



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at <http://www.giac.org/registration/gcia>

SMART IDS – HYBRID LABREA TARPIT

GIAC GCIA Gold Certification

Author: Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

Advisor: Chet Langin

Last Update: December 8, 2009

Abstract

The importance of IDS in corporate defense is seen as an ever growing necessity. Major strides have been made for numerous IDS tools, but some have seen a stalemate. The next evolutionary step in IDS would involve the concept of a 'Smart Intrusion Detection System (IDS)', one that generates signatures. The question of how to generate these signatures becomes instrumental, and can involve a number of different components. In this case, it could involve a tool that uses a hybrid LaBrea concept.

1. Introduction

The bad guys are always changing. It's part of their nature. Malicious users and clients are not necessarily in front of their screen running commands manually. Automated tools are written for a variety of attacks from specific application intrusion to self replicating and information gathering. Attacks in general, have a natural tendency to evolve and adapt, and through the manual intervention of other authors, variants are born. These variants can sometimes evade standard detection because they change just enough to not be recognized by anti-virus (AV) tools or sometimes requiring major changes in the AV signatures. These may also require changes in intrusion detection system (IDS) signatures and other signature based tools.

In this evolving landscape, security professionals are evolving their attack detection methodology. Using tools that specialize in correlating events collected from a number of sources, mainly, but not limited to, security tools. The logic used by these tools and appliances that allow for this correlation is based on the input from those analysts or some of the logic pre-installed by the vendors. Only what is understood as malicious can be alerted on, and because of this, brand new attacks can sometimes be missed.

A new tool from Nemean Networks is attempting to bridge the gap between static signature detection and unknown or unidentified attack profiles. Using a unique strategy comprised of different concepts, previously thought of as independent tools, they have put them together as complementary tools to each other in order to identify a number of threats. Some of the threats that exist people may not know about yet, specifically those that focus on the main protocols in use by the majority of the Internet's client base.

2. Honeypots

2.1. Introduction

Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

Honeypots found their roots in computer security in Stoll's 1990 book *The Cuckoo's Egg: Tracing a Spy through the Maze of Computer Espionage* (Stoll, 1990) and Cheswick's 1990 paper "An Evening with Berferd" (Cheswick, 1990). These two documents describe the interaction between systems and malicious clients, and how the systems and analysts were able to track down information based on the malicious clients' activities. Defining them concisely becomes a greater task than understanding their function. Lance Spitzner has two different definitions for honeypots. In his paper "Honeypots – Definitions and Value of Honeypots", published in December of 2002, he defines it as "a security resource whose value lies in being probed, attacked or compromised." (Spitzner, 2002) In May of 2003, his definition was updated to "an information system resource whose value lies in unauthorized or illicit use of that resource" (Spitzner, 2003) in his new paper of the same title, this time published on the site www.tracking-hackers.com, where the first one was published on www.windowsecurity.com. According to Spitzner, interest had been building throughout the late '90s and into the early 2000s. Honeynets simply expand the concept of a honeypot into an entire network. These can be extremely simple, such as a home network; or extremely complicated, near duplicates of production environments, without the use of production data.

2.2. Background and History

Where Lance Spitzner was writing about the concepts and citing examples in the early 2000s, Bill Cheswick had firsthand experience in implementing some of the earliest honeypots. To be honest, he had experience with just one honeypot... truth be told, he *was* the honeypot. He described his experiences from the late 1980s, from when he first noticed suspicious activity from a cracker, through the interactions between that same cracker and Cheswick pretending to be a vulnerable system, responding directly to the former and providing as many legitimate looking responses as possible. And before Bill Cheswick had the opportunity to do this, Cliff Stoll, a self proclaimed novice "computer jockey", was on the hunt for an ill-natured computer hacker, what is also called a "cracker", who had successfully compromised a few systems.

Cliff's story takes the reader from the world of a novice administrator, with background in other educational fields, to the world of espionage. Exciting and a great idea for a film! The difference is that this one would be "based on a true story." We often see those five words on films or novels exploring the extraordinary events that ordinary people lived through. As Spitzner mentions, this book does indeed cover some of the concepts of a honeypot. Cliff and his colleagues set up a new system, fully patched and with limited functionality other than watching a few other UNIX boxes on their network. Using this monitoring system, they meticulously watched the activity on their network and used what they learned to build a profile of the cracker they were tracing. The difference between what we now understand to be honeypots and what was happening in this situation, was the fact that the systems that were being probed and accessed were live production systems, not specifically put up to be probed. Through creativity and ingenuity, the team at Berkeley was able to begin monitoring for the persistent cracker and begin their in-depth analysis of what was going on in their network. Once they knew that they had been breached, Stoll, with a little help, was able to figure out what to look for and using printouts of the activity, was able to see exactly what the cracker was doing on the systems, all without alerting the overtly paranoid cracker to the fact that he had been watched during his activities.

Cheswick's security natured character in alerting a number of entities of suspicious activity allowed him to further his investigations beyond the limits of the network in which he worked. It also helped that he was working at AT&T Bell Laboratories, providing legitimacy of claims when reaching out to other big organizations, such as Stanford. Because of this pro-active approach to his reporting of analysis performed, he was able to find additional information such as the fact that other entities had been affected by the same cracker and that this turned out to be an international affair. He successfully fooled the cracker into believing he had achieved more than he truly had in terms of compromise. Using what direct interaction and the system he called "The Jail," he learned the crackers behavior, the way he interacted with systems and even the patterns in the cracker's command execution and was able to log in detail the traffic that

Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

was exchanged between the cracker and the supposed compromised system. The cracker's behavior was understood, a feeling of high certainty as to the cracker's location was had, but with a small return of investment on the time spent, versus the progress that could have been made on other efforts, Cheswick's "Jail" system was shut down.

Over the years, progress was made and the concepts evolved into systems and applications whose sole purpose was to respond to traffic that would not be legitimate in the network in which they reside. The depth to which these systems and applications respond to the attackers, automated or not, defines their status: low-interaction or high-interaction.

2.3. How They Work

Low-interaction honeypots emulate specific services and operating systems, but with limits (Provos, 2008). The service itself is not necessarily running, but the system is trained to respond to stimuli in ways that the service would in attempts to allow the unexpected traffic flow to continue as if it were established between the probable malicious client and a real server. **High-interaction** honeypots (Provos, 2003) can emulate an operating system in its entirety as well as the services it runs. Nowadays, these can be virtual machines as they can be easily duplicated as well as held in state through transport, which can be very useful in forensic analysis.

Low-interaction honeypots are limited in their coverage, but limited to a lesser extent in terms of their usefulness. By emulating specific services and/or characteristics of operating systems, they are confined to only responding to what they understand or expect to see. They can provide useful information as to the manner in which different automated attacks work on specific protocols as well as fooling the novice attacker, if that attacker is not well versed in the particular protocol he or she is attacking. By capturing the traffic and any potential binaries that may be dropped during the attempted attack, an analyst can start to see exactly what can be done to detect and/or begin to prevent the attack in other more important environments. An advantage to honeypots, in particular these low-interaction honeypots, is the small amount of logs and data that

would need to be reviewed in order to derive a production value in an environment. Instead of spending countless hours sifting through mountains of data, packet captures and differential reports on operating systems, the analyst can focus on the amount of data that comes from a low interaction honeypot and spend the rest of the time he or she would have spent on the other data sets doing something else for the organization or enjoying his or her personal time.

High-interaction honeypots provide a higher degree of deception to the attacker, but carry more risk associated to them. These types of honeypots typically aren't limited to emulating a couple of services, protocols or specific characteristics of an operating system. Instead, they are usually full systems: operating system in-tact and a list of patched or un-patched services running. The key is to fully deceive the attacker into believing they have achieved a breach and that their efforts can pay off. To further the game, high-interaction honeypot administrators can also provide dummy data to make the attacker believe that this is indeed a real system. An example of this can be seen briefly in the film *Untraceable* (2008), where an attacker jumps into a virtual machine that has been built out as a honeypot and files of seemingly high importance are stolen, such as an account and password list. The risk with high interaction honeypots is far greater than that of low-interaction, mainly due to the nature of what a honeypot is supposed to do and the high-interaction based nature involved here. Often, these systems can be used as a jump point, or a proxy for attacks or other malicious behavior, such as spamming. Protective measures should to be put in place to prevent this type of behavior to exist from the honeypot to limit the risk involved. The scalability of these systems has also been a concern for many individuals; resources may be tapped out when attempting to emulate larger networks. Attackers can also become more adept by detecting the presence of a honeypot and disengage before any useful data can be collected.

As is true for many industries, research is needed to advance and make progress. Honeypots have the advantage of being able to accommodate for both, production use and research use. Low-interaction honeypots have limited research use, but this is not the case for their production use, which is to say that they provide a greater service to

organizations to track potential illegitimate traffic on their networks to know how to improve other areas of their security measures to prevent that traffic from taking over their network. On the other hand, the nature of a high-interaction honeypot, the research capabilities are immense, because of the comprehensive nature of the information tracked in the case of probing or a full compromise. The skeletal operating system and service list allows for ease of analysis when a compromise of the honeypot system can be confirmed, usually done through log analysis or differential analysis between images of the system before and after the confirmed compromise.

3. LaBrea v1 and v2

3.1. Introduction to LaBrea

Example of a LaBrea conversation in human terms:

- Attacker: “John Doe... where do you live?”
- LaBrea: “Here’s my address!”
- Attacker: “I’m coming over”
- LaBrea: ...
- Attacker: “Hello?”
- LaBrea: ...

Meanwhile, the address given is non-existent. The address looks legit, as does the idea that this may indeed be the house for John Doe. And so the influx of junk mail begins. Offers ranging from credit cards with low rates, to coupons to the neighborhood grocery store addressed to John Doe directly, to envelopes with “respond immediately” in huge red letters. The lack of response is troubling, how is this solicitor going to get John to respond? The address is re-verified with the message, with the assumption of no spelling errors.

Next comes a similar attempt, only this time, asking Jon Doe, instead of John Doe. It seems that Jon and John live on the same street. John’s lack of response doesn’t deter the determined solicitor! Now Jon is not responding. Why won’t people respond to junk mail? Baffled, but not giving up yet, the solicitor keeps going through the phone book trying every Doe, spending a small amount of time at trying to get in touch with each. Unfortunately for the solicitor, the total accumulated time wasted accounts for a fair

percentage of the day. Finally, past the Does, the solicitor moves on to the next surname in the phonebook and keeps on his or her quest.

The understanding of this basic idea allows for the understanding of how the LaBrea Tarpit v1 works. Though it grossly over simplifies the actual process and removes some of the logical steps, it helps those unfamiliar with the concept get a good grasp of what the outcome is. It expands in its second version to allow for John or Jon to answer the requests, while telling the solicitor that they are busy, and that they will give them a quick call when they have found the time. The solicitor has by now found out that he or she can indeed get in touch with the Does and checks in every so often to see if they finally have time to chat.

3.2. Background and History

Tom Liston came up with the concept for the LaBrea Tarpit in response to CodeRed, which was released into the wild in 2001 (Liston, 2001). The idea was not the first, but probably one that would ensure that he not only kept his job, but also that he didn't go to jail. His other ideas might have worked great as responses to the worm, but as he says "most of them were illegal." Incidents.org provided Tom with the right forum to discuss the idea and to get feedback on it from other system and network administrators that were seeing similar issues with their network. From its humble beginnings as an Internet Protocol (IP) and Media Access Control (MAC) address spoofer through updates that provided IP and MAC address spoofing but added response spoofing with the ability to change the available window size to 0 allowing for the LaBrea system to keep the worm enthralled with the potential victim for an even longer time frame, this tool has served its purpose well and continues to do so. Because of what it can do, and how it does it, this tool can successfully slow down worms trolling for IP addresses if it is deployed properly.

3.3. How They Work

Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

Using the CodeRed example to explain the functionality, when a system becomes infected with the worm, the worm uses the system as a host whose sole purpose is to allow for the propagation of the worm. Searching for other potential hosts to infect, it begins attempting to find live hosts through the use of Address Resolution Protocol (ARP) requests of random IP addresses. The LaBrea system, listening for the traffic on the network, can tell with high precision whether or not the IP address is in use. If no response for a MAC address is caught by the system within a given time frame, a setting which can be modified, it responds with a phony MAC. This allows the host table of the infected host to update with the fake address association and also can prevent the router from returning an ICMP error message, which can be silenced within the settings of the router, though not always done. See screenshot below for a sample sequence of ARP requests and ARP reply.

```
An example (from a tcpdump of LaBrea running on my network):
• 14:18:28.832187 ARP who-has xxx.xxx.xxx.13 tell xxx.xxx.xxx.1
• 14:18:29.646402 ARP who-has xxx.xxx.xxx.13 tell xxx.xxx.xxx.1
• 14:18:31.707295 ARP who-has xxx.xxx.xxx.13 tell xxx.xxx.xxx.1
Key fingerprint = AE19-FA27-2F94-998D-FDB5-DE3D-E8B5-0
• 14:18:31.707374 ARP reply xxx.xxx.xxx.13 is-at 00:00:00:00:00:00
```

Figure 1 - Example shown on the LaBrea Tarptit introduction and background page: <http://labrea.sourceforge.net/Intro-History.html>

Once the infected host has cached the spoofed the MAC address, it will attempt a connection to the potential next link in the infection chain with a SYN packet to port 80, based on the nature of CodeRed in targeting IIS servers in search of vulnerable systems (Liston, 2001). At this point, the LaBrea system will send the standard response that would be elicited by the SYN packet with a SYN-ACK packet. The infected host will then continue to establish the connection by completing the three-way handshake. If the window size had been modified, the infected host will complete the three way handshake with the understanding that the window size is either smaller or larger than normal. Once that's complete, the infected host will proceed to send the packets it hopes will be reassembled on the other end of the established communication and allow for the system to become compromised. Without an ACK response to the sent packet, the infected host

attempts retransmissions based on the nature of the host. Normal nature seen indicates that retransmissions increase their time window between retransmission attempts, just in case it just hadn't waited long enough to receive the response. After a few retransmission attempts, it realizes that it will not receive a response and moves on. Instead of having jumped past the unused IP address, it has now wasted time establishing the connection and attempting to exploit a non-existent host.

This dramatically slowed the infection rate in networks that employed this innovative method for controlling the environment. On an explanation posted on the LaBrea site quotes a message from Mihnea Stoenescu sent to the distribution list discussing the concept: "For a few hours I've been teergrubing CodeReds via three-way handshake on behalf of an entire C-block, by using only one host. At a rate of 6 hosts per minute hitting my block, I'm consuming circa 15 minutes of effective attack time every minute. A lot of hosts can be scanned in 15 minutes."

4. HoneyD

4.1. Introduction to HoneyD and Background

Honeyd was developed by Niels Provos and intended to run on Unix systems (Provost, 2008). The ability to emulate a number of different operating systems and a number of different services provides a base for an active response model that allows for an in depth analysis of automated and some non-automated attempted exploits. In 2002, Honeyd was the only honeypot that was able to do this. Since version 0.7, Honeyd has also had the ability to perform tarpit functionality to aid in limiting the spread of malware.

4.2. How They Work

Similar to LaBrea, honeyd will spoof ARP responses in order to intercept and respond activity. This honeypot is extremely useful because of the ability to detect activity directed at any of the 65000+ ports. In addition to that, scripts can be written in order to emulate almost any service. With the ability to log details on the attempts, these systems

Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

can be extremely useful in detecting what is currently happening in the environment and also provide necessary details in order to create signatures for IDSs to alert on suspicious activity.

5. IDS

5.1. What is it?

Per PCMag, IDS has a very simple definition:

“(Intrusion Detection System) Software that detects an attack on a network or computer system. A Network IDS (NIDS) is designed to support multiple hosts, whereas a Host IDS (HIDS) is set up to detect illegal actions within the host. Most IDS programs typically use signatures of known cracker attempts to signal an alert. Others look for deviations of the normal routine as indications of an attack. Intrusion detection is very tricky. Too much analysis can add excessive overhead and also trigger false alarms. Insufficient analysis can overlook a valid attack.”

This definition of a Network IDS provides a high level overview of what IDS systems are designed to do. It is generally accepted that IDS devices are a major component of a successful security program. Yet they, in and of themselves, are not the holy grail of security but in fact work in conjunction with other devices and concepts to establish a more secure network environment.

Deploying IDS tools is only a part of the puzzle. Understanding the resources required from a personnel and system perspective are critical to a successful service. Even if open source tools are used, there are other resources that need to be taken into consideration. These include but may not be limited to funds supplied for the purchase of hardware used to serve as the IDS, the hardware itself, and engineering-hours that may be required to successfully configure the system and ongoing analysis and incident response hours for detected attacks. The associated overhead required may grow exponentially depending on the complexity of the network and the obscure nature of the tools used.

IDSs provide a great service to network administrators and security analysts, providing insight into the nature of the anomalous traffic that may travel through the said network.

Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

Signatures are designed to trigger alerts on traffic based on common traits seen for a number of attempted exploits or traffic patterns that are usually not seen. Something as simple as running specific protocols on a non-standard port could lead to an alert firing; the case is the same for something more complicated similar to a buffer overflow.

5.2. Snort (Open Source Intrusion Detection System)

Since its first appearance in 1998, courtesy of Martin Roesch, Snort has become the de facto open source IDS used because of its versatility, the ease of signature writing and the fact that it is released as an open source tool, allowing for anyone that would like to explore it, to download, install and run it. On their website, they have counted over three and a half million downloads along with over two hundred thousand registered users, making it “the most widely deployed IDS/IPS technology worldwide.” (snort.org, 2009) its popularity is based on the many advances it has made over the years to include the possibility for intrusion prevention. Snort has the capability of sitting in-line and dropping the packets identified as matching a signature, by sending reset packets to the client/server, or adding access-list entries to firewalls and routers. Though major advancements have been made, the risk of false positives often hinders IDS administrators from implementing prevention options.

5.3. Signatures and How They Work

Snort signatures are referred to as human readable, meaning the rule language can be understood. The simplicity of their signature language has allowed for the growth of Snort’s user base. Breaking down the signatures is easily explained (Matulis, 2002). The signatures are made of two main components: rule header, rule body. These provide for characteristic based signatures for these devices.

The rule header includes the action to be taken when the signature is triggered, whether alert or log, as well as the nature of the traffic, including protocol, source and destination IPs or networks and the ports involved.

The rule body includes the rule options, which define the alert message, the anomalies to look for in the packets. These anomalies can be located anywhere in the packet, including the protocol headers, the payloads and can also be connection based and dependent on the direction of the traffic as well, whether from client to server or vice-versa. See sample rule below:

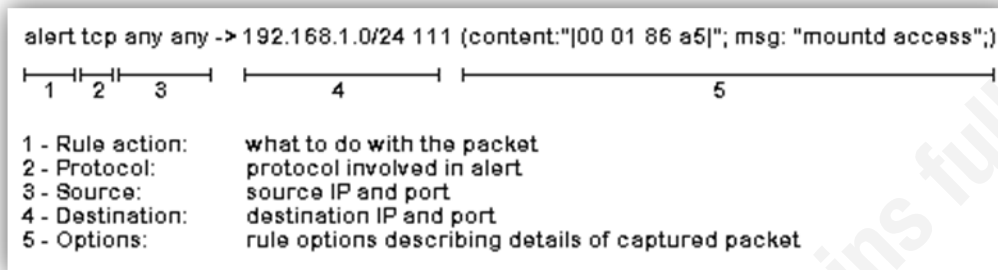


Figure 2 - Sample rule broken down. Sample rule borrowed from <http://www.linuxsecurity.com/content/view/132365/2/> (Keys, 2007)

5.4. The Man-Hours and Resources Needed

With thousands of rules available for Snort, it is ideal to have only the ones that apply to your network enabled. Having personally seen some of the time necessary, it is easy to say that this is no small feat. Often times, analysts begin by enabling all rules and cutting back on those that produce a large number of alerts per rule and a majority of those alerts being false positives, they disable them.

There are ways to improve this process. One of the keys to ensuring the process is run as efficiently and smoothly as possible is to understand the network environment in which these devices are deployed. With a key understanding of the environment, the analyst is able to rule out signatures that can cause a large number of false positives and focus on rules that will trigger on what can be considered legitimate attempts to circumvent security measures and breach the network and the systems in it.

Operating system specific signatures often provide a great way to monitor for attacks designed to target certain characteristics of the applications and or implementations of the TCP/IP stack for those operating systems. These signatures can be noisy when implemented. If implemented in an environment where the residing systems are strictly

of a different operating system, they may create a large number of alerts that lead to investigations where time can be wasted.

For all of these reasons, in addition to others, resources in terms of time can be consumed and the hours can add up rather quickly. Fighting the daily fires of suspicious network activity can occupy multiple head count and often limit resources further because of budgetary constraints.

6. Mixing it all up

All of these aforementioned concepts and tools are immensely useful and some are key elements to a successful security program, whether for research or for corporations and other type of organizations. A number of organizations have transitioned from using multiple vendors to focusing on a single vendor to aid in ease of management of systems as well as troubleshooting. This would alleviate the need to have individuals versed in multiple systems or have multiple individuals, each with their own area of expertise. The reduction of man hours their related cost can be vital in ensuring the survival of the program as well as buy in from management by showing fiscal responsibility when budgeting time comes around. Imagining teams that manage routers, switches and other network devices from the same company is extremely easy. Firewall teams that have a comprehensive understanding of the technology but have perfected their methodology for management and syntax use because of specific firewall types being used are often more proficient in times of expedited incident or network issue resolution.

Centralizing tools has been progressing over time, leading to security information and event management systems. Often shortened to “SIEM” or “SIM”, these devices allow for a central location for logs to be stored. This centralized repository allows for correlation of logs from multiple sources: firewalls, routers, servers of many kinds, IDS and others. Parsing the logs from these very diverse devices consumes resources on SIEM, and some allow custom parsing to occur when logs are received and interpreted, these devices are designed to do this. The correlation of logs with the use of specific customized rules that trigger when a combination of logs can be used to indicate potential security alerts that may necessitate further investigation.

Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

Following the evolution of both of these ideas can lead to a centralized security device that incorporates more than a single measure in the same tool. Security analyst and security administrators often seek a method to the madness and for ways to eliminate false negatives and false positives, something that becomes a never ending race to attempt to get in front of the malicious intended users. The question becomes what currently widespread and used concepts would make the biggest impact and lead to the most evolutionary progress that can be developed with these current tools.

6.1. LaBrea Concept

Tom Liston made came up with an innovative concept in response to a rampant spread of malicious code. The LaBrea Tarpit concept and tool that was first seen in 2001 as a response to CodeRed revolutionized how people thought about mitigation. Though this did not eradicate the threat, it allowed network administrators and security analysts to identify the threat and mitigate it in a more rapid timeline than would have been possible before the idea was put together. Fooling infected systems into attempts to infect non-existent hosts while having the potential to alert on these activities allowed other hosts on the network, real and production hosts, to continue to function undisturbed for just a bit longer.

6.2. Logical Evolution – Honeypot Concept

LaBrea allowed for three-handshakes to complete, or at least make the infected host believe that it completed. Beyond that, it did not do much else in its first revision. Honeyd's configurability allows for a quick and easy honeypot to be set up and respond to more in depth stimuli than the tarpit concept. With a typically preprogrammed IP address, there are limits to what Honeyd can do. Monitoring these Honeyd deployments can be extremely useful in understanding the potential threats to protocols that are often available on networks. The problem is being able to see what can't be seen, understand what can't be understood. An infected host may randomize the IP addresses it probes, it

may jump over the IP address used for the Honeyd deployment. If a honeynet is used this may reduce the odds of missing the probing, but it still exists.

6.3. How They Mix

Intermixing these two concepts together, there is a potential for a honeynet/tarpit of sorts. Infected systems would be able to probe a number of IP addresses as well as attempt to infect those IP addresses, similar to what was seen with LaBrea and CodeRed, but unlike that case, with the honeynet component, there would be responses to the probes and attempted distribution of malware. With the ability to respond, the hybrid honeynet would be able to provide a deeper understanding as to how some of the vulnerabilities are exploited.

Not all networks are created equal! With that in mind, a honeynet can be easily discovered if there are too many systems of a specific type, or not enough of another. Let's say that a malicious user has targeted a major computer producer, Apple. If that user is attempting to compromise Apple's internet facing environment and has vulnerabilities specific to that operating system but only sees Windows 2003 and 2008 servers, the attacker may detect that the network is not truly a production environment, and stop the attack. Though in the end, the end goal was achieved, more reconnaissance could be performed to find a better set of potential systems to attack. The goal would be to provide something as realistic as possible. The ideal hybrid would be able to mix and match server models and provide appropriate responses for the various types of systems that would be found in a typical production environment.

While these two systems together can be a good combination, they would be mainly targeted for use in research efforts. Logging the activity would lead to better understanding of how attacks are formulated, or malware is acting in the wild. For use in corporate environments, there would need to be an additional component, the network monitoring tool that used the data from the hybrid concept to determine whether or not legitimate hosts have been afflicted. This would allow security analysts to be informed of the potentially dangerous traffic that may reside on their network.

While the hybridization of these three concepts can serve for the discovery of malicious traffic, it can also be used to analyze other potentially unexpected traffic, including incorrectly configured systems. This is an added benefit that would be determined after the initial analysis of the flagged traffic was completed.

7. Putting it all together

7.1. Background

Nemean Networks has developed a hybrid system that incorporates a series of honeypots, usually in the form of CIDR blocks, with an intrusion detection system, IDS (Nemean Networks, 2009). Developed over a series of years and a lot of research performed at the University of Wisconsin, the system uses what it learns from activity with its honeypot CIDR block to generate custom signatures that are applied to the IDS component to alert on activity similar to what is seen on the honeypot IP addresses.

Paul Barford, along with a team of fellow researchers from the University of Wisconsin, developed a number of analytical tools based off of new ideas about how to monitor and interpret network activity in hopes of being able to understand the nature of malware that was making its way into the university environment. The evolution from research to production tools may have taken a while since the research was done, but has allowed for an innovative tool, especially in times where automation is becoming more of a business need, rather than a simple ‘nice to have’. Reading and understanding the research helps understand the concept of what is going on within the system, but understanding the concepts of IDS and honeypots and honeynets is also important as they make the overall concept of Nemean easy to understand and can make people wonder why something like this hasn’t been done before. Paul Barford’s research at the University of Wisconsin, followed by the years since and the creation of Nemean Networks, of which he is CEO, makes it clear that this is no easy undertaking.

7.2. Honeypot Portion

7.2.1. Protocols and Responses

The honeypot CIDR block is designed to respond to unused IP addresses, configurable to vary between single IP addresses and large CIDR blocks. There are a small set of protocols that are understood and responded to appropriately by the honeypots. These are common protocols and the ones usually used to spread malware, which is one of the reasons for which these are the more developed protocols with some seemingly ‘high interaction’ nature as well as some ‘low interaction’ nature, depending on the protocol.

With the advent of the internet and the concept of security being on the back of most end-users’ minds, the more common protocols in use across the internet nowadays are HTTP, SMTP and Peer-to-Peer, P2P. Lesser used protocols on the internet include FTP and its secure versions. On local networks, users are becoming more addicted to the thought of instant access to their files and media, even if on different systems, leading to common Microsoft network sharing protocols being used.

For most of the protocols mentioned, interaction is possible with the honeypots in this system. The interaction level varies on the protocol, both for reasons of development decisions as to which protocols to develop first and for the statistical chance that those protocols may be interacted with in relation to the others that were developed. Performing a number of tests on the available protocols, the results below can be obtained.

7.2.2. Depth of Responses and Honeynet

HTTP

The protocols involved respond at different levels of depth. The more developed protocols include HTTP because of the extremely common nature of its use and its use for malware distribution. A comparison between two sessions’ screenshots is shown below. We can see the honeypot HTTP responses and what they provide, as well as the response from a real “blank” site, where no content is held in the HTML code.

The screenshots below show different outputs from the honeynet to basic browsing. These may be just a few IP addresses apart, and show a number of different outputs, but all originate from the same device. Packet captures on the Nemean device would show the interface of the various servers to be the same and therefore have the same MAC address; unfortunately, performing these packet captures on the exploiting system would not be effective to prove that point.

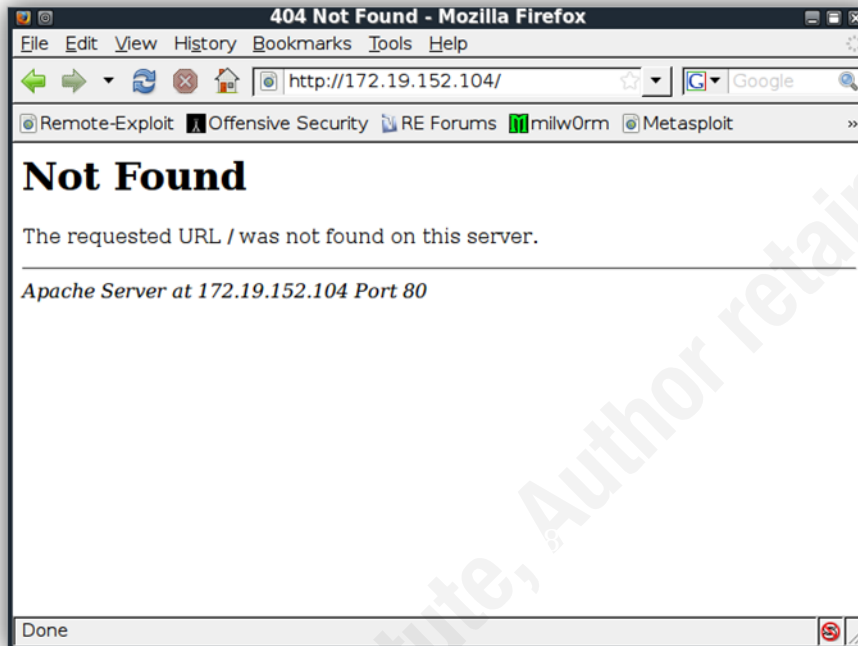


Figure 2 - HTTP 404 Error because no index file was found.

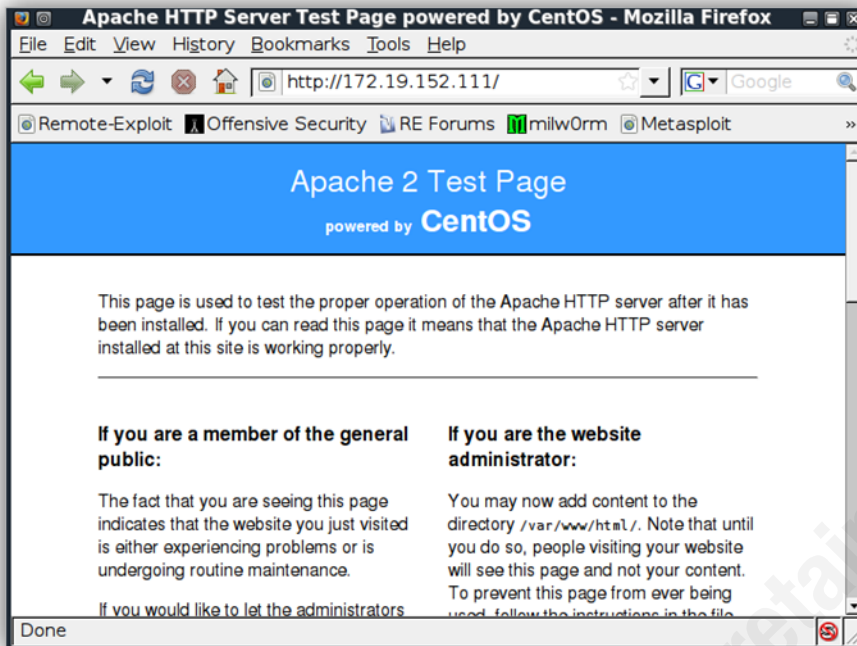


Figure 3 - CentOS with an HTTP server running using a default splash page.

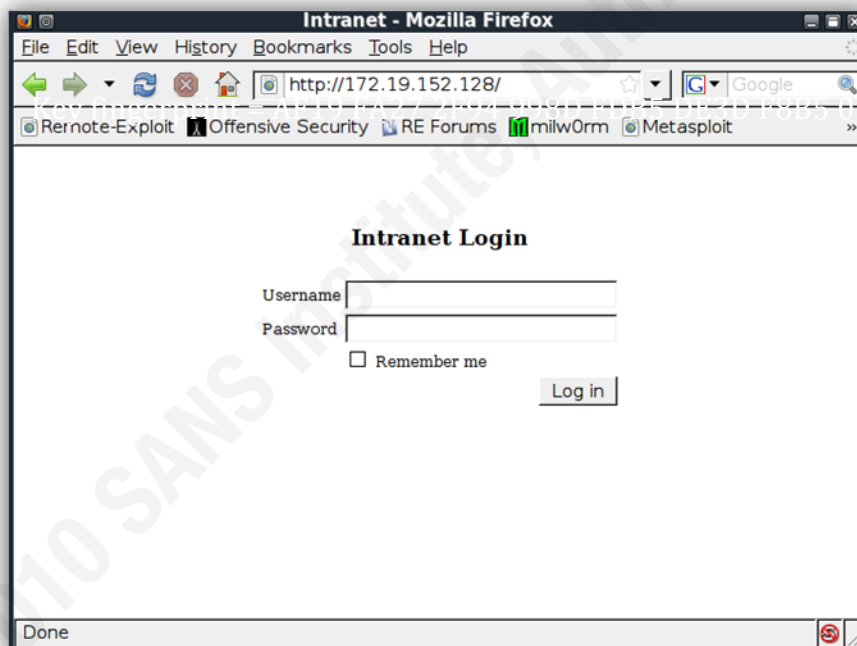


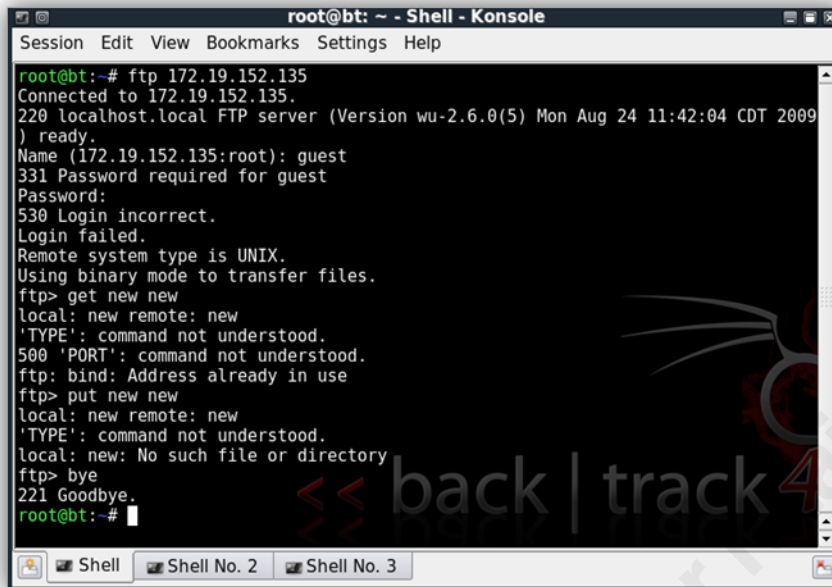
Figure 4 - An Intranet login page asking users to log in to view the site.

FTP

Another protocol that is well covered is FTP. Packet captures below show a few commands that were executed as well as the response from the honeypot FTP server and the real FTP server. This protocol is well emulated also, but not as well as the HTTP

Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

protocol. There are a few specific commands and responses that do not respond as would be expected from a real FTP server.



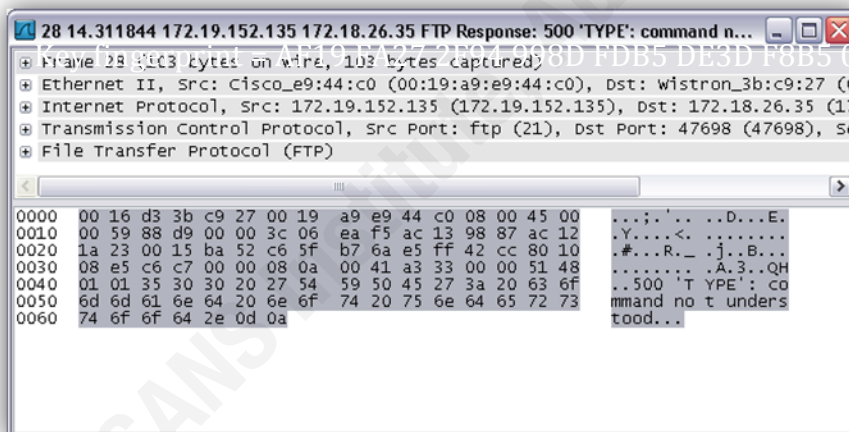
```

root@bt: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

root@bt:~# ftp 172.19.152.135
Connected to 172.19.152.135.
220 localhost.local FTP server (Version wu-2.6.0(5) Mon Aug 24 11:42:04 CDT 2009
) ready.
Name (172.19.152.135:root): guest
331 Password required for guest
Password:
530 Login incorrect.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get new new
local: new remote: new
'TYPE': command not understood.
500 'PORT': command not understood.
ftp> bind: Address already in use
ftp> put new new
local: new remote: new
'TYPE': command not understood.
local: new: No such file or directory
ftp> bye
221 Goodbye.
root@bt:~#

```

Figure 5 - FTP served from the Nemean honeynet.



```

28 14.311844 172.19.152.135 172.18.26.35 FTP Response: 500 'TYPE': command n...
Frame 28: 103 bytes on wire (824 bytes captured) on interface 0
Ethernet II, Src: Cisco_e9:44:c0 (00:19:a9:e9:44:c0), Dst: Wistron_3b:c9:27 (00:19:1a:3d:f8b5:06)
Internet Protocol, Src: 172.19.152.135 (172.19.152.135), Dst: 172.18.26.35 (172.18.26.35)
Transmission Control Protocol, Src Port: ftp (21), Dst Port: 47698 (47698), Seq: 172182635, Win: 0, Len: 0
File Transfer Protocol (FTP)
0000  00 16 d3 3b c9 27 00 19 a9 e9 44 c0 08 00 45 00  ...:..D..E.
0010  00 59 88 d9 00 00 3c 06 ea f5 ac 13 98 87 ac 12  .Y...<.....
0020  1a 23 00 15 ba 52 c6 5f b7 6a e5 ff 42 cc 80 10  .#...R...j..B...
0030  08 e5 c6 c7 00 00 08 0a 00 41 a3 33 00 00 51 48  .....A.3..QH
0040  01 01 35 30 30 20 27 54 59 50 45 27 3a 20 63 6f  ..500'TYPE':co
0050  6d 6d 61 6e 64 20 6e 6f 74 20 75 6e 64 65 72 73  mmand no t unders
0060  74 6f 6f 64 2e 0d 0a                                tood...

```

Figure 6 - FTP error to attempted unauthenticated file transfer.

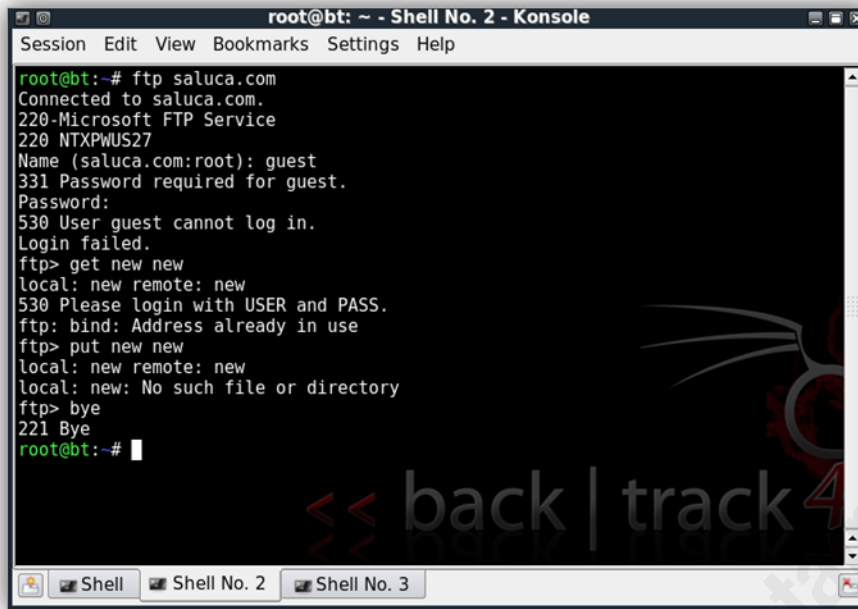


Figure 7 - FTP screenshot from an actual FTP server.

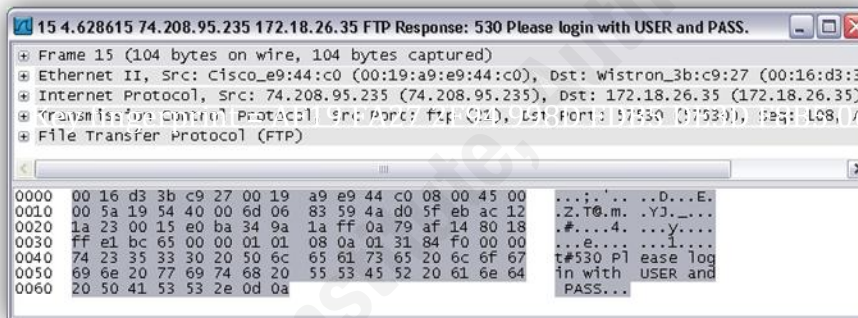


Figure 8 - FTP error and log in request to attempted unauthenticated file transfer.

As seen in the packet captures and their comparisons, we can see the difference between what the real server is responding to specific errors and attempts at file transfers and what the emulated server is responding. Because this type of protocol is not generally as widely used by end-users, the subtle differences may not be readily noticed.

SMTP

SMTP is also emulated. This protocol may not be as developed as the two mentioned before. There are certain commands that are accepted and properly interacted. The server emulated is an open SMTP server, allowing for connections to be initiated and some of the CLI commands to be accepted and properly responded to. Looking at the packet

captures and the screenshots, we can see that there are limits to what the emulated service is currently configured to understand.

7.2.3. Automated/Scripted vs. Manual Interaction

These protocols respond to both automated and manual interaction. Packet captures from both manual and automated packet captures are shown below. The automated interaction is from HTTPrint. The report from the tool shows that a number of different server operating systems that are emulated, as well as the HTTP server type. A sample of potential combinations is listed below.

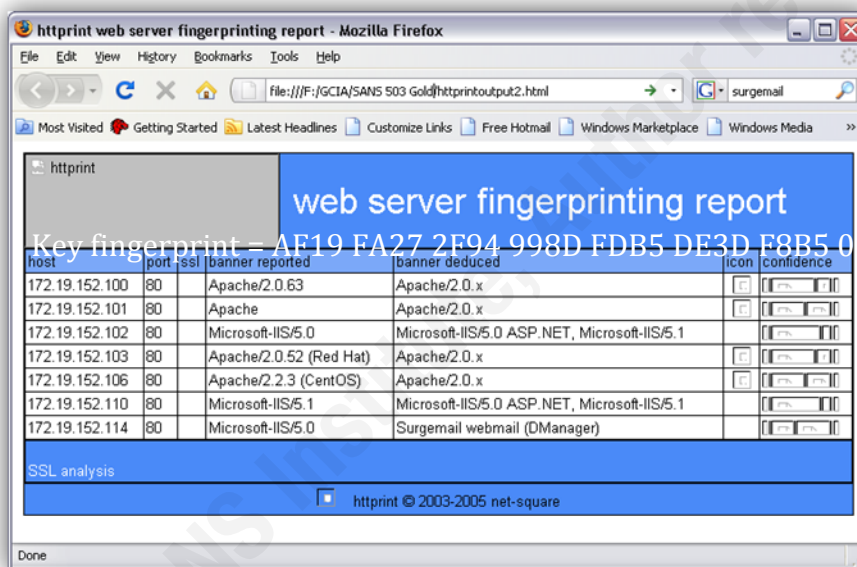


Figure 9 - HTTPrint results for a few IPs of the honeynet.

An Nmap scan of a subset of IP addresses listened for by the system show a variety of different operating systems and services available for interaction. Packet captures show which particular responses indicated which operating system was responding, allowing to mix with the different services that were available. This permits the individual running the scan to see the variety of possibilities that exist with this tool.


```

<host starttime="1250086671" endtime="1250086824">
<status state="up" reason="echo-reply"/>
<address addr="172.19.152.104" addrtype="ipv4" />
<hostnames />
<ports>
<extraports state="filtered" count="985"><extrareasons reason="no-responses" count="985"/></extraports>
<port protocol="tcp" portid="21"><state state="open" reason="syn-ack" reason_ttl="60"/><service name="ftp" method="table" conf="3"
/></port>
<port protocol="tcp" portid="25"><state state="open" reason="syn-ack" reason_ttl="60"/><service name="smtp" method="table" conf="3"
/></port>
<port protocol="tcp" portid="80"><state state="open" reason="syn-ack" reason_ttl="60"/><service name="http" method="table" conf="3"
/></port>
<port protocol="tcp" portid="81"><state state="open" reason="syn-ack" reason_ttl="60"/><service name="hosts2-ns" method="table"
conf="3" /></port>
<port protocol="tcp" portid="135"><state state="open" reason="syn-ack" reason_ttl="60"/><service name="msrpc" method="table" conf="3"
/></port>
...
</ports>
<os><portused state="open" proto="tcp" portid="21" /></os>
<tcpsequence index="187" difficulty="Good luck!" values="2FCD9121,30279C50,3072A5A1,30C7B02F,313C6958,319ECAE5" />
<ipidsequence class="Busy server or unknown class" values="88BE,88CF,88DE,88F0,8907,891A" /><tcptssequence class="unknown class"
values="22864D,22864D,22864D,22864D,22864D,22864D" />
<times srtt="6296" rttvar="1004" to="100000" />
</host>

```

Figure 10 - Abridged host sample from Nmap scan


Manual interaction yields expected results. Previous screenshots show that the protocols respond accordingly, at the very least to the initial session setup. Some of the protocols follow through extremely well in manual interaction, especially the HTTP protocol; on the other hand, some provide some subtle incorrect responses. This is seen in particular with the FTP protocol. In the screenshots, the FTP session is responding to an attempted file transfer from a non-authenticated user, throwing error messages. These error messages show the error message “TYPE: command not understood”. While an actual FTP server is responding to similar attempts, but the error message is different, specifically mentioning the fact that the user is not successfully logged in, an error message that can be expected. Different types of FTP software are going to respond differently, which makes this sort of detection difficult.

7.3. Signature Generation

Based on interaction with the honeypot servers, the Nemean architecture automatically creates signatures. These signatures are created using behavioral analysis of the interaction and some interesting mathematics. The signatures can be very simple, based on a few steps, while others can be extremely complicated. The signatures are state machines with one start state and one end state, but a web of intermediary steps that describe different levels and states of interaction that would occur with the server being monitored or initiated from the client being monitored.

7.3.1. Protocol Specific

The signatures are generated per protocol, based off of a single client and server combination and session. The session behavior and inputs from the client provide the transitions between the states in the state machine like signatures. This continues until that specific session is closed by the client. The more common protocols are currently emulated, and can provide reliable signatures with interaction. This is possible because of the sufficient amount of detail that was worked into the system for the emulation of these protocols.



21, 25, 80, 81, 135, 139, 445, 587, 1433, 8000, 8001, 8008, 8080, 8081

Figure 11 - Responsive TCP ports

7.3.2. Behavior Based

Because these signatures are not configured manually, the way they get around the usual requirement of specific packet characteristics, or characteristics of a series of packets, is by being generated on session behavior. The behavior more important to the system is that of the client that initiates the connection.

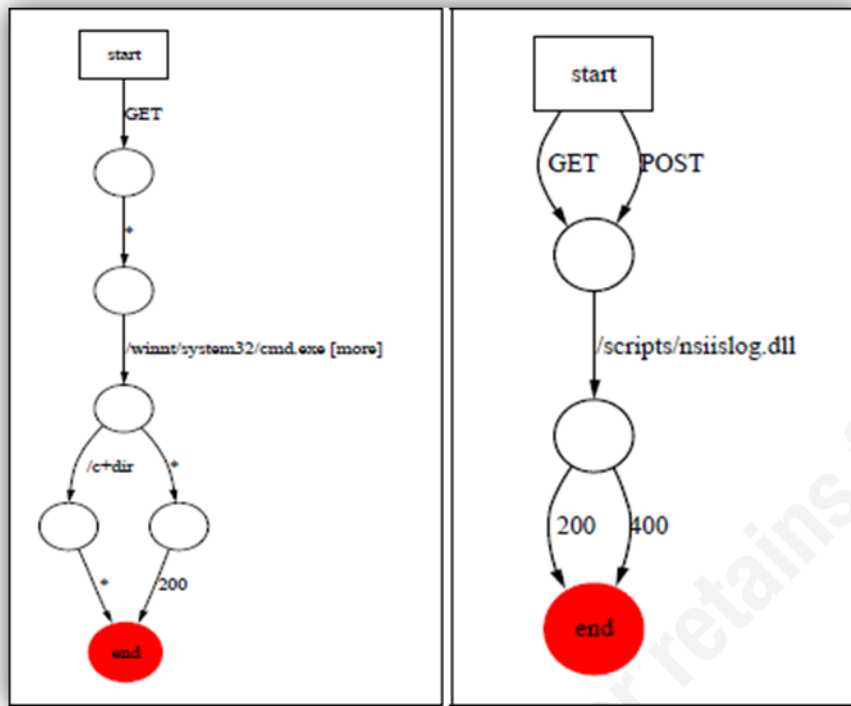


Figure 12 - From the paper "An Architecture for Generating Semantics-Aware Signatures", Nimda and Windows Media Player exploits through connection level signatures. (V. Yegneswaran, J. Giffin, P. Barford and S. Jha., 2005)

Figure 13 shows two examples of specific signatures that can be generated through the use of this tool. These examples provide some insight into how the signatures are created and what an analyst can see when looking at one of these signatures.

7.3.3. Benefits and Drawbacks

The Nemean architecture has an immense benefit of being able to monitor for unexpected activity because of these developed responsive protocols. Because some automated code can start probing for available hosts using what can be considered legitimate queries, this system can monitor for that initial conversation that would not necessarily be noticed by other tools. This in turn can start establishing the nature of the malicious code by understanding that activity from a specific host may not be kosher. Further attempts at sessions from that host may be tagged and the behavioral anomalies involved would provide the architecture the ability to generate these signatures.

Unfortunately, the architecture is unable to have signatures created for it through manual input, and thus specific exploits would first need to be attempted at potential vulnerabilities in order for signatures to be generated. Also, without the ability to have signatures based on specific packet characteristics, it may be difficult for certain types of attempted exploits detected. Analysts may have a difficult time trying to use the honeypot and honeynet in order to understand at a deeper level what is going on, independent of payloads.

7.4. IDS Portion

These signatures are applied to the IDS portion of the architecture (discussed below). The IDS portion is specific to behavioral signatures only, unless a packet is exactly matched in the middle of a session what was already seen, the IDS won't see an attempted breach. On the other hand, signature transfer has become something that is possible, but not readily supported by Nemean. The signatures are *XML* based and provide a significant amount of information for the input from the client, as well as the information sent back to the client from the server.

7.4.1. Where to Deploy

Like all IDS deployments, one of the most important things to identify is where to place the devices, in other words, what traffic is best to monitor. This depends on the importance and value of the systems that one would want to monitor. Depending on the nature of the organization, the more important servers can be some that contain financial information, or one with proprietary designs or information. These 'Golden Jewels' are the things that need the most protecting and monitoring. Below is a sample deployment, monitoring for traffic coming in from potential malicious clients or automated malicious clients.

Here is a sample network diagram that illustrates possibilities for deploying a Nemean system. In this example, the system sits off of a switch which it is using in three ways. First, it's using it as its way to communicate with the network for management purposes and for the user interface. Second, the switch is being used to respond to activity between systems and the honeynet. Finally, it's being used to listen off of a span port on the switch for traffic matching the behavior of the traffic that is presumed to be malicious or questionable.

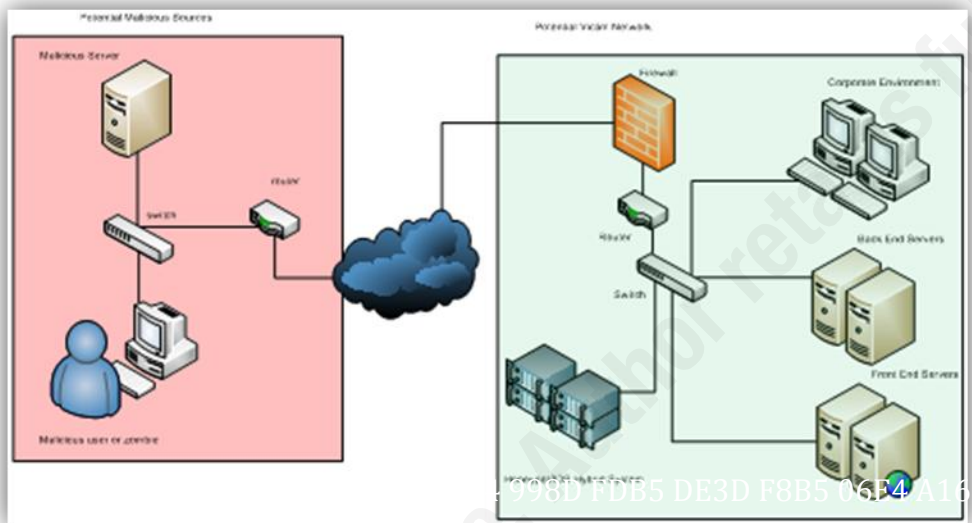


Figure 13 - Sample network diagram for deployment.

7.4.2. Catching Alerts

This architecture does a good job of catching alerts. Whenever traffic is seen that matches the behavioral nature of one of the signatures that had been generated. These alerts are protocol specific alerts and the relative interactions are included.

7.4.3. Benefits and Drawbacks

This architecture has a number of benefits, including the fact that it doesn't need to be manually configured to know on what it needs to alert. This provides a huge advantage

when monitoring for automated malware that may have found its way onto an end user's system.

These automated pieces of malicious code often take advantage of the nature of specific protocols and the fact that they are considered essential so as to not pro-actively block their use, but instead actively monitor with the intent of capturing malicious traffic as quickly as possible, in hopes of preventing or mitigating a potential breach.

7.4.3.1. Packet Captures

As far as IDS devices are concerned, one of the benefits of their nature is the ability to capture the packets involved in the alerts that may have been triggered. These are often used to validate or nullify the risk involved with those packets on a given system. This is true, in particular, for potential exploits to vulnerabilities having to do with the way operating systems reassemble fragmentation. Most operating systems handle fragmentation in their own way and not necessarily per RFC, because of this, they will see things differently. This can usually be seen by looking at the packets themselves and knowing what host they were attempting to exploit. One of the drawbacks of the current Nemean architecture is its inability to perform these captures and present them to the analyst. Instead, it provides only the payloads used in the triggered incident that prove to be the steps taken to attempt a compromise. The payloads may contain information necessary for the understanding of what the code was trying to accomplish, but without the captures, it may be difficult to know whether or not the victim system was exploitable, not exploitable or if it was successfully exploited by the malicious code without further investigation on the system, which may require some high-level scans, some deep file scans or a full on forensic investigation.

7.4.3.2. Snort

Paul Berford mentions the ability to interpret and export to Snort formatted signatures as a highly important feature in one of his research precursors to the current architecture. Unfortunately, this is still not possible, mainly because of the mathematical functions

Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

used and the mathematical difficulty to convert them properly. Instead, their signatures are generated and used only in the proprietary format.

7.4.3.3. Catching Automated Attacks

A huge benefit that this system has is the ability to detect interaction that was not expected. Because of this ability, malware that begins scanning for potentially vulnerable systems and comes across the Nemean honeynet, and because of the potential for compromise, it may attempt to spread itself to the fake servers and thus, generating a signature. This, in turn, can allow for the monitored systems to be properly watched for the potential infection. But without the ability to provide a deeper look into the session beyond payloads, analysts may find themselves with a lot of work chasing systems that may not have become true victims.

7.4.3.4. 'Drive-by' Drops

Unfortunately, based on the nature of this system, it may not be ready to take on the challenge of drive-by attacks or malicious code that is dropped on the system through means other than network-based automatic propagation. This has been a common way for code to be distributed for a long time, and though most of the viruses, worms and Trojans can be caught by using anti-virus software on systems, the newer the virus, the less chance that it has had a signature written for it, and therefore a chance that it may infect systems. From that point on, it may distribute itself through different methods such as email propagation or by setting up an HTTP server on the victim system and waiting for users to get it directly.

8. Conclusion

The threat landscape has evolved over the past few decades from targeting government and educational organizations to widespread use among businesses and homes. The evolution of threats has gone from master hackers who understood protocols and

Cristian Ruvalcaba, cristian_ruvalcaba@intuit.com

operating systems and using that understanding in hopes of gaining the respect and accolades of the hacker community. Eventually, some of these hackers went to the dark side and evolved into their new cracker persona.

These crackers, often fluent in a number of different operating systems and protocols, in the hopes of improving their effectiveness began writing automated scripts that they would use for their specific targets to be able to get in as fast as possible. The scripts spread and they fell into the hands of less skilled individuals who used them, these individuals are sometimes referred to as script kiddies.

Progress diverged into different areas: worms, Trojans, viruses; all of these dropped onto systems in from different locations. Infected web servers, infected files transferred through different methods such as file sharing, email attachments, direct link downloads from emails or websites, or other methods.

Our attack mitigation methodologies have remained rather static in nature. Usually a good understanding of the attacks is used to derive detection methods for them. With the advent of IPS, intrusion prevention systems, creation of the signatures to know how to identify and prevent the infection is still a very manual process, often times requiring packet captures of the infection in progress.

There seems to have been a stalemate in terms of progress in what can be considered passive methods of network anomaly detection. We've been writing signatures for a decade, and though progress has been made in terms of what signatures are written and how, we are still stuck doing our best to write signatures based on what we can derive from specific packets and sometimes a series of packets. As long as there are people involved, mistakes can be made. Some call these mistakes the Layer 8 issue, referring to the OSI model. The next evolutionary step would seem to be an automated method for understanding threats and generating signatures in order to detect them. Eventually we may get to the point where anomalous traffic can be detected and signatures be generated for that traffic for IPSs to protect against. An extremely high confidence in the nature of this methodology would be required and years of testing. Once the false positive rate and false negative rates can be reduced to acceptable levels, or at the very least, have automated testing methodologies that minimize at the very least the false positive rate to

prevent the blocking of legitimate traffic that could cause outages and negative business impact.

© 2010 SANS Institute, Author retains full rights.

9. References

- Cheswick, Bill. (1990) An Evening with Berferd In Which a Cracker is Lured, Endured, and Studied. [Online] Available: <http://www.tracking-hackers.com/papers/berferd.pdf> (accessed 12/16/09)
- J. Kline, S. Nam, P. Barford, D. Plonka and A. Ron. (2008) Traffic Anomaly Detection at Fine Time Scales with Bayes Nets. [Online] Available: http://pages.cs.wisc.edu/~pb/icimp08_final.pdf (accessed 12/16/09)
- Keys, Bill (2007 based on comments on article) Creating Snort Rules with EnGarde [Online] Available: <http://www.linuxsecurity.com/content/view/132365/2/> (accessed 12/16/09)
- Liston, Tom. (2001) LaBrea: “Sticky” Honeypot & IDS. [Online] <http://labrea.sourceforge.net/labrea-info.html> (accessed 12/16/09)
- Matulis, Peter (2002) Understanding Snort (Online) <http://tutorials.papamike.ca/pub/snort.html> (accessed 12/16/09)
- Nemean Networks. (2009) The Next Generation of Situational Awareness Solutions. [Online] Available: <http://www.nemeannetworks.com/> (accessed 12/16/09)
- PC Mag (1996 – 2009) IDS Definition [Online] Available: http://www.pcmag.com/encyclopedia_term/0,2542,t=HIDS&i=44731,00.asp (accessed 12/16/09)
- Provos, Niels (2008) Developments of the Honeyd Virtual Honeypot. [Online] Available: <http://www.honeyd.org/> (accessed 12/16/09)
- Snort (2009) Home Page.[Online] Available: <http://www.snort.org> (accessed 12/16/09)
- Spitzner, Lance (2002) Honeypots – Definitions and Values of Honeypots [Online] Available: http://www.windowsecurity.com/whitepapers/Honeypots_Definitions_and_Value_of_Honeypots.html
- (2003) Honeypots: Tracking Hackers. New York: Addison Wesley.
- (1999-2009) The HoneyNet Project [Online] Available: <http://project.honeynet.org/>
- Stoll, Cliff. (1988) Stalking the Wily Hacker, Association for Computing Machinery, Inc. (1989, 1990) The Cuckoo’s Egg. New York: Pocket.
- Untraceable. Dir. Gregory Hoblit. Perf. Diane Lane, Billy Burke, Colin Hanks.

Cohen/Pearl Productions.. 2008. DVD. Sony Pictures, 2008

- V. Yegneswaran, J. Giffin, P. Barford and S. Jha. (2005) An Architecture for Generating Semantics-aware Signatures [Online] Available:
http://pages.cs.wisc.edu/~pb/usenix05_final.pdf
- V. Yegneswaran, P. Barford, and V. Paxson. (2005) Using Honeynets for Internet Situational Awareness. [Online] Available:
http://pages.cs.wisc.edu/~pb/hotnets05_final.pdf
- V. Yegneswaran, P. Barford, D. Plonka.(2004) On the Design and Use of Internet Sinks for Network Abuse Monitoring. [Online] Available:
http://pages.cs.wisc.edu/~pb/isink_final.pdf