Chapter 1

Practical Investigative Strategies

"A victorious army first wins and then seeks battle; a defeated army first battles and then seeks victory." –Sun Tsu, The Art of War[1]

Evidence scattered around the world. Not enough time. Not enough staff. Unrealistic expectations. Internal political conflicts. Gross underestimation of costs. Mishandling of evidence. Too many cooks in the kitchen. Network forensic investigations can be tricky. In addition to all the challenges faced by traditional investigators, network forensics investigators often need to work with unfamiliar people in different countries, learn to interact with obscure pieces of equipment, and capture evidence that exists only for fleeting moments. Laws surrounding evidence collection and admissibility are often vague, poorly understood, or nonexistent. Frequently, investigative teams find themselves in situations where it is not clear who is in charge, or what the team can accomplish.

An organized approach is key to a successful investigation. While this book is primarily designed to explore technical topics, in this chapter, we touch on the fundamentals of investigative management. This is particularly important because network forensics investigations often involve coordination between multiple groups and evidence that may be scattered around the globe.

We begin by presenting three cases from different industries in order to give you some examples of how network forensics is used to support investigations in the real world. Next, we explore the fundamentals of evidence collection and distinctions that are made between different types of evidence in many courts. We discuss the challenges specific to network-based evidence, such as locating evidence on a network and questions of admissibility. Finally, we present the OSCAR investigative methodology, which is designed to give you an easy-to-remember framework for approaching digital investigations.

1.1 Real-World Cases

How is network forensics used in real life? In this section, we present three cases:

- Hospital Laptop Goes Missing
- Catching a Corporate Pirate
- Hacked Government Server

1. Sun Tsu (Author), Lionel Giles (Translator), *The Art of War,* El Paso Norte Press, 2005.

These cases have been chosen to provide examples of common IT security incidents and illustrate how network forensics is frequently used. Although these cases are based on real-life experiences, they have been modified to protect the privacy of the organizations and individuals involved.

1.1.1 Hospital Laptop Goes Missing

A doctor reports that her laptop has been stolen from her office in a busy U.S. metropolitan hospital. The computer is password-protected, but the hard drive is not encrypted. Upon initial questioning, the doctor says that the laptop may contain copies of some patient lab results, additional protected health information (PHI) downloaded from email attachments, schedules that include patient names, birth dates, and IDs, notes regarding patient visits, and diagnoses.

1.1.1.1 Potential Ramifications

Since the hospital is regulated by the United States' Health Information Technology for Economic and Clinical Health (HITECH) Act and Health Insurance Portability and Accountability Act (HIPAA), it would be required to notify individuals whose PHI was breached.[2] If the breach is large enough, it would also be required to notify the media. This could cause significant damage to the hospital's reputation, and also cause substantial financial loss, particularly if the hospital were held liable for any damages caused due to the breach.

1.1.1.2 Questions

Important questions for the investigative team include:

1. Precisely when did the laptop go missing?
2. Can we track down the laptop and recover it?
3. Which patient data was on the laptop?
4. How many individuals' data was affected?
5. Did the thief leverage the doctor's credentials to gain any further access to the hospital network?

1.1.1.3 Technical Approach

Investigators began by working to determine the time when the laptop was stolen, or at least when the doctor last used it. This helped establish an outer bound on what data *could* have been stored on it. Establishing the time that the laptop was last in the doctor's possession also gave the investigative team a starting point for searching physical surveillance footage and access logs. The team also reviewed network access logs to determine whether the laptop was subsequently used to connect to the hospital network after the theft and, if so, the location that it connected from.

2. "HITECH Breach Notification Interim Final Rule," U.S. Department of Health and Human Services, http://www.hhs.gov/ocr/privacy/hipaa/understanding/coveredentities/breachnotificationifr.html.

There are several ways investigators could try to determine what time the laptop went missing. First, they could interview the doctor to establish the time that she last used it, and the time that she discovered it was missing. Investigators might also find evidence in wireless access point logs, Dynamic Host Control Protocol (DHCP) lease assignment logs, Active Directory events, web proxy logs (if there is an enterprise web proxy), and of course any sort of laptop tracking software (such as Lojack for Laptops) that might have been in use on the device.

Enterprise wireless access point (WAP) logs can be especially helpful for determining the physical location in the facility where a mobile device was most recently connected, and the last time it was connected. In order to ensure uniform availability of wireless networks, enterprises typically deploy a fleet of WAPs that all participate in the same network. Although they appear to the end user as a single wireless network, network operators can view which mobile devices were connected to specific access points throughout the building. Some enterprises even have commercial software that can graphically represent the movement of wirelessly connected devices as they traverse the physical facility. If the laptop was still connected to the hospital's wireless network as the thief exited the building, investigators might be able to use wireless access point logs to show the path that the thief navigated as he or she exited the building. This might also be correlated with video surveillance logs or door access control logs.

Once investigators established an approximate time of theft, they could narrow down the patient information that might have been stored on the system. Email logs could reveal when the doctor last checked her email, which would place an outer bound on the emails that could have been replicated to her laptop. These logs might also reveal which attachments were downloaded. More importantly, the hospital's email server would have copies of all of the doctor's emails, which would help investigators gather a list of patients likely to have been affected by the breach. Similarly, hospital applications that provide access to lab results and other PHI might contain access logs, which could help investigators compile a list of possible data breach victims.

There might be authentication logs from Active Directory domain controllers, VPN concentrators, and other sources that indicate the laptop was used to access hospital resources even after the theft. If so, this might help investigators track down the thief. Evidence of such activities could also indicate that additional information was compromised, and that the attacker's interests went beyond merely gaining a new laptop.

1.1.1.4 Results

Leveraging wireless access point logs, the investigative team was able to pinpoint the time of the theft and track the laptop through the facility out to a visitor parking garage. Parking garage cameras provided a low-fidelity image of the attacker, a tall man wearing scrubs, and investigators also correlated this with gate video of the car itself as it left the lot with two occupants. The video was handed to the police, who were able to track the license plate. The laptop was eventually recovered amongst a stack of stolen laptops.

The investigative team carefully reviewed VPN logs and operating system logs stored on the central logging server and found no evidence that the doctor's laptop was used to attempt any further access to hospital IT resources. Hard drive analysis of the recovered laptop showed no indication that the system had been turned on after the theft. After extensive consultation with legal counsel, hospital management concluded that patient data had ultimately not been leaked.

In response to the incident, the hospital implemented full-disk encryption for all laptop hard drives, and deployed physical laptop locking mechanisms.

1.1.2 Catching a Corporate Pirate

GlobalCorp, Inc., has a centrally managed intrusion detection system, which receives alerts from sites around the world. Central security staff notice an alert for peer-to-peer (P2P) file-sharing, and on closer inspection see filename references to movies that are still in theaters. Fearing legal ramifications of inaction, they investigate.

1.1.2.1 Potential Ramifications

Management at GlobalCorp, Inc., were highly concerned that an employee was using the company network for trafficking in pirated intellectual property. If this activity were detected, the owner of the intellectual property might sue the company. This case occurred in 2003, at the height of Digital Millennium Copyright Act (DMCA) fervor, and it was assumed that if an individual within the company was illicitly trading pirated music or movies, then it could place the company at risk of costly legal battles.

1.1.2.2 Questions

Important questions for the investigative team include:

1. Where is the source of the P2P traffic physically located?
2. Which user is initiating the P2P traffic?
3. Precisely what data is being shared?

1.1.2.3 Technical Approach

Using the IP address from the IDS alerts, investigators identified the physical site that was the source of the traffic. In order to specifically identify the client system, its location, and primary user, investigators worked with local network management staff.

Meanwhile, intrusion analysts in the central office began capturing all of the P2P-related packets involving the IP address in question. The local facility confirmed that this IP address was part of a local DHCP pool on the wired local area network (LAN). Intrusion analysts reviewed DHCP lease assignment logs for relevant time periods, and recovered the media access control (MAC) address associated with the suspicious activity. From the MAC address investigators identified the manufacturer of the network card (Dell, in this case).

In order to trace the IP address to a specific office, local networking staff logged into switches and gathered information mapping the IP address to a physical port. The physical port was wired to a cubicle occupied by an email system administrator. Investigators entered his office after hours one evening and recovered his desktop for forensic analysis.

Upon examination, however, it was clear that the confiscated desktop was not the source of the P2P activity. The MAC address of the network card in the confiscated system (a Hewlett-Packard desktop) was not consistent with the MAC address linked to the suspicious activity. Subsequent analysis of the company's email server produced evidence that the suspect, an email system administrator, had leveraged privileged access to read emails of key networking staff involved in the investigation.

Local networking staff took caution to communicate out-of-band while coordinating the remainder of the investigation. Investigators conducted a thorough search of the premises for a system with the MAC address implicated. The matching desktop was eventually found in the desktop staging area, buried in a pile of systems queued for reimaging.

1.1.2.4 Results

Network forensic analysts examined full packet captures grabbed by the IDS, and were ultimately able to carve out video files and reconstruct playable copyrighted movies that were still in theaters. Hard drive analysis of the correct desktop produced corroborating evidence that the movies in the packet capture had been resident on the hard drive. The hard drive also contained usernames and email addresses linking the hard drive and associated network traffic with the suspect.

Case closed!

1.1.3 Hacked Government Server

During a routine antivirus scan, a government system administrator was alerted to suspicious files on a server. The files appeared to be part of a well-known rootkit. The server did not host any confidential data other than password hashes, but there were several other systems on the local subnet that contained Social Security numbers and financial information of thousands of state residents who had filed for unemployment assistance. The administrative account usernames and passwords were the same for all servers on the local subnet.

1.1.3.1 Potential Ramifications

State laws required the government to notify any individuals whose Social Security numbers were breached. If the servers containing this sensitive information were hacked, the state might be required to spend large amounts of money to send out notifications, set up hotlines for affected individuals, and engage in any resulting lawsuits. In addition, disclosure of a breach might damage the careers of high-ranking elected state officials.

1.1.3.2 Questions

Important questions for the investigative team include:

- Was the server in question truly compromised?
- If so, how was the system exploited?
- Were any other systems on the local network compromised?
- Was any confidential information exported?

1.1.3.3 Technical Approach

The server in question appeared to contain files with names that fit the pattern for a well-known rootkit. Investigators began by examining these files and concluded that they were, indeed, malicious software. The rootkit files were found in the home directory of an old local administrator account that staff had forgotten even existed.

Investigators found that the local authentication logs had been deleted. Fortunately, all servers on the subnet were configured to send logs to a central logging server, so instead investigators reviewed Secure Shell (SSH) logs from the central logging server that were associated with the account. From the SSH logs, it was clear that the account had been the target of a brute-force password-guessing attack. Investigators used visualization tools to identify the times that there were major spikes in the volume of authentication attempts. A subsequent password audit revealed that the account's password was very weak.

The SSH logs showed that the source of the brute-force attack was a system located in Brazil. This was surprising to IT staff because according to network documentation the perimeter firewall was supposed to be configured to block external access to the SSH port of servers on the subnet under investigation. Investigators gathered copies of the current, active firewall configuration and found that it did not match the documented policy—in practice, the SSH port was directly accessible from the Internet. Subsequently, investigators analyzed firewall logs and found entries that corroborated the findings from the SSH logs.

When one system in the environment is compromised, there is a significant probability that the attacker may use credentials from that system to access other systems. IT staff were concerned that the attacker might have used the stolen account credentials to access other systems on the local subnet.

Fortunately, further analysis of the server hard drive indicated that the attacker's access was short-lived; the antivirus scan had alerted on the suspicious files shortly after they were created. Investigators conducted a detailed analysis of authentication logs for all systems on the local subnet, and found no other instances of suspicious access to the other servers. Furthermore, there were no records of logins using the hacked account on any other servers. Extensive analysis of the firewall logs showed no suspicious data exportation from any servers on the local subnet.

1.1.3.4 Results

Investigators concluded that the server under investigation was compromised but that no other systems on the local subnet had been exploited and no personal confidential information had been breached. To protect against future incidents, the state IT staff corrected the errors in the firewall configuration and implemented a policy in which firewall rules were audited at least twice per year. In addition, staff removed the old administrator account and established a policy of auditing all server accounts (including privileges and password strength) on a quarterly basis.

1.2 Footprints

When conducting network forensics, investigators often work with live systems that cannot be taken offline. These may include routers, switches, and other types of network devices, as well as critical servers. In hard drive forensics, investigators are taught to minimize system modification when conducting forensics. It is much easier to minimize system modification when working with an offline copy of a write-protected drive than with production network equipment and servers.

In network forensics, investigators also work to minimize system modification due to forensic activity. However, in these cases investigators often do not have the luxury of an offline copy. Moreover, network-based evidence is often highly volatile and must be collected through active means that inherently modify the system hosting the evidence. Even when investigators are able to sniff traffic using port monitoring or tapping a cable, there is always some impact on the environment, however small. This impact can sometimes be minimized through careful selection of acquisition techniques, but it can never be eliminated entirely.

Every interaction that an investigator has with a live system modifies it in some way, just as an investigator in real life modifies a crime scene simply by walking on the floor. We use the term "footprint" throughout this book to refer to the impact that an investigator has on the systems under examination.

You will always leave a footprint. Often, the size of the footprint required must be weighed against the need for expediency in data collection. Take the time to record your activities carefully so that you can demonstrate later that important evidence was not modified. Always be conscious of your footprint and tread lightly.

1.3 Concepts in Digital Evidence

What is evidence? The *Compact Oxford English Dictionary* defines "evidence" as:[3]

evidence (noun)

1. information or signs indicating whether a belief or proposition is true or valid.
2. information used to establish facts in a legal investigation or admissible as testimony in a law court.

In this book, we are concerned with both of the above definitions. Our goal in many investigations is to compile a body of evidence suitable for presentation in court proceedings (even if we hope never to end up in court!). Both relevance to the case and admissibility are important, but the first goal is to ascertain the facts of the matter and understand truly and correctly what has transpired.

Consequently, we define "evidence" in the broadest sense as any observable and recordable event, or artifact of an event, that can be used to establish a true understanding of the cause and nature of an observed occurrence.

Of course, it's one thing to be able to reconstruct and understand the events that comprise an occurrence, and yet another to be able to demonstrate that in such a way that

3. "Oxford Dictionaries Online—English Dictionary and Language Reference," *Oxford Dictionaries*, http://www.askoxford.com/concise_oed/evidence?view=uk.

Chapter 2

Technical Fundamentals

"If you know the enemy and know yourself, you need not fear the results of a hundred battles." –Sun Tsu, The Art of War[1]

The Internet ecosystem is varied and complex. In any network forensic investigation, there are an enormous number of places where evidence may live, some more accessible than others. Often, network-based evidence exists in places that local networking staff may not have considered, leaving it up to the investigator to review the network diagrams and make suggestions for evidence collection.

On the flip side, many networks are configured for functionality and performance, not for monitoring or auditing—and certainly not for forensic investigations. As a consequence, the specific instrumentation or evidence that an investigator might wish for may not exist. Many vendors do not include rich data recording capabilities into their products, and even when they do, such capacities are often disabled by default. For example, local switches may not be configured to export flow record data, or server logfiles might roll over and overwrite themselves on a daily basis.

From Nortel's line of enterpise-class networking equipment to Linksys' ubuquitous WRT54G router series in homes and small businesses, organizations can select from a wide variety of equipment for any networking function. As a result, the experience of the network forensic analyst is sure to be varied, inconsistent, and challenging.

In this chapter, we take a high-level look at classes of networking devices, discuss their value for the forensic investigator, and briefly touch upon how their contents might be retrieved. Subsequently, we review the concept of protocols and the OSI model in the context of network forensic investigations. Finally, we talk about key protocols in the Internet Protocol suite, including IPv4, IPv6, TCP, and UDP. This will provide you with a strong technical and conceptual foundation, which we build upon throughout the remainder of this book.

2.1 Sources of Network-Based Evidence

Every environment is unique. Large financial institutions have very different equipment, staff, and network topologies than local government agencies or small health care offices. Even so, if you walk into any organization you will find similarities in network equipment and common design strategies for network infrastructure.

1. Sun Tsu (Author), Lionel Giles (Translator), *The Art of War*, El Paso Norte Press, 2005.

There are many sources of network-based evidence in any environment, including routers, web proxies, intrusion detection systems, and more. How useful is the evidence on each device? This varies depending on the type of investigation, the configuration of the specific device, and the topology of the environment.

Sometimes data from different sources can overlap, which is useful for correlation. Other times the data sources are merely complementary, each containing snippets of evidence not found elsewhere. In this section, we review the types of network equipment that are commonly found in organizations and discuss the types of evidence that can be recovered from each.

2.1.1 On the Wire

Physical cabling is used to provide connectivity between stations on a LAN and the local switches, as well as between switches and routers. Network cabling typically consists of copper, in the form of either twisted pair (TP) or coaxial cable. Data is signaled on copper when stations on the shared medium independently adjust the voltage. Cabling can also consist of fiber-optic lines, which are made of thin strands of glass. Stations connected via fiber signal data through the presence or absence of photons. Both copper and fiber-optic mediums support digital signaling.

Forensic Value Network forensic investigators can tap into physical cabling to copy and preserve network traffic as it is transmitted across the line. Taps can range from "vampire" taps, which literally puncture the insulation and make contact with copper wires, to surreptitious fiber taps, which bend the cable and cut the sheathing to reveal the light signals as they traverse the glass. Many commercial vendors also manufacture infrastructure taps, which can plug into common cable connectors and are specifically designed to replicate signals to a passive station without degrading the original signal. (Please see Chapter 3, "Evidence Acquisition," for more details.)

2.1.2 In the Air

An increasingly popular way to transmit station-to-station signals is via "wireless" networking, which consists of radio frequency (RF) and (less commonly) infrared (IR) waves. The wireless medium has made networks very easy to set up. Wireless networks can easily be deployed even without "line-of-sight"—RF waves can and do travel through air, wood, and brick (though the signal will be degraded substantially by particularly dense materials). As a result, enterprises and home users can deploy wireless networks without the expense and hassle of installing cables.

Forensic Value Essentially, wireless access points act as hubs, broadcasting all signals so that any station within range can receive them. As a result, it is often trivial for investigators to gain access to traffic traversing a wireless network. The traffic may be encrypted, in which case investigators would have to either legitimately obtain the encryption key or break the encryption to access the content.

However, investigators can still gather a lot of information from encrypted wireless networks. Although data packets that traverse a wireless network may be encrypted, commonly management and control frames are not. In the clear, wireless access points advertise their names, presence, and capabilities; stations probe for access points; and access points respond to probes. Even when analyzing encrypted wireless traffic, investigators can identify

MAC addresses of legitimate authenticated stations, unauthenticated stations and suspicious stations that may be attacking the wireless network. Investigators can also conduct volume-based statistical traffic analysis and analyze these patterns.

With access to unencrypted or decrypted wireless traffic, of course, investigators can review the full packet captures in detail.

2.1.3 Switches

Switches are the glue that hold our LANs together. They are multiport bridges that physically connect multiple stations or network segments together to form a LAN. In modern deployments, we tend to see switches connected to other switches connected to other switches *ad nauseum* to form a complex switched network environment.

In a typical deployment, organizations have "core" switches, which aggregate traffic from many different segments, as well as "edge" switches,[2] which aggregate stations on individual segments. Consequently, traffic from one station to another within an enterprise may traverse any number of switches, depending on the physical network topology and the locations of the stations within it.

Forensic Value Switches contain a "content addressable memory" (CAM) table, which stores mappings between physical ports and each network card's MAC address. Given a specific device's MAC address, network investigators can examine the switch to determine the corresponding physical port, and potentially trace that to a wall jack and a connected station.

Switches also provide a platform from which investigators can capture and preserve network traffic. With many types of switches, network staff can configure one port to "mirror" (copy) traffic from any or all other ports or VLANs, allowing investigators to capture traffic from the mirroring port with a packet sniffer. Please see Section 3.1.4.1 for more detail.

2.1.4 Routers

Routers connect different subnets or networks together and facilitate transmission of packets between different network segments, even when they have different addressing schemes.

Routers add a layer of abstraction that enables stations on one LAN to send traffic destined for stations on another LAN. Internetworking using routers is what allows us to build campus-wide metropolitan area networks (MANs) or connect remote offices around the globe via wide area networks (WANs). From a certain perspective, the entire global Internet is nothing but a global area network (GAN), connected through a complex multilayer web of routers.

Forensic Value Where switches have CAM tables, routers have routing tables. Routing tables map ports on the router to the networks that they connect. This allows a forensic investigator to trace the path that network traffic takes to traverse multiple networks. (Note that this path can vary dynamically based on network traffic levels and other factors.)

Depending on the specific device's capabilities, routers may also function as packet filters, denying the transmission of certain types of traffic based on source, destination, or port.

2. "Encyclopedia—PC Magazine," http://www.pcmag.com/encyclopedia_term/0,2542,t=edge+switch&i=42362,00.asp.

Routers may log denied traffic (or sometimes maintain statistics on allowed traffic). Many enterprise-class routers can be configured to send logs and flow record data to a central logging server, which is extremely helpful for investigators, as this makes it very easy to correlate events from multiple sources. Logs stored on the router itself may be volatile and subject to erasure if the device is rebooted or if the available storage is exceeded.

In short, routers are the most rudimentary network-based intrusion detection systems in our arsenal, and also the most widely deployed.

2.1.5 DHCP Servers

The Dynamic Host Configuration Protocol (DHCP) is widely used as the mechanism for assigning IP addresses to LAN stations, so that they can communicate with other stations on the local network, as well as with systems across internetworked connections. In the early days of networking, administrators had to manually configure individual desktops with static IP addresses. DHCP was developed to provide automated IP address assignments, which could change dynamically as needed, dramatically reducing manual workload for administrators. DHCP service is often provided by the edge devices that perform routing between networks (routers, wireless access points, etc.), but it is not uncommon to find this service provided by infrastructure servers instead.

Forensic Value Frequently, investigations begin with the IP address of a host that is suspected of being involved in some sort of adverse event—whether it was the victim of an attack, the origin, or perhaps both. One of the first tasks the investigator must undertake is to identify and/or physically locate the device based on its IP address.

When DHCP servers assign (or "lease") IP addresses, they typically create a log of the event, which includes the assigned IP address, the MAC address of the device receiving the IP address, and the time the lease was provided or renewed. Other details, such as the requesting system's hostname, may be logged as well. Consequently, DHCP logs can show an investigator exactly which physical network card was assigned the IP address in question during the specified time frame.

2.1.6 Name Servers

Just as we need a mechanism to map MAC addresses to IP addresses, we also need a mechanism to map IP addresses to the human-readable names that we assign to systems and networks. To accomplish this, enterprises typically use the Domain Name System (DNS), in which individual hosts query central DNS servers when they need to map an IP address to a hostname, or vice versa. DNS is a recursive hierarchical distributed database; if an enterprise's local DNS server does not have the information to resolve a requested IP address and hostname, it can query another DNS server for that information.

Forensic Value DNS servers can be configured to log queries for IP address and hostname resolutions. These queries can be very revealing. For example, if a user on an internal desktop browses to a web site, the user's desktop will make a DNS query to resolve the host and domain names of the web server prior to retrieving the web page. As a result, the DNS server may contain logs that reveal connection attempts from internal to external systems, including web sites, SSH servers, external email servers, and more.

DNS servers can log not only queries, but also the corresponding times. Therefore, forensic investigators can leverage DNS logs to build a timeline of a suspect's activities.

2.1.7 Authentication Servers

Authentication servers are designed to provide centralized authentication services to users throughout an organization so that user accounts can be managed in one place, rather than on hundreds or thousands of individual computers. This allows enterprises to streamline account provisioning and audit tasks.

Forensic Value Authentication servers typically log successful and/or failed login attempts and other related events. Investigators can analyze authentication logs to identify brute-force password-guessing attacks, account logins at suspicious hours or unusual locations, or unexpected privileged logins, which may indicate questionable activities. Unlike analysis of authentication logs on a single hard drive, a central authentication server can provide authentication event information about all devices within an entire authentication domain, including desktops, servers, network devices, and more.

2.1.8 Network Intrusion Detection/Prevention Systems

Unlike the devices we've discussed so far, network intrusion detection systems (NIDSs) and their newer incarnations, network intrusion prevention systems (NIPSs), were specifically *designed* to provide security analysts and forensic investigators with information about network security–related events. Using several different methods of operation, NIDS/NIPS devices monitor network traffic in real time for indications of any adverse events as they transpire. When incidents are detected, the NIDS/NIPS can alert security personnel and provide information about the event. NIPSs may further be configured to block the suspicious traffic as it occurs.

The effectiveness of NIDS/NIPS deployments depends on many factors, including precisely where sensors are placed in the network topology, how many are installed, and whether they have the capacity to inspect the increasing volumes and throughputs of traffic we see in the modern enterprise environment. While NIDS/NIPS may not be able to inspect, or alert on, *every* event of interest, with a well-engineered deployment, they can prove indispensable.

Forensic Value At a high level, the forensic value of a NIDS/NIPS deployment is obvious: They are designed to provide timely data pertaining to adverse events on the network. This includes attacks in progress, command-and-control traffic involving systems already compromised, or even simple misconfigurations of stations.

The value of this data provided by NIDS/NIPS is highly dependent upon the capabilities of the device deployed and its configuration. With many devices it is possible to recover the entire contents of the network packet or packets that triggered an alert. Often, however, the data that is preserved contains little more than the source and destination IP addresses, the TCP/UDP ports, and the time the event occurred. During an ongoing investigation, forensic investigators can request that network staff tune the NIDS to gather more granular data for specific events of interest or specific sources and destinations. (Please see Chapter 7 for more detail.)

2.1.9 Firewalls

Firewalls are specialized routers designed to perform deeper inspection of network traffic in order to make more intelligent decisions as to what traffic should be forwarded and what traffic should be logged or dropped. Unlike most routers, modern firewalls are designed to make decisions based not only on source and destination IP addresses, but also based on the packet payloads, port numbers, and encapsulated protocols.

These days, nearly every organization has deployed firewalls on their network perimeters—the network boundaries between the enterprise and their upstream provider. In an enterprise environment, firewalls are also commonly deployed within internal networks to partition network segments in order to provide enclaves that are protected from each other. Even home users often have at least rudimentary firewall capabilities built into home routers, which are leased from their ISPs or purchased off-the-shelf.

Forensic Value Originally, event logging was of secondary importance for firewall manufacturers. Firewalls were not initially designed to *alert* security personnel when security violations were taking place, even though they were most definitely designed to implement security policies to prevent violations.

Today, modern firewalls have granular logging capabilities and can function as both infrastructure protection devices and fairly useful IDSs as well. Firewalls can be configured to produce alerts and log allowed or denied traffic, system configuration changes, errors, and a variety of other events. These logs can help operators manage the network and also serve as evidence for forensic analysts.

2.1.10 Web Proxies

Web proxies are commonly used within enterprises for two purposes: first, to improve performance by locally caching web pages and, second, to log, inspect, and filter web surfing traffic. In these deployments, web traffic from local clients is funneled through the web proxy. The web proxy may either allow or deny the web page request (often based on published blacklists of known inappropriate or malicious sites or on keywords in the outbound web traffic).

Once the page is retrieved, the web proxy may inspect the returned content and choose to allow, block, or modify it. For performance, the web proxy may cache the web page and later serve it to other local clients upon request so that there is no need to make multiple requests across the external network for the same web sites within a short period of time. The precise functionality of the web proxy is strongly dependant upon the specific configuration; some organizations choose to limit logging and run their web proxy only for performance reasons, whereas other organizations heavily monitor and filter incoming and outbound web traffic for security reasons.

There are also "anonymizing proxies," which are set up with the explicit purpose of providing anonymity to clients who deliberately funnel web surfing traffic through them. End clients send their traffic through an anonymous web proxy so that the remote web server only sees the proxy's IP address rather than the end-user's IP address. Depending on the precise configuration of the anonymizing proxy, the proxy server may still cache extensive information regarding the end-user's web surfing behavior.

Forensic Value Web proxies can be a gold mine for forensic investigators, especially when they are configured to retain granular logs for an extended period of time. Whereas forensic analysis of a single hard drive can produce the web surfing history for users of a single device, an enterprise web proxy can literally store the web surfing logs for an entire organization.

There are many commercial and free applications that can interpret web proxy logs and provide visual reports of web surfing patterns according to client IP address or even username (i.e., when correlated with Active Directory logs). This can help forensic analysts gather lists of users who may have succumbed to a phishing email, investigate a roving user's inappropriate web surfing habits, or identify the source of web-based malware. If the web proxy is configured to cache web pages, it is even possible to retrieve the content that an end-user viewed or carve malware out of a cached web page for further analysis.

2.1.11 Application Servers

Every organization has a variety of application servers, which vary depending on the organization's industry, mission, size, budget, and many other factors. Common types of application servers include:

- Database servers
- Web servers
- Email servers
- Chat servers
- VoIP/voicemail servers

Forensic Value There are far too many kinds of application servers for us to review every one in depth (there have been dozens if not hundreds of books published on each type of application server). However, when you are leading an investigation, keep in mind that there are many possible sources of network-based evidence. Review local network diagrams and application documentation to identify the sources that will be most useful for your investigation.

2.1.12 Central Log Servers

Central log servers aggregate event logs from a wide variety of sources, such as authentication servers, web proxies, firewalls, and more. Individual servers are configured to send logs to the central log server, where they can be timestamped, correlated, and analyzed by automated tools and humans far more easily than if they resided on disparate systems.

The precise contents of a central logging server vary enormously depending on the organization and applicable regulations. Once a luxury that many enterprises did not bother to invest in, deployment of central logging servers has been spurred by regulations and industry standards. As a result, they are much more common today than they were just a few years ago.

Forensic Value Much like intrusion detection systems, central log servers are designed to help security professionals identify and respond to network security incidents. Even if an individual server is compromised, logs originating from it may remain intact on the central log server. Furthermore, devices such as routers, which typically have very limited storage space, may retain logs for very short periods of time, but the same logs may be sent in real time to a central log server and preserved for months or years. Some organizations have installed commercial log analysis products that can provide forensic analysts with complex reports and graphical representations of log data, correlated across a variety of sources.

Chapter 12

Malware Forensics

''andy; I'm just doing my job, nothing personal, sorry''
–String found within the W32/MyDoom self-mailer worm code, circa 2004[1]

Malware is big business. As computers themselves have evolved to be increasingly networked, so too has malicious software, or "malware." Many people have remarked upon the strong analogies between malware and natural organisms, from self-reproductive techniques to the emergence of evolution. In real life, viruses, parasites, and bacteria spread by piggybacking on the normal mechanisms that hosts use to communicate and exchange resources. Similarly, as personal computers evolved from isolated word processors into complex network-oriented communications devices, the strategies and behaviors of malware have become increasingly network-oriented.

There are many goals of malware forensics, including:

- Understanding malware and associated vulnerabilities in order to produce antivirus/ IDS signatures
- Detecting compromised systems on the network
- Determining the scope of a breach, after the fact
- Containing an infection
- Tracking down the source of malware
- Gathering evidence for court

In this chapter, we explore malware forensics, particularly as it relates to network forensics, and discuss how network instrumentation can be used to help identify and track malware on the network. We begin by reviewing a few of the recent trends in malware evolution, and discuss how they impact network forensic techniques. Next, we discuss propagation, command-and-control, and payload behaviors, including ways that these can be detected and addressed through network forensics. Finally, we look to the future and discuss how network forensic investigators can prepare for the malware of tomorrow.

1. G. Sinclair, "Win32.Mydoom.B@mm (Win32.Novarg.B@mm) Removal Tool," *bitdefender*, January 28, 2004, http://www.bitdefender.com/VIRUS-1000035-en–Win32.Mydoom.B@mm-(Win32.Novarg.B@mm).html.

12.1 Trends in Malware Evolution

Once upon a time, malware was spread via floppy disks by infecting the MBR upon boot or through transfer of infected files and programs (i.e., freeware utilities). While USB drives and similar physical devices are still a notable source of malware, the vast majority of malware activity occurs over the network. Even when the malware is injected via a physical storage medium, it typically still communicates over the network after infecting a new host. Network forensics and malware analysis have increasingly overlapped as malware has evolved to become more dependant on the network for propagation, control, and payload functionality.

12.1.1 Botnets

Modern botnets are the convergence of advancements in remote control, automated propagation, and hierarchical, distributed management techniques. Despite their sophistication, botnets are not anything new. Long before the "Storm" botnet compromised millions of systems worldwide,[2] there were, for example, the rudimentary Tribe Flood Network and trinoo (1999), which were distributed networks of compromised hosts controlled by attackers through hierarchical, redundant command-and-control channels. The evolution of botnets from their early roots has been strongly influenced by the development of intrusion detection systems, which led malware developers to incorporate sophisticated IDS evasion and anti-forensic features.

12.1.1.1 Early Developments in Distributed Management

During the early to mid-1990s, attackers developed methods to remotely control compromised systems through the network. As networking became ubiquitous, there was an explosion of malware that took advantage of networked resources in order to send out email spam, conduct distributed denial-of-service (DDoS) attacks, host pirated software, and more. During this time period, compromised systems were dubbed "zombies" because they had rudimentary remote control and automation, usually limited to DDoS attacks or other simple behaviors.

During the late 1990s, attackers developed advanced systems for *centrally coordinating* the activities of hundreds or even thousands of compromised systems. The Tribe Flood Network (TFN), for example, emerged around 1999 in order to facilitate DDoS attacks.[3] Notably, it was designed with redundant command-and-control channels (often referred to as "C&C" or "C^2" channels) so that attacker systems controlled a network of compromised "client" systems, which in turn controlled an even larger network of "daemon" systems, which could be instructed to simultaneously attack victims upon command.[4]

2. Bruce Schneier, "Gathering 'Storm' Superworm Poses Grave Threat to PC Nets," *Wired.com*, October 4, 2007, http://www.wired.com/politics/security/commentary/securitymatters/2007/10/securitymatters_1004.

3. Dave Dittrich, "Tribe Flood Network," November 1, 1999, http://staff.washington.edu/dittrich/talks/cert/tfn.html.

4. Dave Dittrich, "The 'Tribe Flood Network' Distributed Denial of Service Attack Tool," October 21, 1999, http://staff.washington.edu/dittrich/misc/tfn.analysis.

The Tribe Flood Network 2000 (TFN2K) system incorporated additional anti–network forensics techniques, such as randomized and encrypted C&C packets that made traffic filtering difficult.[5]

12.1.1.2 Early Developments in Full-Featured Control

At the same time, other "black hat" and "gray hat" developers were focusing on extending the *features* of remote access trojans (RATs), designed to facilitate remote control of individual compromised endpoints. During the late 1990s, RATs such as the Back Orifice and Sub7 applications emerged. These provided remote attackers with a wide range of features, enabling powerful, point-and-click control of zombies. Neither BackOrifice nor Sub7 were initially designed for a hierarchical management of a distributed network of agents; the goal was to provide flexible, full-featured remote control of a compromised host by a single attacker.

Ultimately, botnet authors married sophisticated endpoint control with automated propagation techniques, automated self-update mechanisms, and multilayer, hierarchical, and/or peer-to-peer C&C channels. Botnets today incorporate many of the same features as legitimate enterprise networks, including internal DNS, web, email, and software update mechanisms. In accomplishing this, botnet authors have essentially built the equivalent of enterprise systems management software that scales far better than many aboveboard commercial offerings. This all succeeds in a hostile environment where thousands of people are constantly trying to disassemble the network. No wonder botnet developers are making money!

12.1.1.3 Implications for Network Forensics

To date, the most mature and most publicized aspects of malware analysis continue to focus on reverse-engineering the behavior of samples caught "in the wild." This is conducted in an effort to understand the nature and mechanisms of compromise and to develop reliable antivirus or IDS signatures for detecting the presence of malware. Such samples have historically been recovered by identifying a known-to-be-compromised host and extracting the malicious code from the running or powered-down system (i.e., from either memory or disk). Given the accompanying maturity of malware authors' attempts to obfuscate or hide their code on the compromised host, this can be a Herculean, if not impossible task.

As a result, network forensics has had to step into the breach—quite literally. Once malware began to emerge that was difficult or impossible to detect simply by monitoring host-based system activity, defenders turned to monitoring and analyzing hosts' external *network behaviors* to find, track, block, and prevent malware.

12.1.2 Encryption and Obfuscation

Encryption techniques have been used in malware since the early 1990s, when the Cascade virus encrypted its own payload in order to avoid detection by antivirus software.[6] Malware authors use encryption to hide functionality and create random-appearing payloads that are

5. Jason Barlow and Woody Thrower, "TFN2k An Analysis," *Packet Storm*, March 7, 2000, http://packetstormsecurity.org/files/view/10135/TFN2k_Analysis-1.3.txt.

6. Peter Szor, *The Art of Computer Virus Research and Defense* (Upper Saddle River, NJ: Addison-Wesley, 2005).

difficult for antivirus software and IDS systems to detect. Over time, malware has evolved increasingly sophisticated techniques to obscure decryptors and decryption keys (at times even requiring the malware to brute-force decrypt itself) in order to evade detection and forensic analysis.

12.1.2.1 Early IDS/Antivirus Evasion

Early network-based techniques to evade detection included session splicing and fragmentation. In session splicing, the attacker chops up a string from a session and splits it across multiple packets to foil NIDS/NIPS pattern matching. The network monitoring device must reassemble the session in order to detect the string, which is processor-intensive. Similarly, fragmentation attacks are designed to split individual packets into much smaller packets. The NIDS/NIPS must reassemble the packet fragments to properly analyze them, which uses up significant resources.

12.1.2.2 Modern Web Obfuscation/Encryption

Modern malware, which is often web-based, commonly leverages obfuscation techniques to embed malicious code (i.e., JavaScript) in web pages. Obfuscation takes the form of simple Base64 encoding or XOR-ing, or more complex layered systems of obfuscation. While malware analysts can and do de-obfuscate malicious code, often using automated tools, these techniques help attackers evade web filters and NIDS/NIPS systems, which are often unable to properly de-obfuscate on the fly.[7]

12.1.2.3 Hiding C&C Channels

With the rise of remote-control zombies and the maturation of C&C channels, malware began incorporating obfuscation and encryption not only to disguise the injection vectors and payloads, but also to hide the content of the C&C channel itself. This was a very important development that allowed botnets to evade detection and analysis through network forensics of packet contents, long after the initial compromise. For example, Symantec researcher Gilou Tenebro writes that the Waledac worm's HTTP-based command-and-control messages "[go] through at least four transformations before being sent to its peer. Anybody monitoring the HTTP packets on the wire will not easily be able to comprehend the messages."[8]

The result is that network forensic investigators must increasingly turn to statistical flow analysis rather than content-based analysis in order to efficiently detect and dissect malware on the network.

12.1.2.4 Maintaining Control

Encryption is used not just to evade detection and analysis by network defenders but also to ensure that the botherders maintain control of their networks. Some botnets require

7. "Malicious Hidden Iframes using Publicly Available Base64 encode/decode Script," *Zscaler Research*, May 2, 2010, http://research.zscaler.com/2010/05/malicious-hidden-iframes-using-publicly.html.

8. Gilou Tenebro, "W32.Waledac Threat Analysis," *Symantec*, 2009, http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/W32_Waledac.pdf.

client and/or server systems to cryptographically authenticate to the C&C channel, further foiling attempts by network forensic analysts to investigate or interfere with botnet functionality.

An interesting twist, illustrated by the Storm worm, is the use of encryption for the purposes of segmenting botnets to facilitate resale. Botnets themselves have become valuable commodities, rented and sold through underground black markets for purposes such as distributing spam, coordinating DDoS attacks, and stealing financial information. In 2007, the Storm worm began to use a weak (40-byte) encryption key to encrypt communications between peer-to-peer nodes (later, this was beefed up to 64-bit RSA encryption). While this made it more difficult to recover the contents of the packets on the fly, and added significant time and effort to traffic content analysis, malware analysts could still reverse-engineer samples to recover encryption keys and decrypt the packet contents. Joe Stewart of SecureWorks commented that the encryption did ensure that "each node will only be able to communicate with nodes that use the same key. This effectively allows the Storm author to segment the Storm botnet into smaller networks. This could be a precursor to selling Storm to other spammers, as an end-to-end spam botnet system, complete with fast-flux DNS and hosting capabilities."[9]

Note that Waledec, known as a "new and improved version of the Storm botnet," upgraded to AES 128-bit and RSA 1,024-bit encryption (along with Base64 encoding).[10]

12.1.3 Distributed Command-and-Control Systems

Early on, attackers realized that maintaining control of a compromised host was both challenging and also very important. With ongoing access facilitated by C&C channels, malware authors could use compromised hosts for a wide variety of purposes, and adapt on-the-fly as needed. Today, botherders have such excellent control over their networks that many are able to segment and sell botnets, easily transferring control to third parties.

In this section, we follow the evolution of C&C channels from simple, direct connections with central servers (i.e., IRC) to highly complex, distributed, redundant multilayer systems with built-in security features.

12.1.3.1 The Early Days: Internet Relay Chat (IRC)

Early botnets relied upon centralized control systems in which compromised agents communicated directly with a small group of central servers. Internet Relay Chat (IRC) has been a very common mechanism for malware command-and-control since the emergence of worms such as "Pretty Park" in 1999. Malware was hard-coded with server IP addresses or domain names, as well as IRC channel information, and then infected systems connected back to the central servers to drop off information about infected systems and/or receive commands and updates.

9. Joe Stewart, "The Changing Storm," *Dell SecureWorks*, October 14, 2007, http://www.secureworks.com/research/blog/index.php/2007/10/15/the-changing-storm/.

10. G. Sinclair, C. Nunnery, and B.B.H. Kang, "The Waledac Protocol: The How and Why," in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference*, 2010, 6977, http://isr.uncc.edu/paper/WaledacProtocolHowWhyMalware2009_remediation.pdf.

12.1.3.2 Drawbacks of Centralized C&C

Of course, centralized command-and-control channels had some major drawbacks for malware developers. They made it very easy for forensic analysts and security professionals to identify and track down compromised systems. Local organizations could simply alert on connection attempts to specific IP addresses or domains and block them, crippling the botnet. ISPs could shut down central command-and-control servers on their networks. Especially to the extent that IRC—a plain-text communications protocol—was being used, it was fairly straightforward to inspect, detect, and block that channel's abuse. (Within *most* organizations, IRC is really *only* used by botnets, with few users engaged in anything remotely legitimate via that protocol.)

The Stuxnet worm, which targets industrial control systems primarily in Iran, is an excellent example of how a sophisticated worm can be crippled due to reliance on a centralized command-and-control system. As described by researchers at Symantec:

> [The Stuxnet agent] contacts the command and control server on port 80 and sends some basic information about the compromised computer to the attacker via HTTP. Two command and control servers have been used in known samples:
>
> www[.]mypremierfutbol[.]com
>
> www[.]todaysfutbol[.]com
>
> The two URLs above previously pointed to servers in Malaysia and Denmark; however they have since been redirected to prevent the attackers from controlling any compromised computers.

Symantec set up sensors in July 2010 to track connection attempts back to Stuxnet command-and-control servers. Interestingly, in August 2010 they noted a sudden drop in newly infected connection attempts from Iran, as shown in Figure 12–1. "Looking at newly infected IP addresses per day, on August 22 we observed that Iran was no longer reporting new infections. This was most likely due to Iran blocking outward connections to the command-and-control servers, rather than a drop-off in infections." This illustrates how forensic analysts and security professionals can disrupt the operations of botnets—and other forensic analysts—simply by identifying and blocking known ports, IP addresses, and domains used for central command-and-control.[11]

12.1.3.3 Evolution Toward Distributed C&C

Newer botnets have been moving toward a partially distributed, rather than fully centralized, command-and-control architecture. In the distributed command-and-control model, individual endpoint agents do not connect directly back to a central server. Rather, they connect to servers in a redundant, distributed network. Distributed models typically involve a multilevel hierarchy so that systems at each level communicate with each other, and at least some systems in each level can communicate with systems in the level above. Using a

11. N. Falliere, L. Murchu, and E. Chien, "W32.Stuxnet Dossier: Version 1.4 (February 2011)," *Symantec*, 2011, http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

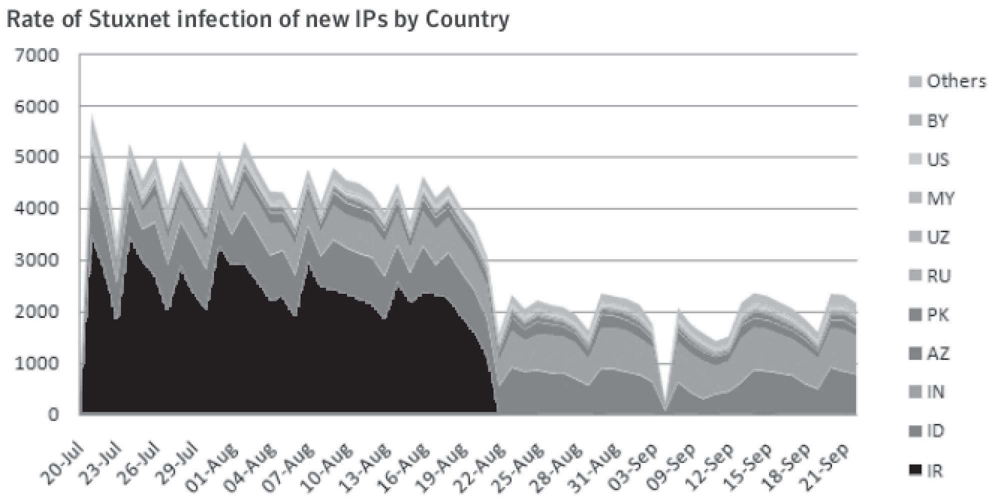


Figure 12–1. A chart from Symantec's "W32.Stuxnet Dossier," which illustrates a sudden drop in reported newly infected connection attempts from Iran (black). The infections were tracked based on connections to central C&C servers, which were likely blocked by Iran on August 22, 2010.[12] Reprinted with permission. [Note: Image colors have been modified to print in grayscale.]

distributed model, attackers can send commands and updates from a relatively small number of central servers to thousands or millions of compromised hosts, and the compromised hosts can pass information back up the chain with far greater redundancy and lower risk of detection/disruption.

12.1.3.4 Advantages of Distributed C&C

Distributed C&C offers the following advantages:

- Redundancy—If a C&C server is blocked or shut down, the functionality of the botnet is not affected.
- Detection Evasion—It is more difficult for forensic analysts to detect distributed C&C traffic. Centralized C&C traffic can be captured and analyzed at the perimeter of a LAN, where enterprises tend to focus monitoring resources, whereas a far greater percentage of decentralized C&C traffic is contained within a LAN and may never reach the perimeter firewall/IDS. Furthermore, organizations that track infections Internet-wide may never see evidence of internal botnet nodes, which communicate only between themseves. The Stuxnet worm actually incorporated some distributed C&C traffic in order to enable infections and updates of compromised hosts on internal LANs. Distributed C&C traffic is also less likely to generate a spike in traffic relating to a few systems, which commonly trigger alerts.

12. *Ibid.*

- Attacker Concealment—Distributed C&C channels make it far more difficult to track attacks back to central servers, since compromised endpoint nodes do not directly communicate back to central servers. In the mid-2000s, authorities such as the FBI began to investigate and prosecute botherders as part of coordinated programs including “Operation: Bot Roast.” These legal actions provided greater incentives for attackers to place more layers of indirection in between themselves and compromised botnet endpoints.[13, 14]
- Asymmetry—Distributed C&C peers can result in a sufficiently “meshed network” that the defender’s containment and eradication efforts are reduced to a frustrating and ineffective game of “Whac-A-Mole.” Small levels of effort on the attacker’s part can necessitate a disproportionate level of effort from the responders. Depending on the scale, this can be a game-changing element.

For example, the Downadup worm (also known as Conficker), which was capable of self-updating, responded to defense mechanisms by evolving increasingly distributed and dynamic C&C systems. The initial infections of W32.Downadup.A contacted one of 250 pseudorandom domains in five top-level domains each day for updates. Later, W32.Downadup.B expanded the number of top-level domains to eight.

With the update to W32.Downadup.C, the worm significantly changed its strategy for registering and querying domain names for use when downloading new instructions. This made it far more difficult for security professionals to alert upon and track its spread. As described by Porras et al. (SRI International) in their analysis of Conficker, “C further increases Conficker’s top-level domain (TLD) spread from five TLDs in Conficker A, to eight TLDs in B, to 110 TLDs that must now be involved in coordination efforts to track and block C’s potential DNS queries. With this latest escalation in domain space manipulation, C . . . represents a significant challenge to those hoping to track its census.”[15]

This was a classic example of *asymmetric warfare*, in that the malware author had only to recode his or her instruction set to expand the array of TLDs to be searched, and to redistribute. The result was an order-of-magnitude increase in effort required by *human* responders in order to identify and contain a threat that was largely automated.

News headlines in April 2009 reported that “Experts Bicker Over Conficker Numbers: Is it 4.6 million infected PCs or not?”[16] The number of infected systems became increasingly difficult to estimate as the malware evolved. “The Working Group got its data by setting up ‘sinkhole’ servers at points on the Internet used by infected machines to download instructions,” reported Bob McMillan of the IDG News Service. “They did this by taking over the Internet domains that Conficker is programmed to visit to search for those instructions. . . . To complicate matters further, a new variant of Conficker was spotted last week, and this

13. C. Schiller et al, *InfoSecurity 2008 Threat Analysis* (Burlington, MA: Elsevier, 2008), 19.

14. Dan Goodin, “FBI Logs its Millionth Zombie Address,” *The Register*, June 13, 2007, http://www.theregister.co.uk/2007/06/13/millionth_botnet_address/.

15. Phillip Porras, Hassen Saidi, Vinod Yegneswaran, “An Analysis of Conficker C,” *SRI International*, March 8, 2009, http://mtc.sri.com/Conficker/addendumC/.

16. Robert McMillian, “Experts Bicker Over Conficker Numbers,” *Techworld*, April 15, 2009, http://news.techworld.com/security/114307/experts-bicker-over-conficker-numbers/.

one communicates primarily using peer-to-peer techniques, which are not easily measured by the Working Group's sinkhole servers."

12.1.3.5 Peer-to-Peer C&C

The Storm worm took distributed command-and-control to new extremes. Designed from the start with a distributed, multilayer model, Storm incorporated the Overnet/eDonkey peer-to-peer filesharing protocol for distributed, dynamic C&C between endpoint nodes. Joe Stewart, Director of Malware Research at Dell SecureWorks, wrote, "When Storm Worm runs, it attempts to link up with other infected hosts via peer-to-peer networking. Through this conduit it gets a URL that points to a second-stage executable, which in turn downloads additional stages onto the infected system. The protocol in use is actually the eDonkey/Overnet protocol, which has been adapted by the virus author as a means to distribute the second-stage URL without being shut down as it might be if the URL was hard-coded in the body of the virus or was downloaded from another website."[17, 18]

In short, the Storm botnet uses a peer-to-peer network of compromised hosts to dynamically communicate and change the locations of higher-layer C&C nodes, undercutting the effectiveness of network flow analysis. Botherders can distribute locations of many C&C servers and change them as necessary. Bruce Schneier writes, "[E]ven if a C&C node is taken down, the system doesn't suffer. Like a hydra with many heads, Storm's C&C structure is distributed."[19]

12.1.4 Automatic Self-Updates

Malware that survives is malware that can adapt. During the late 1990s, malware emerged that was designed to automatically update its own code over the network, allowing attackers to arbitrarily change propagation strageties, payloads, and other behavior on the fly. This was an enormous leap forward. Before this time, an attacker had to reinfect or manually update systems in order to spread new code. Automatic self-updates enabled attackers to easily adapt and maintain their footholds on compromised systems by improving code and quickly distributing it to a large number of compromised systems.

12.1.4.1 Early Self-Updating Systems

The first self-updating systems were very simple. In 1999, the W95/Babylonia self-mailer worm automatically checked a web site for updates after infection. This functionality was completely destroyed when authorities disabled the web site.[20]

17. Joe Stewart, "Storm Worm DDoS Attack", *Dell SecureWorks*, February 8, 2007, http://www.secureworks.com/research/threats/storm-worm/.

18. Joe Stewart, "Inside the Storm: Protocols and Encryption of the Storm Botnet" (SecureWorks, 2008), http://www.blackhat.com/presentations/bh-usa-08/Stewart/BH_US_08_Stewart_Protocols_of_the_Storm.pdf.

19. Bruce Schneier, "Gathering 'Storm' Superworm Poses Grave Threat to PC Nets," *Wired*, October 4, 2007, http://www.wired.com/politics/security/commentary/securitymatters/2007/10/securitymatters_1004.

20. Peter Szor, *The Art of Computer Virus Research and Defense*, 345.

12.1.4.2 Authenticated Updates

More sophisticated update systems began to emerge the following year. For example, in late 2000, the W95/Hybris worm was released. It was a global collaborative effort that included experienced malware authors from several countries. The W95/Hybris worm was designed to check for updates from a web site and newsgroups. The designers realized that without any sort of authentication system, defenders would be able to publish their own updates and disable the worm network. To protect against this, the W95/Hybris worm used RSA public-key encryption and a 128-bit hash algorithm. The attackers distributed a public key with the virus, and then cryptographically signed updates with the corresponding private key. Before installing updates, the compromised systems checked the integrity and authenticity of the updates.[21]

In 2004, the authors of the widespread MyDoom self-mailer worm learned the importance of authenticating updates. Once a system is compromised, MyDoom opens a backdoor on a TCP port and listens for connections. Attackers can connect to the backdoor and upload and execute arbitrary files.[22] Backdoors of this type can allow worms to automatically distribute updated software. However, in the case of MyDoom, other malware authors took advantage of the backdoor and used it to spread their own worms (such as "W32/Doomjuice") to systems previously infected with early versions of MyDoom.[23]

12.1.4.3 Going Meta: Updating the Updating System

Modern malware uses automatic self-updates to distribute changes not just in propagation mechanisms and payloads, but also the command-and-control system itself. For example, the Waledac (late 2008) command-and-control network is a distributed hierarchy that automatically self-updates over HTTP. This includes a layer of reverse proxy systems (referred to by researchers as "TSL" servers, after a Windows registry entry used by the malware to store server locations). "Repeater nodes obtain the list of TSL servers' IP addresses simply by asking a fellow repeater node for the current TSL server list," wrote researchers Sinclar, Nunnery, and Kang of the University of North Carolina at Charlotte. "[T]he author(s) of Waledac sign TSL IP updates using a RSA signature. The signature uses a private key held by the attacker(s) to sign the entire payload of the TSL update (excluding the signature portion, of course). Each Waledac binary contains a copy of the public key in order to verify that unauthorized entities have not altered the contents of the TSL update. If Waledac detects that the signature does not match the calculated signature value, the binary discards the TSL IP update. The use of a public/private key pair to sign the TSL update prevents defenders from inserting themselves into the TSL tier as well as prevents defenders from disrupting the upper tier communication."[24]

21. Peter Szor, *The Art of Computer Virus Research and Defense*, 347.

22. "Threat Description: Worm:W32/Mydoom," *F-Secure*, 2009, http://www.f-secure.com/v-descs/novarg.shtml.

23. Peter Szor, *The Art of Computer Virus Research and Defense*, 351.

24. G. Sinclair, C. Nunnery, and B.B.H. Kang, "The Waledac Protocol: The How and Why," in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference*, 2010, 6977, http://isr.uncc.edu/paper/WaledacProtocolHowWhyMalware2009_remediation.pdf.

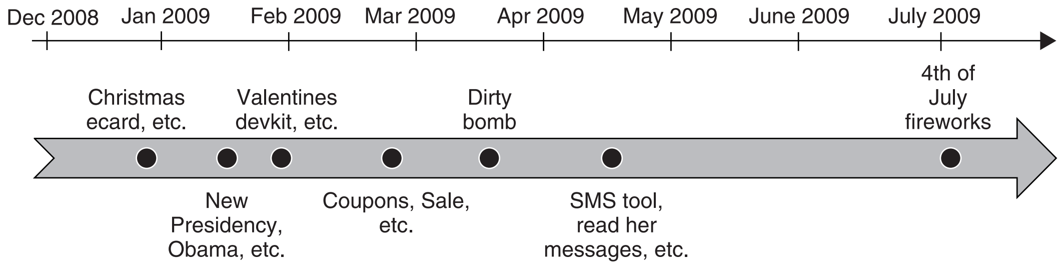


Figure 12–2. Symantec's timeline of the Waledac worm's propagation mechanisms. The Waledac worm uses automatic self-update mechanisms to routinely distribute new spam templates. Reprinted with permission.[25]

12.1.4.4 Success and Failure

Waledac's automatic self-update system has been an integral component to its success. Waledac spreads through email, and also uses email as a vector to advertise products. Often, the topic of the spam emails is related to timely holidays or events, such as Christmas or Valentine's Day, as shown in Figure 12–2. Frequent updates helped make Waledac's social engineering tactics very effective.

Waledac's automatic self-update system has also been key to its demise. Although Waledac cryptographically signs lists of "TSL" servers, the lower-layer peer-to-peer networks of compromised Waledac systems do not cryptographically validate lists of peer IP addresses exchanged through the network. Eventually, researches were able to infiltrate lower layers of the Waledac network by setting up their own fake Waledac bots. The IP addresses of the fake Waledac bots were propagated through the peer-to-peer exchange system.

As part of Microsoft's "Operation b49" coordinated takedown effort in early 2010, researchers used fake Waledac bots to "poison" the lower-layer peer-to-peer command-and-control system so that "all communication to the C&C infrastructure [was] redirected to [the Operation b49] infrastructure." This tactic, combined with an effective dismantling of the Waledac fast-flux network, cut off compromised endpoints so that they could not receive updates and commands through the Waledac C&C channels.[26] Although the endpoint nodes were still infected, once cut off from the command-and-control channel, they could no longer receive updates and were outside the control of the botherders.[27, 28]

25. Gilou Tenebro, "”Waledac: An Overview," *Symantec*, August 12, 2010, http://www.symantec.com/connect/blogs/waledac-overview.

26. Ben Stock, "If Only Botmasters Used Google Scholar ... The Takedown of the Waledac Botnet," March 10, 2011, http://www.enisa.europa.eu/act/res/botnets/workshop-presentations/ben-stock-presentation.

27. Nick Wingfield and Ben Worthen, "Microsoft Battles Cyber Criminals—WSJ.com," *Wall Street Journal*, February 26, 2010, http://online.wsj.com/article/SB10001424052748704240004575086523786147014.html?mod=WSJ_hps_sections_business.

28. Gilou Tenebro, "”Waledac: An Overview," *Symantec*, August 12, 2010, http://www.symantec.com/connect/blogs/waledac-overview.

12.4 Case Study: Ann's Aurora

The Case: *Ann Dercover is after SaucyCorp's Secret Sauce recipe. She's been trailing the lead developer, Vick Timmes, to figure out how she can remotely access SaucyCorp's servers. One night, while conducting reconnaissance, she sees him log into his laptop (10.10.10.70) and VPN into SaucyCorp's headquarters.*

Leveraging her connections with international hacking organizations, Ann obtains an exploit for Internet Explorer and launches a client-side spear phishing attack against Vick Timmes. Ann carefully crafts an email to Vick containing tips on how to improve secret sauce recipes and sends it. Seeing an opportunity that could get him that Vice President of Product Development title (and corner office) that he's been coveting, Vick clicks on the link. Ann is ready to strike ...

Meanwhile ... *Knowing that he is a high-value target, long ago Vick Timmes set up a traffic monitoring system on his home network. When suspicious activity is discovered relating to Vick's account at SaucyCorp, he provides investigators with packet captures so they can help him identify a compromise.*

Challenge: You are the forensic investigator. Your mission is to:

- Identify the source of the compromise.
- Recover malware from the packet capture and provide it to investigators for further analysis.

Network:

- Vick Timmes's internal computer: 10.10.10.70
- External host: 10.10.10.10 [Note that for the purposes of this case study, we are treating 10.10.10.70 as a system on "the Internet." In real life, this is a reserved nonroutable IP address space.]

Evidence: You are provided with one file containing data to analyze:

- **evidence-malware.pcap**—A packet capture (.pcap) file containing a small amount of network traffic from Vick Timmes's home network.

12.4.1 Analysis: Intrusion Detection

Since we're not quite sure what nastiness lurks in this packet capture, let's begin by running it through an IDS. For this case, we'll use Snort, configured with extra malware detection rules from SRI International's Malware Threat Center.[85]

85. "Most Effective Malware-Related Snort Signatures," *SRI International*, 2011, http://mtc.sri.com/live_data/signatures/.

```
$ snort -c /etc/snort/snort.conf -r evidence-malware.pcap
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
...
===============================================================================

Snort processed 2554 packets.
===============================================================================

Breakdown by protocol (includes rebuilt packets):
      ETH: 2554       (100.000%)
  ETHdisc: 0          (0.000%)
     VLAN: 0          (0.000%)
     IPV6: 0          (0.000%)
  IP6 EXT: 0          (0.000%)
  IP6opts: 0          (0.000%)
  IP6disc: 0          (0.000%)
      IP4: 2554       (100.000%)
  IP4disc: 0          (0.000%)
    TCP 6: 0          (0.000%)
    UDP 6: 0          (0.000%)
    ICMP6: 0          (0.000%)
  ICMP-IP: 0          (0.000%)
      TCP: 2553       (99.961%)
      UDP: 1          (0.039%)
     ICMP: 0          (0.000%)
  TCPdisc: 0          (0.000%)
  UDPdisc: 0          (0.000%)
  ICMPdis: 0          (0.000%)
     FRAG: 0          (0.000%)
   FRAG 6: 0          (0.000%)
      ARP: 0          (0.000%)
    EAPOL: 0          (0.000%)
  ETHLOOP: 0          (0.000%)
      IPX: 0          (0.000%)
    OTHER: 0          (0.000%)
  DISCARD: 0          (0.000%)
InvChkSum: 0          (0.000%)
   S5 G 1: 0          (0.000%)
   S5 G 2: 0          (0.000%)
    Total: 2554
===============================================================================

Action Stats:
ALERTS: 13
LOGGED: 13
PASSED: 0
===============================================================================

...
```

```
HTTP Inspect - encodings (Note: stream-reassembled packets included):
    POST methods:                   0
    GET methods:                    2
    Headers extracted:              2
    Header Cookies extracted:       0
    Post parameters extracted:      0
    Unicode:                        0
    Double unicode:                 0
    Non-ASCII representable:        0
    Base 36:                        0
    Directory traversals:           0
    Extra slashes ("//"):           0
    Self-referencing paths ("./"):  0
    Total packets processed:        1660
===============================================================================

dcerpc2 Preprocessor Statistics
  Total sessions: 0
===============================================================================

===============================================================================

Snort exiting
```

After running Snort on the packet capture, we check the alerts file, /var/log/alerts. Notably, Snort produced the following alerts, based on one of the Malware Threat Center's rules:

```
[**] [1:5001684:99] E3[rb] BotHunter Malware Windows executable (PE) sent
    from remote host [**]
[Priority: 0]
04/28-17:40:00.841061 10.10.10.10:4444 -> 10.10.10.70:1036
TCP TTL:64 TOS:0x0 ID:37696 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0xE31E89E1  Ack: 0x72ACC97B  Win: 0x16D0  TcpLen: 20

...

[**] [1:5001684:99] E3[rb] BotHunter Malware Windows executable (PE) sent
    from remote host [**]
[Priority: 0]
04/28-17:42:03.220699 10.10.10.10:4445 -> 10.10.10.70:1044
TCP TTL:64 TOS:0x0 ID:24030 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x559CF78D  Ack: 0x75FAD66D  Win: 0x16D0  TcpLen: 20
```

The rule that fired was as follows:

```
# -- Egg Download, Inbound: 5347 of 8211, from 12/23 to 05/31
alert tcp $EXTERNAL_NET !20 -> $HOME_NET any (msg:"E3[rb] BotHunter Malware
    Windows executable (PE) sent from remote host";  content: "MZ"; content: "
    PE|00 00|"; within:250; flow: established; sid:5001684; rev:99;)
```

This rule searches for the content "MZ" (the magic number of Windows executable files) and then "PE|00 00|" within 250 bytes of the first match. This indicates that the remote host sent a Windows executable (PE) file to a local system.

12.4.2 TCP Conversation: 10.10.10.10:4444–10.10.10.70:1036

Let's examine the first stream that triggered this alert to see if we can recover the executable file and gain information about its function. First, we list all TCP conversations in the packet capture, as shown below using tshark (the output has been edited to fit on the page). Notice that in this packet capture, there are 10 TCP conversations, all of which are between 10.10.10.10 (remote host) and Vick's computer, 10.10.10.70. One of these conversations relates to 10.10.10.10:4444 (remote host on TCP port 4444). Judging by the source and destination ports on both systems, this is the conversation that triggered the Snort alert above.

It is also worth noting that 8 of the 10 TCP conversations in this packet capture relate to 10.10.10.10:4445 (remote host on TCP port 4445). There is also one conversation relating to 10.10.10.10:8080 (remote host, TCP port 8080).

```
$ tshark -q -n -z conv,tcp -r evidence-malware.pcap
================================================================================

TCP Conversations
Filter:<No Filter>
                                    |      <-      |      ->      |
                                    |Frames  Bytes |Frames  Bytes |
10.10.10.10:4444 <-> 10.10.10.70:1036 424   120920  979  1293203
10.10.10.10:4445 <-> 10.10.10.70:1044 263    26876  664   869359
10.10.10.10:4445 <-> 10.10.10.70:1043  15      930   15      900
10.10.10.10:4445 <-> 10.10.10.70:1042  15      930   15      900
10.10.10.10:4445 <-> 10.10.10.70:1041  15      930   15      900
10.10.10.10:4445 <-> 10.10.10.70:1040  15      930   15      900
10.10.10.10:4445 <-> 10.10.10.70:1039  15      930   15      900
10.10.10.10:4445 <-> 10.10.10.70:1038  15      930   15      900
10.10.10.10:4445 <-> 10.10.10.70:1037  15      930   15      900
10.10.10.10:8080 <-> 10.10.10.70:1035   5      946    8     6463
```

Using tcpflow, let's reconstruct and save the first stream which triggered the Snort alert above, 10.10.10.10:4444<−>10.10.10.70:1036:

```
$ tcpflow -vr evidence-malware.pcap 'src host 10.10.10.10 and src port 4444
    and dst host 10.10.10.70 and dst port 1036'
tcpflow[23578]: tcpflow version 0.21 by Jeremy Elson <jelson@circlemud.org>
tcpflow[23578]: looking for handler for datalink type 1 for interface
    evidence-malware.pcap
tcpflow[23578]: found max FDs to be 16 using OPEN_MAX
tcpflow[23578]: 010.010.010.010.04444-010.010.010.070.01036: new flow
tcpflow[23578]: 010.010.010.010.04444-010.010.010.070.01036: opening new
    output file
```

12.4.2.1 File Carving: 10.10.10.10:4444–10.10.10.70:1036

Now, let's use the Foremost file carving tool to see if there are any files of interest in this TCP conversation:

```
$ foremost -T -i 010.010.010.070.01036-010.010.010.010.04444
$ cat /tmp/output_Fri_Jun__3_11_44_01_2011/audit.txt
Foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus
```

```
Audit File

Foremost started at Fri Jun  3 11:44:01 2011
Invocation: foremost -T -i 010.010.010.010.04444-010.010.010.070.01036
Output directory: /tmp/output_Fri_Jun__3_11_44_01_2011
Configuration file: /etc/foremost.conf
------------------------------------------------------------------
File: 010.010.010.010.04444-010.010.010.070.01036
Start: Fri Jun  3 11:44:01 2011
Length: 1 MB (1239098 bytes)

Num  Name (bs=512)         Size  File Offset   Comment

0:   00000000.dll       730 KB            4    04/03/2010 04:07:31
Finish: Fri Jun  3 11:44:01 2011

1 FILES EXTRACTED

exe:= 1
------------------------------------------------------------------

Foremost finished at Fri Jun  3 11:44:01 2011
```

It looks like Foremost carved out an executable file. We take the cryptographic checksums, as shown below:

```
$ md5sum 00000000.dll
b062cb8344cd3e296d8868fbef289c7c  00000000.dll

$ sha256sum 00000000.dll
14f489f20d7858d2e88fdfffb594a9e5f77f1333c7c479f6d3f1b48096d382fe  00000000.
    dll
```

Next, let's run this executable through an antivirus scanner, ClamAV, to see if it is known malware:[86]

```
$ clamscan 00000000.dll
00000000.dll: OK

----------- SCAN SUMMARY -----------
Known viruses: 970347
Engine version: 0.96.5
Scanned directories: 0
Scanned files: 1
Infected files: 0
Data scanned: 0.71 MB
Data read: 0.71 MB (ratio 1.01:1)
Time: 4.889 sec (0 m 4 s)
```

ClamAV did not recognize the file we carved as malware. However, it is possible that a different antivirus scanner might have a signature for it. Another option for the investigator

86. "Clam AntiVirus," 2011, http://www.clamav.net/.

is to upload the executable file to an online service such as VirusTotal, which "analyzes suspicious files and URLs and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines."[87] There are benefits and drawbacks to using a service such as VirusTotal. The benefits are that you get free analysis of any file you upload, and can take advantage of collective intelligence gathered from the global malware research community and multiple antivirus vendors. The drawback is that you must provide a copy of your suspicious file to a third party, which may raise issues of security and privacy.

For this case, we will upload our suspicious sample to VirusTotal. As you can see from the results below, 31 out of 43 antivirus vendors flagged the file as suspicious. There does not appear to be a clear consensus as to what the suspicious file is, precisely; it was variously identified as "TR/Swrort.A.964," "Trojan.Generic.4165076," "UnclassifiedMalware," and more.

Here are the results from VirusTotal:

```
File name: 00000000.dll
Submission date: 2011-06-03 17:40:49 (UTC)
Current status: finished
Result: 31/ 43 (72.1%)

Antivirus Version Last Update Result
AhnLab-V3 2011.06.04.00 2011.06.03  Win-Trojan/Xema.variant
AntiVir 7.11.9.3  2011.06.03  TR/Swrort.A.964
Antiy-AVL 2.0.3.7 2011.06.03  -
Avast 4.8.1351.0  2011.06.03  Win32:Hijack-GL
Avast5  5.0.677.0 2011.06.03  Win32:Hijack-GL
AVG 10.0.0.1190 2011.06.03  Generic2_c.ADPV
BitDefender 7.2 2011.06.03  Trojan.Generic.4165076
CAT-QuickHeal 11.00 2011.06.03  Trojan.Swrort.a
ClamAV  0.97.0.0  2011.06.03  -
Commtouch 5.3.2.6 2011.06.03  W32/MalwareS.BHOC
Comodo  8934  2011.06.03  UnclassifiedMalware
DrWeb 5.0.2.03300 2011.06.03  -
Emsisoft  5.1.0.5 2011.06.03  Virus.Win32.Hijack!IK
eSafe 7.0.17.0  2011.06.02  -
eTrust-Vet  36.1.8365 2011.06.03  -
F-Prot  4.6.2.117 2011.06.03  W32/MalwareS.BHOC
F-Secure  9.0.16440.0 2011.06.03  Trojan.Generic.4165076
Fortinet  4.2.257.0 2011.06.03  -
GData 22  2011.06.03  Trojan.Generic.4165076
Ikarus  T3.1.1.104.0  2011.06.03  Virus.Win32.Hijack
Jiangmin  13.0.900  2011.06.01  Trojan/Generic.cfa
K7AntiVirus 9.104.4763  2011.06.03  Riskware
Kaspersky 9.0.0.837 2011.06.03  -
McAfee  5.400.0.1158  2011.06.03  Generic.dx!spo
McAfee-GW-Edition 2010.1D 2011.06.03  Generic.dx!spo
Microsoft 1.6903  2011.06.03  Trojan:Win32/Swrort.A
NOD32 6177  2011.06.03  probably a variant of Win32/Agent.FZFSSSB
Norman  6.07.07 2011.06.03  W32/Suspicious_Gen2.ATQPC
nProtect  2011-06-03.02 2011.06.03  Trojan/W32.Agent.748032.U
Panda 10.0.3.5  2011.06.03  Trj/Downloader.MDWNDARY
PCTools 7.0.3.5 2011.06.03  Trojan.Gen
Prevx 3.0 2011.06.03  -
```

87. "VirusTotal—Free Online Virus, Malware and URL Scanner," 2011, http://www.virustotal.com/.

```
Rising  23.60.03.09 2011.06.03  -
Sophos  4.66.0  2011.06.03  Troj/Swrort-B
SUPERAntiSpyware  4.40.0.1006 2011.06.03  -
Symantec  20111.1.0.186 2011.06.03  Trojan.Gen
TheHacker 6.7.0.1.215 2011.06.02  -
TrendMicro  9.200.0.1012  2011.06.03  TROJ_GEN.R07C1DI
TrendMicro-HouseCall  9.200.0.1012  2011.06.03  TROJ_GEN.R07C1DI
VBA32 3.12.16.0 2011.06.03  Trojan.Win32.Rozena.gjd
VIPRE 9473  2011.06.03  Trojan.Win32.Generic!BT
ViRobot 2011.6.3.4494 2011.06.03  -
VirusBuster 14.0.66.0 2011.06.03  Trojan.Swrort!CpBH1zymeR4
Additional information
MD5   : b062cb8344cd3e296d8868fbef289c7c
SHA1  : b22683394afda7c3fa1d559169ce479c1fdad4f9
SHA256: 14f489f20d7858d2e88fdfffb594a9e5f77f1333c7c479f6d3f1b48096d382fe
...
File size : 748032 bytes
First seen: 2010-05-22 23:31:53
Last seen : 2011-06-03 17:40:49
TrID:
DOS Executable Generic (100.0%)
...
PEInfo: PE structure information

[[ basic data ]]
entrypointaddress: 0x627C5
timedatestamp....: 0x4BB6BF03 (Sat Apr 03 04:07:31 2010)
machinetype......: 0x14c (I386)

[[ 4 section(s) ]]
name, viradd, virsiz, rawdsiz, ntropy, md5
.text, 0x1000, 0x757E8, 0x75800, 6.68, d691fd422e760657489d7308f452d24a
.rdata, 0x77000, 0x25C6C, 0x25E00, 5.97, 69cdfb2a638cbc03d431ca94f329d42d
.data, 0x9D000, 0x166EC, 0x11400, 5.16, bb850f7bbadd797bf072f6a1c88a5ed4
.reloc, 0xB4000, 0x9AB2, 0x9C00, 5.82, 1722d6b7db1ed23d9b035a3184fca518
...
ThreatExpert:
http://www.threatexpert.com/report.aspx?md5=b062cb8344cd3e296d8868fbef289c7c
...
```

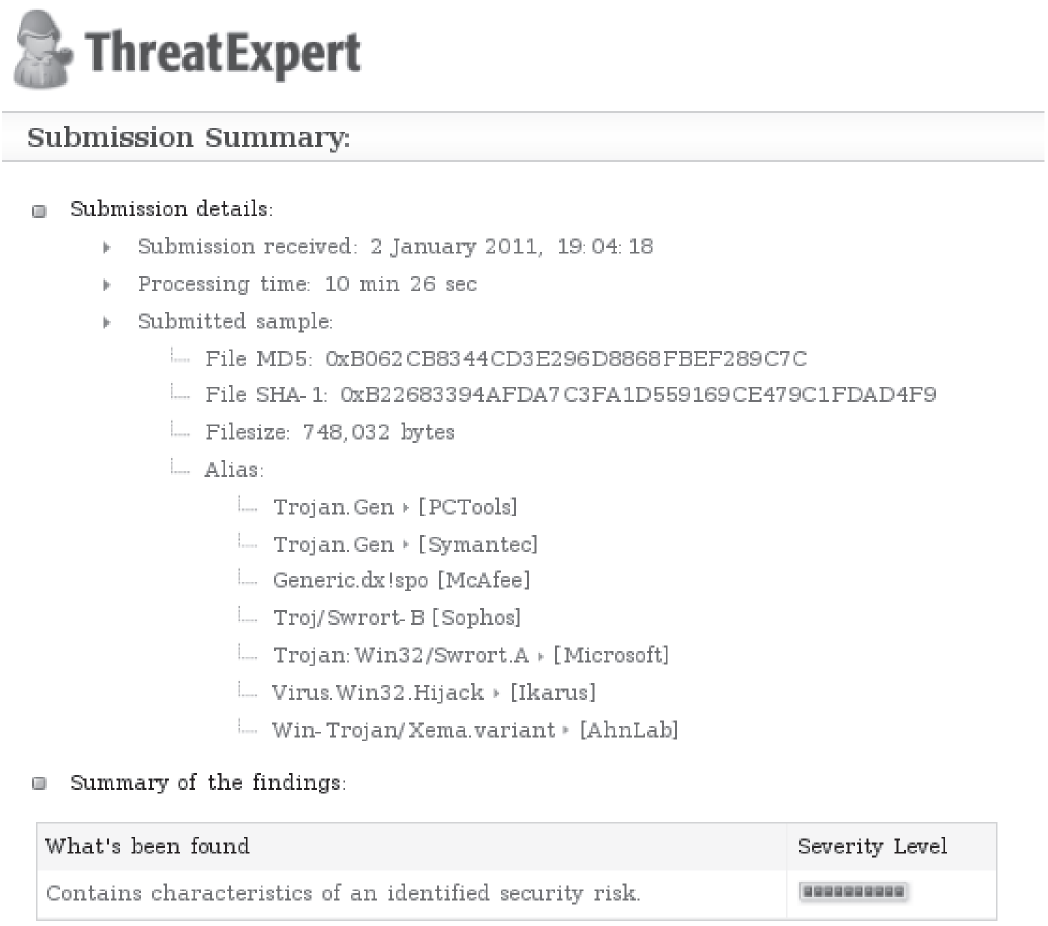
Notice that at the end of the VirusTotal report, there is another link to a "ThreatExpert" page, as shown in Figure 12–9. According to ThreatExpert, the executable is "A malicious trojan horse or bot that may represent security risk for the compromised system and/or its network environment."

Certainly worth saving for further analysis ...

12.4.2.2 Traffic Analysis: 10.10.10.10:4444–10.10.10.70:1036

Now let's take a closer look at the stream that we carved this file from, to see if we can gain any insight as to its function. Recall that we carved this executable file from the following TCP conversation:

```
10.10.10.10:4444     <-> 10.10.10.70:1036
```


ThreatExpert

Submission Summary:

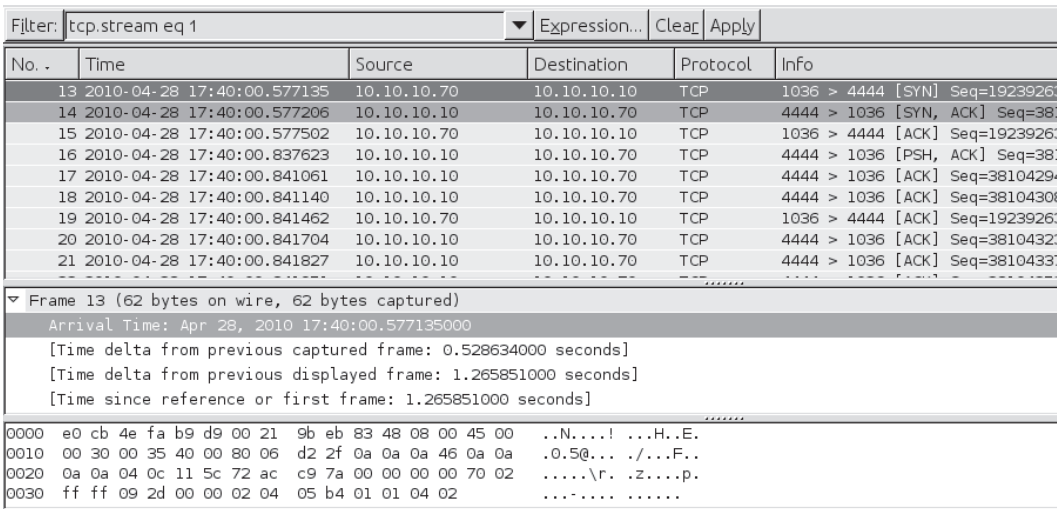
- Submission details:
  - Submission received: 2 January 2011, 19:04:18
  - Processing time: 10 min 26 sec
  - Submitted sample:
    - File MD5: 0xB062CB8344CD3E296D8868FBEF289C7C
    - File SHA-1: 0xB22683394AFDA7C3FA1D559169CE479C1FDAD4F9
    - Filesize: 748,032 bytes
    - Alias:
      - Trojan.Gen [PCTools]
      - Trojan.Gen [Symantec]
      - Generic.dx!spo [McAfee]
      - Troj/Swrort-B [Sophos]
      - Trojan:Win32/Swrort.A [Microsoft]
      - Virus.Win32.Hijack [Ikarus]
      - Win-Trojan/Xema.variant [AhnLab]
- Summary of the findings:

What's been found	Severity Level
Contains characteristics of an identified security risk.	▨▨▨▨▨▨▨▨▨▨

Figure 12–9. ThreatExpert's report on the executable file that we carved out.

Using Wireshark, we can filter on this conversation. In Figure 12–10, we can see that the conversation occurred on April 28, 2010. It began at 17:40:00.577135000 and completed at 17:41:26.898764000. As you can see in Figure 12–10(a), the first three packets show a complete TCP handshake: 10.10.10.70 (Vick's computer) sends a TCP SYN, the remote server (10.10.10.10) sends a TCP SYN/ACK, and then 10.10.10.70 sends an ACK, completing the handshake.

Using Wireshark's "Protocol Hierarchy Statistics" feature, we can see that the conversation contains only IPv4/TCP traffic and no higher-layer protocols were identified. See Figure 12–11 for details. By default, Wireshark decodes protocols based on known port numbers, so this may be caused by a higher-layer protocol running on an unusual port, or simply a protocol that is not known to Wireshark.



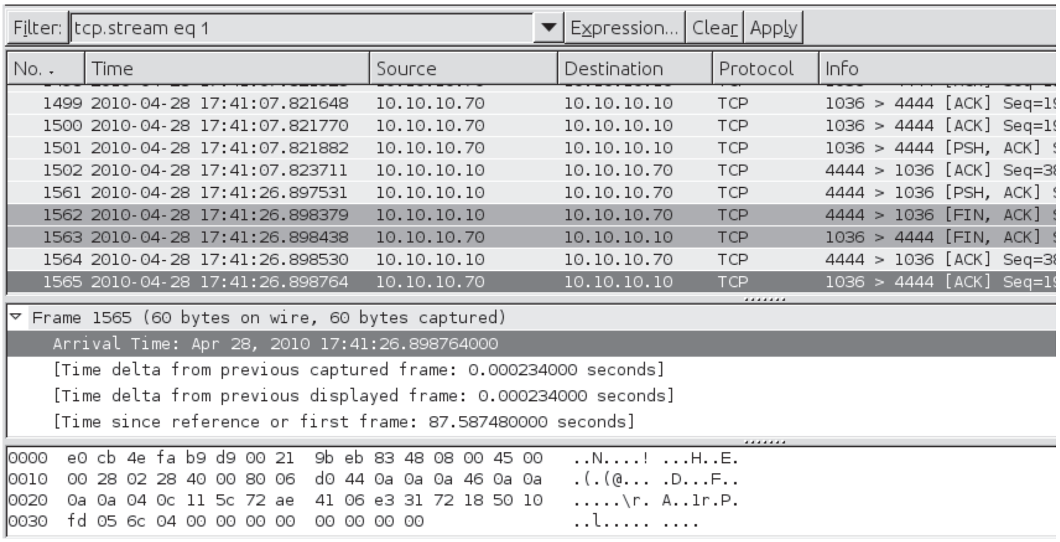
```
Filter: tcp.stream eq 1                    Expression... Clear Apply

No. .  Time                        Source        Destination   Protocol  Info
   13 2010-04-28 17:40:00.577135  10.10.10.70   10.10.10.10   TCP       1036 > 4444 [SYN] Seq=1923926
   14 2010-04-28 17:40:00.577206  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [SYN, ACK] Seq=38
   15 2010-04-28 17:40:00.577502  10.10.10.70   10.10.10.10   TCP       1036 > 4444 [ACK] Seq=1923926
   16 2010-04-28 17:40:00.837623  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [PSH, ACK] Seq=38
   17 2010-04-28 17:40:00.841061  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [ACK] Seq=3810429
   18 2010-04-28 17:40:00.841140  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [ACK] Seq=3810430
   19 2010-04-28 17:40:00.841462  10.10.10.70   10.10.10.10   TCP       1036 > 4444 [ACK] Seq=1923926
   20 2010-04-28 17:40:00.841704  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [ACK] Seq=3810432
   21 2010-04-28 17:40:00.841827  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [ACK] Seq=3810433

▽ Frame 13 (62 bytes on wire, 62 bytes captured)
    Arrival Time: Apr 28, 2010 17:40:00.577135000
    [Time delta from previous captured frame: 0.528634000 seconds]
    [Time delta from previous displayed frame: 1.265851000 seconds]
    [Time since reference or first frame: 1.265851000 seconds]

0000  e0 cb 4e fa b9 d9 00 21  9b eb 83 48 08 00 45 00   ..N....! ...H..E.
0010  00 30 00 35 40 00 80 06  d2 2f 0a 0a 0a 46 0a 0a   .0.5@... ./...F..
0020  0a 0a 04 0c 11 5c 72 ac  c9 7a 00 00 00 00 70 02   .....\r. .z....p.
0030  ff ff 09 2d 00 00 02 04  05 b4 01 01 04 02         ...-.... ......
```

(a)



```
Filter: tcp.stream eq 1                    Expression... Clear Apply

No. .  Time                        Source        Destination   Protocol  Info
 1499 2010-04-28 17:41:07.821648  10.10.10.70   10.10.10.10   TCP       1036 > 4444 [ACK] Seq=1
 1500 2010-04-28 17:41:07.821770  10.10.10.70   10.10.10.10   TCP       1036 > 4444 [ACK] Seq=1
 1501 2010-04-28 17:41:07.821882  10.10.10.70   10.10.10.10   TCP       1036 > 4444 [PSH, ACK]
 1502 2010-04-28 17:41:07.823711  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [ACK] Seq=3
 1561 2010-04-28 17:41:26.897531  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [PSH, ACK]
 1562 2010-04-28 17:41:26.898379  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [FIN, ACK]
 1563 2010-04-28 17:41:26.898438  10.10.10.70   10.10.10.10   TCP       1036 > 4444 [FIN, ACK]
 1564 2010-04-28 17:41:26.898530  10.10.10.10   10.10.10.70   TCP       4444 > 1036 [ACK] Seq=3
 1565 2010-04-28 17:41:26.898764  10.10.10.70   10.10.10.10   TCP       1036 > 4444 [ACK] Seq=1

▽ Frame 1565 (60 bytes on wire, 60 bytes captured)
    Arrival Time: Apr 28, 2010 17:41:26.898764000
    [Time delta from previous captured frame: 0.000234000 seconds]
    [Time delta from previous displayed frame: 0.000234000 seconds]
    [Time since reference or first frame: 87.587480000 seconds]

0000  e0 cb 4e fa b9 d9 00 21  9b eb 83 48 08 00 45 00   ..N....! ...H..E.
0010  00 28 02 28 40 00 80 06  d0 44 0a 0a 0a 46 0a 0a   .(.(@... .D...F..
0020  0a 0a 04 0c 11 5c 72 ae  41 06 e3 31 72 18 50 10   .....\r. A..1r.P.
0030  fd 05 6c 04 00 00 00 00  00 00 00 00               ..l..... ....
```

(b)

Figure 12–10. This screenshot shows the TCP conversation between 10.10.10.10:4444 and 10.10.10.70:1036 on April 28, 2010. The top image (a) illustrates that the conversation began at 17:40:00.577135000, while the bottom image (b) shows that it completed at 17:41:26.898764000.