salour and Su). Given their self-transmitting nature, these types of worms tend to also be self-activating – initiating its own execution rather than injecting code for another process to inadvertently run (Weaver, Paxson and Staniford).

Second Channel

While like self-carried propagation, rather than carrying all the information in a single transmission, a second channel worm transmits itself and downloads the payload from a previously infected device or via Internet (Li, Salour and Su). When the malware cannot send all of itself to another target, it sends only the "head" of the worm. The "body" is the payload, it contains the necessary code to carry out the attack. In this instance, the body is obtained through a second channel, hence the name "second channel propagation" (Boukerche and Zhang).

Embedded

Resembling a second channel worm, this modifies incoming traffic for the payload to be discretely delivered (Li, Salour and Su). The body can be hidden in a few different ways, one of which is appending itself or replacing a normal message, this takes control of expected traffic and is much more difficult to detect. Additionally, the entire worm can be hidden inside of a file such as a modified program or host file, and is typically the first point of entry, the weakest link (Li, Salour and Su).

The discussion of propagation methods is still relevant today, studies are done on worm propagation to create biological epidemiology models due to the striking similarities in how transmission rates work, notable in the midst of COVID-19 (Khanh). In wireless sensor networks, this factor has been frequently modeled using mathematical approaches to describe how a worm spreads from an infected node. The modeling by Khanh concludes that transmission is fundamentally the same in wireless networks, to implement proper protections requires the development of “effective firewall network systems” and that “further research is to be done on new antivirus programs and advanced network quarantine systems” using artificial intelligence. If anything, the transmission of worms in wireless networks becomes more widespread while using the same techniques.

In addition to wireless network transmission, research from Singh et al. has exposed the potential for worms to transmit through other wireless means such as Bluetooth. One example of a worm that uses Bluetooth to transmit is Cabir, repeatedly sending itself to other Bluetooth enabled devices (Singh, Awasthi and Singh). Furthermore, a variant of Cabir later appeared, also transmitting itself using MMS (Singh, Awasthi and Singh). This information is modeled using the same approach as Khanh, defining the different variables that affect rates of transmission, even being compared to epidemiological models. With how much more connected methods like Bluetooth, Wi-Fi, and MMS are compared to wired networks, there is an increase in the rate of transmission given the lack of proper detection and prevention methods.

Activation Methods

Once a worm has successfully infiltrated a target, it must now be activated to carry out its attack. Activation can be performed automatically or through external processes, including user-initiated actions such as running a compromised software. Because of this dependency, the rate of infection is largely determined by the method used (Weaver, Paxson and Staniford).

Human Activation

As suggested by the name, human activation requires the user to trigger the attack. A user running a compromised program is an example of such a method. Likely due to the necessity for user involvement, this is slowest activation method, potentially never initiating the virus. Getting users to run such a worm is typically done through social engineering techniques, using ordinary files such as email attachments, images, and other executables and files that can exploit bugs (Weaver, Paxson and Staniford).

Human-Activity Based Activation

In contrast to human activation, this method makes use of user behavior to begin the attack. Some examples of this include restarting the machine and logging into accounts. This is usually carried out from the worm modifying startup files so that when the device is restarted or a login occurs, the respective files are loaded, thereby indirectly executing the malware (Weaver, Paxson and Staniford).

Scheduled Process Activation

Faster than the previously discussed methods, scheduled process activation uses scheduled system processes to initiate the attack. One of the ways this is performed is through the operating system's automatic update program; with authentication vulnerabilities, the worm can use this to serve the malware to the updater and infect the system. In addition to auto-updaters, worms can also be activated through other exploitable schedule processes (Weaver, Paxson and Staniford).

Self Activation

The fastest type of worm, self activation is when the worm triggers its own payload. It often attaches itself to a running service and executes other commands to run the code. Most importantly, this activation scheme requires the running programs to have vulnerabilities (Weaver, Paxson and Staniford).

Following the breakdown of a worm's components, it is important to note that a worm may use any combination of the schemes discussed. The implementation of these methods is everchanging which means that worms may forever be a threat, however, there are prevention and detection techniques that can minimize the risks of such a virus.

Worm Detection Techniques

The first and arguably most vital step in combating computer worms is detecting one's existence on a system or network. The characteristics and propagation techniques of worms, as described in previous sections, provide key indications of their presences. Worm detection is done in two broad manners, being either *signature-based* or *anomaly-based* methods. Signature-based worm detection is done by comparing network traffic and/or behavior of a program to previously generated signatures of both which are known to be malicious. The anomaly-based counterpart is done by observing patterns in normal network traffic and program behavior and raising alarm when troubling deviation from the norms are found.

Each form of detection has its inherent strengths and weaknesses. By the very nature of it, signature-based worm detection would obviously not be effective in finding novel worms as it relies on known malicious elements. On the flip side, it is a near-perfect method if one needs to detect well-known worms on a network or system. Anomaly-based worm detection, on the other hand, is much better at detecting worms previously unseen, but is much more susceptible to giving false-positive alarms. Given these characteristics, a natural intuition is to use both a signature and anomaly-based detection system in tandem wherein the anomaly-based arm can detect new worms and write signatures for the signature-based part to use.

In the following subsections we will describe the behaviors that each type of detection identifies as well as how such identification is done.

Anomaly-Based Detection

As previously described, anomaly-based worm detection looks for deviations from normal network traffic and application behavior. The behaviors of a worm that are most apt to be found by anomaly-based detection include the target scanning, transference, and activation mechanism.

As previously described, worms will scan a network to look for victims in a variety of different ways – recalling a few, we looked at four different types, including random scanning and routable scanning, among the others. Using anomaly-based detection, a majority of the scanning mechanisms of common worms can be found. Some anomalies that can be found when a worm is scanning for targets include (a) some host trying to connect to an inordinately large volume of unique machines over TCP/IP, (b) an infected host failing a large volume of connection attempts, and (c) port destination-source correlation.

(a) *High Unique Connection Count*: Since the goal of most worms is to spread as quickly as possible, most of the scanning methods one uses will look for victim machines rapidly, thus each of the methods cause an infected host to connect to many unique different IP addresses in a short period of time. More technically, detection systems that look for this anomaly will track the number of TCP SYN packets being sent out from suspected infected systems – such packets are the connection-initiating standard under the TCP/IP protocol.

(b) *High Connection Failure Rate*: On the flip side, with many connection attempts will oftentimes result in many connection failures – a large amount of them in a certain period of time could indicate infection of a system. Again, on the technical side this would be indicated by a large amount of TCP RST packets or ICMP packet errors being received by a system. TCP RST packets indicate that a TCP connection attempt to a known real host on a port that is closed. ICMP “destination unreachable” packet errors are received by a host that tries and fails to connect to a port on a known system or tries to connect to a non-existent IP address.

(c) *Port Destination-Source Correlation*: A system described in (Gu, Sharif and Qin; Qin, Dagon and Gu). A host’s ports are monitored as such: if it starts to send packets to a certain port number on other machines after receiving one on the same port, the counter for said port is incremented with each sent packet. The idea is to see if the host has been infected by a worm operating on a certain port.

Most worm scanning methods can be detected by monitoring anomalies on these three characteristics. The methods of random scanning and sequential scanning could be easily found by looking at either (a) or (b), while hit-list scanning and routable scanning can be often detected by looking at (b) alone. Assuming a worm operated on a specific port, (c) could detect it regardless of its scanning method (unless it is using passive scanning).

Worm distribution can often be found using the same anomaly-based detection methods as for the detection of scanning, as connection to a victim and transferring the worm to the same victim often happen in parallel. Otherwise, an anomaly that can detect worms being sent out is checking if many outgoing packets contain identical byte sequences. Of course, this could only work if the worm is not obfuscated in any way.

Finally, the activation of a worm on a machine can be also uncovered by anomaly-based detection systems. More specifically, anomaly-based detection is effective in finding worms which activate on their own without user interaction. The detection systems looking towards worm self-activation will often look for buffer overflows in programs by using “taint tracking”, which is a system wherein incoming network packets are marked with a “taint” if they are suspicious, and the system then detects if the return address of a function in some program is overwritten by taint-marked data. The latter method is described in (Costa, Crowcroft and Castro).

Signature-Based Detection

The other major type of worm detection is that which rely on previously recorded signatures. Signatures can be anything from byte sequences in packets to sequences of system calls, and as well can be made by a user or generated automatically by an anomaly-based detection system. The main characteristic of a worm that signature-based detection focuses on is the payload, i.e., the code of the worm that makes it take malicious action. Man-made signatures are often crafted by first running a worm in a controlled environment and then recording strings in the packets it sends or system calls it uses.

Additionally, signature-based detection can help in finding the activation of a worm when it cannot be detected by the anomaly-based systems, i.e., when the worm does not activate on its own. Mainly, antivirus software will often detect a worm either when it is written to the disk or when a user attempts to run it – most antivirus software is signature-based, and when a user attempts to run a worm (or any malicious program) the antivirus software will prevent him or her from doing so. Similarly, signature-based detection can be used to detect the transference of worms in the form of firewalls – most firewalls can detect malicious inbound and outbound packets and drop them.

Tools Used in Worm Detection

Honeypots. For anomaly detection, one of the most simple ways to find a scanning worm is to set up a *honeypot*, which is a host device on a network that is designed such that it should not be receiving any network traffic under normal circumstances. Thus, when/if it does, one can determine that something is amiss – if no machine or user on the network has connected to it accidentally, it could be a scan from a worm. So, if one sets up many honeypots on a network and they all suddenly get a packet from the same source, one could track down where the worm is scanning from. The main limitation with using a honeypot is intrinsic in its design: it cannot actively seek out worms, only sit and wait for one to find it. So, it is mainly effective against random and sequential scanning worms, and less so against hit-list and passive scanning. A honeypot could be effective against routable scanning worms if one plants a target towards it on the infected host. Some open-source honeypot tools include LaBrea (lorgor) and Honeycomb (Kreibich).

Intrusion Detection Systems. A class of tools useful in both anomaly-based and signature-based detection of worms is that of Intrusion Detection Systems, or IDS's. These tools can either operate on either a network or host level. Most IDS tools have customizable detection "rules" as well as signatures to monitor potential anomalies and malicious packets, respectively. The leverage of the rules with the IDS makes them act as automated packet sniffers, raising alarms if anything is suspicious. Oftentimes the signatures for these tools can be written in regular expressions, and other times the tools may have their own languages for writing signatures. With all of this in mind, IDS's can detect the target scanning, transference, and payload of worms. Some examples of open-source network-based IDS's include Snort (Combs) and Zeek (formerly "Bro") (The Zeek Project).

Worm Prevention and Remediation Techniques

The just as important as the knowledge of how to detect worms are both the ability to prevent them from entering a system and the processes to treat machines infected with them. The prevention of allowing a computer worm to infect a network or host machine branches into a two-pronged strategy: education and gatekeeping.

A worm infects a machine oftentimes by sneaking by the detection systems and system administrators, but can also do so by duping the average user on a network. When we speak broadly of education when it comes to preventing computer worm infection, we mean that it is required that every user on a network is computer security "literate" to some extent. It is pertinent that each user follows best practices in order to prevent all viruses (including worms) from being levied on the system. Such best practices include:

- *Keeping all software and operating systems updated*. In 2012, cybersecurity firm Kaspersky found that one half of all cyberattacks on individual computers were conducted through unpatched Java (Oracle Java surpasses Adobe Reader as the most frequently exploited software). This is not unique – most cybersecurity exploits, including those that computer worms take advantage of, are on unpatched software. Attacks using zero-day exploits (those unknown to and unpatched by a software's creator/maintaining entity) make up only a tiny fraction of all attacks performed each year.
- *Regular use of antivirus and firewalls*. Having users on a network regularly scan their machines using proprietary antivirus software, as well as maintaining usage of firewalls, is usually more than enough to prevent a worm attack on a network.
- *Avoiding social engineering attacks*. A user could be fooled by a phishing campaign and unsuspectingly open an untrusted email or attachment to an email that could be a computer worm or other virus. Ensuring that users are trained in identifying shady communications and that they know not to click on advertisements or visit untrusted sites pays dividends in preventing viruses (touhid58).

The second blanket prevention method, "gatekeeping", refers to the mechanisms that can be used at the router and computer levels to stop malicious traffic from ever entering a network or machine. One such mechanism that is standard to most modern routers is the Access Control List, or ACL. The ACL of a router or switch is a list of rules that detail what should be permitted or denied when packets attempt to enter or leave a network. At this router level, one should also employ address blocking – that is, when an entity is identified through detection methods as infected, packets coming in from said entity are blocked (Li, Salour and Su). Firewalls, previously mentioned in only their capacity to detect worms, can also be used on individual machines to filter malicious packets, and thus prevent worms as well.

In what could be thought of as a combination of the two methods, using modern internet standards in ones network can also keep worms from spreading through it. For instance, using IPv6 (which has become the IEEE Internet Standard since 2017) rather than the archaic IPv4 increases the address space size of a network from 2^{32} to 2^{64}. This increase makes it virtually impossible for random scanning worms to spread through an entire subnet (Zou, Towsley and Cai). Additionally, using Network Address Translation, or NAT, also hinders worm scanning by decreasing the amount of globally-reachable hosts and reducing the efficiency of the worm to spread from within the private address space (if the infected host is inside of it) to outside of it (Rajab, Monrose and Terzis).

However, completely preventing worms from entering a network may not be possible. Once a worm has been detected and identified in a network, the next logical step one takes is figuring out how to prevent it from infecting additional machines and figuring out how to remove it from already infected ones. The former usually comes down to slowing the worms spread and tricking it into trapping itself, whereas the latter is mostly using the detection methods to locate infected machines and letting antivirus software deal with the worm causing said infection.

Coming back to the address blocking method mentioned earlier, it can be implemented at an individual host-by-host level as well. That is, each host in a network maintains a blacklist, and if it is detected that any other hosts are exhibiting behavior suggestive of infection, packets coming from them are dropped until the infection is remedied. In addition to blocking certain addresses, using signature-based detection hosts can block packets based on their content, as mentioned in previous sections. This would allow safe traffic to come in from possibly infected machines, but not malicious packets. Both host-level address blocking and content-based blocking of packets can help slow the spread of a worm through a network.

As stated, tricking the worm into isolating itself is also a popular method of containment. “Black holing” is a common method similar to using an ACL or address blocking – it involves specifying an address to either a non-existent or non-running machine on a network, and forwarding certain traffic (in this case, worm-related traffic) to it, where it is discarded without letting the sender know it didn’t go to its intended target. In many networks there is a “null0” interface, which is an always-running simulated interface which is not able to forward or send traffic outside itself (Cisco Systems, Inc.). This would then make the worm believe it is infecting all the hosts it wants to on a certain network, and it would not continue attempts after it had finished.

Another method of deceiving the attacking worm comes from a clever use of the Honeypot we discussed in a prior section. Niels Provos of Google proposed a system of using a virtual system of many honeypots as a decoy of sorts – it would work by having a small amount of virtual honeypots (which simulate only the network stack of many different OS’s, rather than entire virtual machines) set up initially, and waiting until one of them detected a worm had connected to it. Once one has a worm, a very large volume of other virtual honeypots with vulnerable applications on them would be deployed continuously and lure the worm into attacking them, rather than actual machines. In his research, Provos tested the system by deploying it 20 minutes after a worm had started infecting a network, wherein over 250,000 virtual honeypots deployed and essentially stopped the spread of the test worm to real devices completely (Provos).

Novel Research

Emerging research in the field of worm propagation modelling and detection methodology are done within all of the emerging topics in the broader subject of Computer Science research. This includes but is not limited to machine learning, Mobile Ad-Hoc Networks (MANETs), Vehicular Ad-Hoc Networks (VANETs), proliferation to various non-PC devices, and so on.

Detecting Worm Scanning with Machine Learning. (Ochieng, Mwangi and Ateya) proposed to use machine learning models to detect maliciousness of network packets and aimed to determine the most accurate model and the most important features. Using the "Three Days of Conficker" Dataset to extract samples of malicious packet fragments and the "Two Days in November 2008" Dataset to get non-malicious samples, various learning algorithms were applied such as k-Nearest Neighbor, Naïve Bayes, etc. The researchers found the C5.0 Decision Tree algorithm to be most accurate, identifying packets with the Conficker Worm with 95.75% accuracy. Also determined was that the value of the packet snippet and the source IP address were the most important features extracted. The paper showed that machine learning can be used to automate anomaly-based detection, possibly to be implemented in Intrusion Detection Systems.

Detecting Payload Based Internet Worms with Neural Network Classifier. (Tharani and Leelavathi) sought to detect internet worms using common behaviors rather than signature identification. Using the KDD CUP99 and CAIDA datasets for their study, the team used common parameters including source and destination IP addresses and ports to train the neural network. Detection of worms was then marked by selection and ranking of features, scored by counting the frequency in both bad and good datasets and then creating a function to mark each one. After building the neural network, the detection system was then able to determine a precision rate of over 95% (Tharani and Leelavathi). This work, along with the previously described work, certainly shows a promising outlook in the incorporation of machine learning to aid in detection systems.

Worm Propagation in VANETs. (Trullols-Cruces, Fiore and Barcelo-Ordinas) analyzed the ways in which a worm can spread vehicle-to-vehicle through a large road network, modelling the propagation speed possible depending upon the propagation method and location of the "patient-zero" vehicle. The paper found that VANETs are unfortunately ideal for worm spread due to the mobility of the network nodes, the vehicles, and the high volume of connections being made by them at any given time. The researchers also concluded that, for the same reasons that a worm can spread very rapidly, as well as the assumption that an area where a worm outbreak occurs is likely to be covered by 4G cellular coverage and smart vehicles have 3G or 4G radios, patching of vulnerabilities can be leveraged very quickly over a VANET. The difficulty in the patching lies in accurately estimating the area of spread, for which the researchers used the same modelling as for tracking the spread of a worm.

PLC-Blaster. (Spenneberg, Bruggemann and Schwartke) demonstrated the viability of a worm that lives on and transfers to other programmable logic controllers (PLCs) without the need of connection to traditional computer systems, such as PCs, to operate. The research was done on the SIMATIC S7-1200v3, a new model at the time of publication. The paper documents how a worm can detect targets using unique ports, transfer by mimicking a proprietary protocol (the Siemens TIA-Portal protocol), and inject into already-running user programs to activate under "Human-Activity based" activation. The paper concluded that of the three protection features offered by the model of PLC tested, only one could stop the worm (being the Access Protection feature wherein access to the PLC is protected by a password). The paper did note though that this is disabled by default, effectively concluding that the designed worm could spread through a PLC network easily.

NADTW. (Anbar, Abdullah and Munther) determined a novel approach to TCP worm detection. The standard methods have been previously discussed, however, NADTW is a new approach that lies in the assumption of TCP worms performing network scanning to find an initial target. Fundamentally, this new technique is a composite of other pre-existing methods: statistical cross-relation for network scanning detection (SCANS) and destination source correlation detection (DSC). Initially, SCANS is performed to detect "both TCP and UDP random and sequential scanning" (Anbar, Abdullah and Munther). If a source IP is identified using the SCANS approach, NADTW proceeds onto the DSC detection phase as detecting a scanner does not necessarily mean a worm is active. The flagged IP is observed to check for DSC behavior, this can be characterized by the machine at the address receiving data at a specific port and then sending data to other machines with the same destination port (Anbar, Abdullah and Munther). When the combination of the two are found, the detected IP is likely infected. In the end, Anbar, Abdullah, and Munther found NADTW to be a more efficient TCP worm detection method compared to the current method that relies solely on DSC detection.

Future Research

Further developments are to be made in the world of worms. One topic not thoroughly discussed that is beginning to appear in research is the transmission of worms in a wireless network. As one might expect, network traffic is much different when done through air rather than wire. More research should be done on the security of wireless protocols and the different devices it affects.

In addition to wireless research, worms are still clear threats to systems. As previously discussed, the WannaCry worm was relatively recent and impacted major offices, including hospitals. The question that remains is how IT administrators can effectively monitor and catch worms before they can attack. These types of questions have been covered, but it is clear that no perfect solution exists, will there ever be? More research needs to be done in this area, there needs to be further development in response to the question of, “What more can one do to prevent this?”

Lastly, pertaining to threat detection and prevention, one should look deeper into preventing users from bringing worms into a network. It has been mentioned that the user is the number one vulnerability in the system, how does one restrict them while also allowing the same level of access? There are certainly ways to give the illusion of free access, research may be able to find a better way to lessen the threat that a user poses while making them feel just as capable.

Ultimately, the further research boils down to *what more can be done?* Prevention and detection has improved as the knowledge of worms has grown, but there is certainly more to handle. All viruses evolve over time, and worms will follow suit, what must be done is to stay ahead of it, to determine how it will evolve and discover the weaknesses it may exploit before it can act.

Conclusion

Worms are one of the better understood viruses, yet they maintain the same threat. The composition is clear: how it starts, how it searches, how it spreads; more has been done in detecting it, preventing it, squashing it before it has the chance to act. As devices and protocols continue to expand and change in different ways, this malware will find new vulnerabilities, better ways to propagate, learn to move with greater stealth. This is no different than any other virus in that it persists, so as the systems change, it will follow; but, research and development will also follow, it will discover the new worms and how they work, with it, better techniques will continue to extinguish this threat.

Works Cited

Anbar, Mohammed, et al. "NADTW: new approach for detecting TCP worm." *Neural Computing & Applications* (2017): 525-538.

Boukerche, Azzedine and Qi Zhang. "Countermeasures against Worm Spreading: A New Challenge for Vehicular Networks." *ACM Comput. Surv.* (2019): 1-25.

Chen, Thomas M. "Trends in Viruses and Worms." *The Internet Protocol Journal* (2003): 23-33.

Cisco Systems, Inc. "REMOTELY TRIGGERED BLACK HOLE FILTERING—DESTINATION BASED AND SOURCE BASED." 2005. *Cisco.* Document. 23 November 2020.

Combs, Russ. "snort3." 16 November 2020. *GitHub.* Repository. 23 November 2020.

Costa, Manuel, et al. "Vigilante: end-to-end containment of internet worms." *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*. New York, NY, USA: ACM Press, 2005. 133-147.

Edwards, Sam and Ioannis Profetis. "Hajime: Analysis of a decentralized internet worm for IoT devices." Paper. 2016.

Fosnock, Craig. "Computer Worms: Past, Present, and Future." 30 September 2005. *Infosec Writers.* Article. 22 November 2020.

Gu, Guofei, et al. "Worm Detection, Early Warning and Response Based on Local Victim Information." *Proc. 20th Annual Comp. Sec. Apps. Conf.* 2004.

Karnouskos, Stamatis. "Stuxnet worm impact on industrial cyber-physical system security." *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*. Melbourne, VIC, Australia: IEEE, 2011. 4490-4494.

Khanh, Nguyen Huu. "Dynamics of a Worm Propagation Model with Quarantine in Wireless Sensor Networks." *Applied Mathematics & Information Sciences* (2016): 1739-1746.

Kreibich, Christian. *Honeycomb*. 26 July 2019. 23 November 2020.

Li, Pele, Mehdi Salour and Xiao Su. "A survey of internet worm detection and containment." *IEEE Communications Surveys & Tutorials* (2008): 20-35.

lorgor. *LaBrea: "Sticky" Honeypot and IDS*. n.d. 23 November 2020.

Mohurle, Savita and Manisha Patil. "A brief study of Wannacry Threat: Ransomware Attack 2017." *International Journal of Advanced Research in Computer Science* (2017): 1938-1940.

Ochieng, Nelson, et al. "Detecting Scanning Computer Worms Using Machine Learning and Darkspace Network Traffic." (2017).

"Oracle Java surpasses Adobe Reader as the most frequently exploited software." 21 December 2012. *kaspersky.* 24 November 2020.

Orman, Hilarie. "The Morris worm: a fifteen-year perspective." *IEEE Security & Privacy* (2003): 35-43.

Pratama, Andhika and Fauzi Adi Rafrastara. "Computer Worm Classification." *International Journal of Computer Science and Information Security* (2012).

Provos, Niels. "A Virtual Honeypot Framework." October 2003. *The Regents of the University of Michigan* . PDF. 22 November 2020.

Qin, Xinzhou, et al. "Worm Detection Using Local Networks." (2004).

Rajab, Moheeb Abu, Fabian Monrose and Andreas Terzis. "On the impact of dynamic addressing on malware propagation." *Proceedings of the 4th ACM workshop on Recurring malcode (WORM '06)*. New York, NY, USA: Association for Computing Machinery, 2006. 51-56.

Schoch, John F. and Jon A. Hupp. "The “Worm” Programs—Early Experience with a Distributed Computation." *Commun. ACM* 25 (1982): 172-180.

Singh, Akansha, et al. "Modeling and Analysis of Worm Propagation in Wireless Sensor Networks." *Wireless Personal Communications* (2018): 2535-2551.

Smith, Craig, et al. "Computer Worms: Architectures, Evasion Strategies, and Detection Mechanisms." *Journal of Information Assurance and Security* (2009): 69-83.

Spenneberg, Ralph, Maik Bruggemann and Hendrik Schwartke. "PLC-Blaster: A Worm Living Solely in the PLC." *Black Hat '16*. Ed. OpenSource Security. Las Vegas, 2016.

Tharani, A and B Leelavathi. "Payload Based Internet Worm Detection Using Neural Network Classifier." *International Journal of Computer Science Engineering* (2017).

The Zeek Project. "zeek." 26 November 2020. *GitHub.* Repository. 2020 27 2020.

touhid58. "Tips on how to prevent computer worms?" 6 September 2019. *Cyber Threat Portal.* Article. 24 November 2020.

Trullols-Cruces, Oscar, Marco Fiore and Jose M. Barcelo-Ordinas. "Worm Epidemics in Vehicular Networks." *IEEE Transactions on Mobile Computing* (2015): 2173-2187.

Weaver, Nicholas, et al. "A Taxonomy of Computer Worms." *Proceedings of the 2003 ACM workshop on Rapid malcode*. New York, NY, USA: Association for Computing Machinery, 2003. 11–18.

Zou, Cliff, Don Towsley and Songlin Cai. "Routing Worm: A Fast, Selective Attack Worm Based on IP Address Information." *Workshop on Principles of Advanced and Distributed Simulation (PADS)*. IEEE, 2005. 199- 206.