Global Information Assurance Certification Paper

Implementing Active Defense Systems on Private Networks

GIAC (GCIA) Gold Certification

Author: Josh Johnson, jcjohnson34@gmail.com

Advisor: Stephen Northcutt

Accepted: TBD

Abstract

Adversaries are using client-side attacks and malware to bypass traditional perimeter defenses and establish footholds on internal networks. Preventive controls are failing to keep attackers out of private networks, and media outlets publicize another breach at a major organization on an almost regular basis. More needs to be done in order to identify and slow down attackers who have established pivot points into private networks before data exfiltration occurs. Active defense systems implementing active deception techniques can provide a mechanism to hinder an attacker's ability to quickly and accurately identify where sensitive data resides while alerting incident response teams of malicious activity. Furthermore, these systems can be used to augment the effectiveness of IDS and SIEM strategies, strengthening the overall security posture of organizations deploying them. Active defenses offer another effective layer of security that can be used to protect an organization's crown jewels.

1. Introduction

As attacks become increasingly complex due to the sophistication, organization and motivation of adversaries, defensive strategies must mature in order to remain effective. Looking at the anatomy of an intrusion using a kill chain approach allows defenders to assess their ability to properly thwart an attack before the attacker reaches their end goal, which is typically data exfiltration. (Hutchins, Cloppert & Amin) The term "kill chain" has been adapted from a military context and can be used describe the progression and phases of an intrusion. The associated approach involves an assessment of security posture as well as intelligence gathering abilities at each individual phase of an intrusion. Various defensive techniques can be implemented at each point in the kill chain to disrupt an attack, and as long as the successful completion of the final phase – actions on the attacker's objectives – is prevented, the overall defense can be considered successful. Effective implementations of this model result in a more robust security program that is constantly improving due to the intelligence gathering processes at each phase of an attack.

With traditional antivirus solutions failing to keep pace and endpoint security being a low-hanging fruit, attackers are afforded the ability to use malware and client-side attacks to bypass some of the more established perimeter defenses, making endpoints the new perimeter (Kadrich, 2007). Social engineering and phishing attempts are able to fool even technology-savvy users into divulging sensitive information or browsing to malicious sites. For example, social engineering played a significant role after the HB Gary attack and in the related rootkit.com compromise in early 2011. (Bright, 2011) In this attack, aggressors were able to use an already compromised email account to convince a skilled security administrator to provide both a username and password to the targeted web server. If professionals working within the field of information security can be fooled in this manner, it is almost unreasonable to expect less technical end users to be able to identify and react appropriately to all social engineering attacks. Furthermore, preventive controls cannot be relied upon to stop all malicious software from making it onto endpoints. Antivirus and other signature-based defenses are routinely bypassed using well-known obfuscation techniques. Laptops and mobile devices are able to leave secure networks, connect to potentially less protected networks and then return, provided the same trust and access as before they left. Because of the breadth of the attack surface as well as the current

deficiencies in preventive controls, successful attacks against endpoints allow adversaries to accomplish several early phases of an intrusion while avoiding detection.

Mandiant's "Anatomy of an Attack" identifies the steps in a Spear Phishing attack and shows that, after a client-side attack is successful, much of the aggressor's work occurs within the victim's private network. Once the initial attack vector allows a Remote Access Trojan or backdoor to be installed on a victim's endpoint, an attacker has a pivot point into the internal network with several possibilities. (Skoudis & Liston, 2006) At this point, all reconnaissance occurs behind the perimeter firewalls; command and control traffic can be encrypted and tunneled within other legitimate protocols to bypass egress inspection and filtering systems. SIEM and properly configured IDS devices are essential to detecting these types of attacks and post-exploitation activities. However, even with these systems in place, attackers are still successfully evading detection while locating and exfiltrating the data they desire. With the implementation of active defense systems on non-Internet facing, private networks, defenders can slow down and contain attackers who have already breached perimeter defenses. Furthermore, these systems can be used to augment the effectiveness of internal IDS/IPS and SIEM systems.

Active defense systems can be defined as "any measures originated by the defender against the attacker" and broken into categories of "counterattack, preemptive attack, and active deception." (Holdaway, 2001) Counterattack techniques and Pre-emptive attacks are outside the scope of this paper due to the legal liabilities associated with these actions. However, these types of defenses are emerging areas of research and could play an important role in future security strategies as the legislation around cyber security changes and matures.

The active deception category of active defense systems can provide significant value within most organizations. The basic idea behind these systems is to increase the "cost" required for an attacker to successfully exfiltrate sensitive data. If defenders can cause their adversaries to spend more time and resources to accomplish their goals, additional intelligence can be gathered on the attackers' capabilities at each observed phase of the intrusion. With this intelligence, preventive and detective controls can be added or improved upon to better respond to future incidents from these adversaries or others using similar techniques. Another benefit of several active defense tools is the ability to "trap" or quarantine an attacker once malicious intent

is confirmed, affording defenders more time to eradicate the attacker before further damage can occur. With the implementation of these tools on private networks as opposed to Internet-facing systems, defenders are able to identify and as a result, isolate the compromised systems being used as a pivoting point by the attacker.

1.1. Legal Considerations

A wide range of active defense utilities have become popular topics of discussion over the past few years. The types of active defense utilities in public existence today range from traditional honeypots which simply monitor attacker activity, to weaponized documents that compromise attackers' systems when opened. On the more aggressive end of this spectrum, several legal issues arise when defenders consider using tools related to "hacking back" against attackers. Instead, this paper focuses on the implementation of active deception based systems on private networks in order to further reduce legal risks. In any case, it is important for network security administrators to seek qualified legal counsel before considering the implementation of any active defense systems on their networks.

1.2. Placement and Integration

The active defense tools discussed in this paper expand on traditional honeypot functionality to actively entice and trap attackers. Through listening on well-known ports and presenting tempting responses to attackers' probes, these systems presume that any address that is actually connecting to their services is suspicious. In every example presented in this paper, the implementation of active defense tools occurs on private, non-Internet addressable networks. With this configuration, all actions such as blocking, tar pitting, sink holing, etc. should be taken against RFC1918 IP address space and machines already on a private network, limiting legal liabilities for the defender. In this configuration, the sole purpose of these defenses is to identify attackers who have already infiltrated past perimeter defenses and are attempting to perform further reconnaissance and eventually data exfiltration.

DNS is an essential service frequently used within both public and private networks. Legitimate use of this service is an easy way for attackers to conceal internal reconnaissance activities through slow probing of private DNS servers. When implementing active defense systems and honeypots on private networks, administrators can assign tempting DNS names to

these systems. Since no legitimate documentation, links, etc. should contain these names, it can be assumed that anyone requesting addresses for them is malicious. Whenever a request for one of these names occurs, the DNS servers will respond with an address of an active defense system. As a result, attackers probing DNS servers for those names and then communicating with the supplied IP addresses may trigger alerts of their presence. These techniques, as well as some of the techniques implemented by several modern active defense systems serving the purpose of active deception, are not necessarily new ideas in the field of information security. For example, Fred Cohen's Deception Toolkit performed many of the same functions in 1998 as tools that are currently being developed. However, these systems and practices are still not ubiquitous in environments even though they can be deployed at low costs and offer another effective layer of detective controls.

While deception-based active defense systems are well-suited for identifying malicious activity, they can provide even more value when integrated into a defender's SIEM and IDS strategies. IDS, although not a completely foolproof technology, is an important defensive layer helping to reduce risk. (Northcutt & Novak, 2002) Since legitimate users will never connect to active defense systems, IDS rules can be created to alert on any connections to their IP addresses or specific ports. Furthermore, logging and forwarding these events to SIEM devices can allow defenders to correlate these events with logs from other systems on the network. Analysts can use a SIEM as a centralized location to review these logs, and additional IDS and active defense data provides valuable and actionable information when investigating potential security incidents. Because of these integration opportunities, active defense, IDS and SIEM systems provide significantly more value together than when operating independently. When applied to a kill chain model, each system has visibility into various phases of an intrusion, and, operating cohesively, the intelligence provided by these systems can be the deciding factor between a breach and a successfully mitigated attack.

2. Active Defense Utilities

Several existing active defense systems falling into the active deception category are available publicly. In fact, a Linux distribution called the Active Defense Harbinger Distribution (ADHD) is available containing multiple installed and preconfigured active defense systems and corresponding tutorials. (Robish, Johnson & Strand, 2013) A utility called Artillery, also

preinstalled on ADHD, is one example of an active defense system that is useful for active deception.

2.1. Artillery

Artillery is an open-source Python application created by David Kennedy from TrustedSec, also the developer of the popular Social Engineer Toolkit (SET). Artillery provides defenders the ability to install this active defense utility directly on a system that needs to be protected, and an important benefit of Artillery is the ability to install this utility on existing servers without affecting their functionality on the network. The Python-based application runs on Linux, Windows, and Mac OS X; however, the Linux version is the most full-featured. The Linux version of Artillery provides several features, including honeypot functionality, file system monitoring, brute-force and DoS protections, and threat intelligence feeds.

With all options enabled, Artillery provides a wealth of security information to system administrators. Upon startup, SSH configuration options along with web directory permissions are examined and any insecure settings are alerted to the administrator. File integrity monitoring functionality can be implemented and user-defined directories can be monitored for changes to any files within those directories. SSH login attempts to the server running Artillery are monitored, and if a user-defined threshold for failed logins is met from a single IP address, the offending address is blocked from any future communications. Connections to user-defined ports can also be limited to a specific threshold in order to prevent denial of service attacks. A threat intelligence feed is available and can be downloaded from TrustedSec, based on malicious data gathered from several of their Artillery sensors. In public-facing Artillery instances, this feature allows packet filtering rules to be created for known-malicious public IP addresses.

2.1.1. Artillery Configuration

After downloading and installing via the setup.py script, Artillery must be configured for the environment in which it is running. The file named "config" within the installation directory contains all configuration options for Artillery. For the most basic active deception setup, only the "Honeypot" section needs to be configured and all other features may be disabled. When both are set to "YES," the HONEYPOT/HONEYPOT_BAN options direct Artillery to listen on the ports identified in the PORTS configuration option. Whenever one of the defined honeypot

ports receives a connection, if the HONEYPOT_BAN option is enabled, Artillery will utilize a feature within the operating system to disallow all future communications from the connecting IP address. For example, if running on a Linux system, Artillery will create an iptables rule to block any IP address connecting to these ports. The WHITELIST_IP option gives the defender the ability to allow connections from certain addresses to ports on which Artillery is listening. This option allows for network mapping, vulnerability scanners and other systems to connect to the server running Artillery without triggering the utility to ban the whitelisted hosts. An excerpt from sample config file is shown below, defining Artillery's active defense behavior:

```
HONEYPOT=YES
HONEYPOT_BAN=YES
WHITELIST_IP=127.0.0.1,localhost
PORTS="1433,8080,21,5900,25,53,110,1723,1337,10000,5800,80,135,139,445"
```

Artillery requires no command line arguments and can be run by typing:

```
python artillery.py
```

When a Linux machine running Artillery takes action against an attacking host, it runs the following iptables command, which results in all packets sourcing from the attacker's current IP address being dropped:

```
iptables -I ARTILLERY 1 -s <ATTACKER IP ADDRESS> -j DROP
```

Once Artillery bans an IP address, the local iptables firewall will drop any connection attempts from the offending host and, as a result, all TCP connections will time out from the attacker's perspective. If the system running Artillery is being targeted, the attacker will need to utilize a new IP address or compromise another host on the network in order to communicate further with the target. Both options slow down the attacker and cause more "noise" that should be identified by IDS and SIEM systems.

2.1.2. Artillery, IDS and SIEM

Another way defenders can augment their defenses using Artillery is to create specific IDS rules to trigger whenever the Artillery ports receive a connection. Since there are no legitimate services listening on the Artillery ports, all connections to these ports can be assumed

to be malicious. Of course, IP addresses that are in the Whitelist section of the Artillery config file should be excluded from IDS rules so that Artillery and the IDS are consistently alerting on only important events. If network devices are not already performing full packet captures, IDS systems can provide added value if used to capture packets sourcing from suspicious hosts. When serving this purpose, defenders are able to monitor suspicious hosts' activities on the network even if these activities don't match a specific signature.

Snort, the widely used open source IDS, can perform this functionality when a rule is created using the post-detection "tag" option. When this option is specified in a triggered rule, Snort captures subsequent packets based on directives within the rule options. When creating the rule, a user can specify that the Snort sensor should capture traffic either within the session in which the rule was triggered or based on the source or destination host from the packet triggering the rule.

If a Snort sensor is monitoring traffic to and from a network on which Artillery is running, a rule could be created to identify a connection to an Artillery port and then capture subsequent activity from the offending source address. This provides tremendous value because Snort will log all visible traffic even after Artillery blocks any further communications to the target host. If the sensor is inspecting and logging traffic between the offending IP address and other hosts or networks in this manner, incident response teams may be able to better follow and understand the attacker's activities after being blocked by the system running Artillery. The definition of the IP and port variables preceding the rule and allowing for easier administration, and located within the Snort configuration file, is shown below:

```
ipvar ARTILLERY_INSTANCES [192.168.78.131]
portvar ARTILLERY_PORTS [1433,8080,21,5900,25,53,110,1723,1337,10000,58000,80,135,139,445]
```

Within the rules file defined in the Snort configuration file, the following rule uses these variables:

```
alert tcp any any -> $ARTILLERY_INSTANCES $ARTILLERY_PORTS (msg: "Internal IP connecting to
Artillery Port"; priority: 2;tag: host, 300, seconds, src; sid: 1000001;)
```

With this rule, Snort will alert whenever any host connects to one of the Artillery instances over one of the TCP ports on which Artillery is listening. Furthermore, for 5 minutes

after the initial rule is triggered, this rule specifies that any other packets to or from the source address that triggered the alert should be logged as well. For incident response teams, this additional information can be crucial to successfully containing and eradicating an attacker who has already breached perimeter defenses.

As another alerting opportunity, Artillery can also be configured to send emails to administrators whenever a violation occurs. However, since its logs can be forwarded via syslog, all alerts can be centrally managed by a SIEM with alerting capabilities. The configuration file for Artillery contains an option, SYSLOG_TYPE, to log to a remote syslog server. Setting that option to "REMOTE" and defining the SYSLOG_REMOTE_HOST option with an IP address of the log aggregator will allow for the following event to be analyzed by a SIEM whenever one of the honeypot ports receives a connection:

2013-04-24 21:23:50.992204 [!] Artillery has blocked (and blacklisted the IP Address: <OFFENDING IP ADDRESS> for connecting to a honeypot restricted port: 10000\x0a

A SIEM could identify this event and correlate it with any other anomalous activity to provide administrators with visibility that may otherwise remain unseen. For example, a rule could exist to look for connections from an individual IP address to Artillery ports as well as anomalous outbound connections from the same address, either by volume of data or by number of connections, to provide automated alerting and escalation capabilities. However, depending on the sophistication and methodologies used in the attack, a skilled analyst reviewing all available events and flows related to the source IP address blocked by Artillery may be necessary to identify a potentially compromised system. In this case, the SIEM provides the convenience and efficiency of having all known information regarding the suspicious IP address in a single location and normalized format.

At the time of this writing, the current version of Artillery (0.7.1) has a bug where the source code will need to be modified slightly in order for the logging functionality to work correctly. Some portions of the script look for the HONEYPOT_BAN option to equal "YES" while others look for it to equal "ON." Before starting Artillery, this can be corrected by editing the source of the "honeypot.py" script and editing all lookups to be consistent with the setting configured in the config file.

Josh Johnson, jcjohnson34@gmail.com

Using Artillery or another active defense utility running directly on an existing system, attackers on the internal network may be more easily identified. However, the intelligence available when these systems are integrated with IDS and SIEM systems provides substantially more value. Snort's tagging and a SIEM's correlation abilities, for example, allow incident response teams to quickly and easily see relevant data surrounding the suspicious IP addresses reported by Artillery. As a result, false positives can be more easily identified while true security incidents can initiate incident response procedures more quickly and with higher confidence.

2.2. Nova – Network Obfuscation and Virtualized Anti-Reconnaissance

When running active defense systems like Artillery, additional software must be installed on the protected system, which in some cases may not be an option. For example, installing these utilities on embedded systems may not be feasible. In such cases, active defense systems can achieve similar results when installed "nearby" on internal network segments. Several instances of Artillery could be installed on separate, dedicated honeypot servers on the same network as the embedded systems. A major downside of this specific approach is that there is currently no centralized management console for deploying multiple Artillery instances. Instead, a utility called Nova provides defenders the ability to spawn several virtualized honeypots from a single management console.

Nova is an open source project developed by DataSoft, a technology company well known for innovative products in the wireless communications space. The purpose of Nova is to identify reconnaissance activity within private networks. It serves as a management console for Honeyd, a separate open source project; Honeyd was developed by Niels Provos with a purpose of creating virtual honeypots that run from a single machine. While Honeyd is dedicated to building and deploying virtual honeypots, Nova extends its functionality by providing a utility that classifies all connections to the running honeypots. The purpose of Nova is to create a "haystack" of virtual hosts on the network, taking advantage of unused address space for anti-reconnaissance purposes. Once several of these honeypots are up and running, attackers attempting to map the network must weed through the haystack nodes in order to find the "real" servers they are seeking. Since Nova is always classifying traffic, if the attacker performs any port scans or other known reconnaissance techniques, the offending source address will be flagged as suspicious and provide alerts to network administrators. Using Nova on internal

networks, defenders can identify attackers and quickly take action to quarantine compromised systems and eradicate threats.

2.2.1. Nova Installation

The installation and setup of Nova is fairly straightforward, and it can be installed on systems with only a single network interface since Honeyd virtualizes the haystack nodes. When installing Nova, it is important to refer to the readme file and make sure all dependencies are installed before starting the startup daemon. Ubuntu 12.04 (64-bit) is currently the recommended platform for installation, and the developers have created a helper script, called novaInstallHelper.sh to install the required applications before configuring Nova on this distribution. The script can be run by navigating to the same directory, ensuring that the correct permissions are set on the file, and then typing:

```
./novaInstallHelper.sh
```

Once the system is up and running, most of Nova's functionality can be configured from the web interface, although there is also a command line toolset that can also be used to quickly administer the utility. The web server, which runs on the Node JS platform, can be started with the following command:

```
quasar
```

2.2.2. Nova Configuration

If configuring Nova using the web interface, a setup wizard is available to quickly get the utility up and running. Initial options include changing default credentials, setting and configuring alert and logging thresholds, and building the haystack of virtualized honeypots. Since the Nova default credentials are well known, it is important to change the username and password for the web interface if Nova is being used outside of a lab environment. If SIEM integration is desired, the "Use RSyslog" option should be checked, and the remote log server fields must be defined. Rsyslog is an open source utility improving on the syslog standard to provide several advantages including TCP and encryption support, and this software is the default logging utility used by Nova. Logging options can also be configured at a later time from the "Basic Options" menu.

Once the initial wizard is completed, users are redirected to the status page, which will show all suspected malicious hosts identified by Nova. Initially, the Packet Classifier and the Haystack will both be disabled, but both can be started from anywhere within the web GUI. Once started, the Packet Classifier will begin sniffing the network and capturing all available traffic to determine if hosts are benign or may be malicious. Figure 1 shows the Nova web interface when the packet classifier and haystack are started but have not yet seen enough network traffic to classify hosts as malicious or benign.

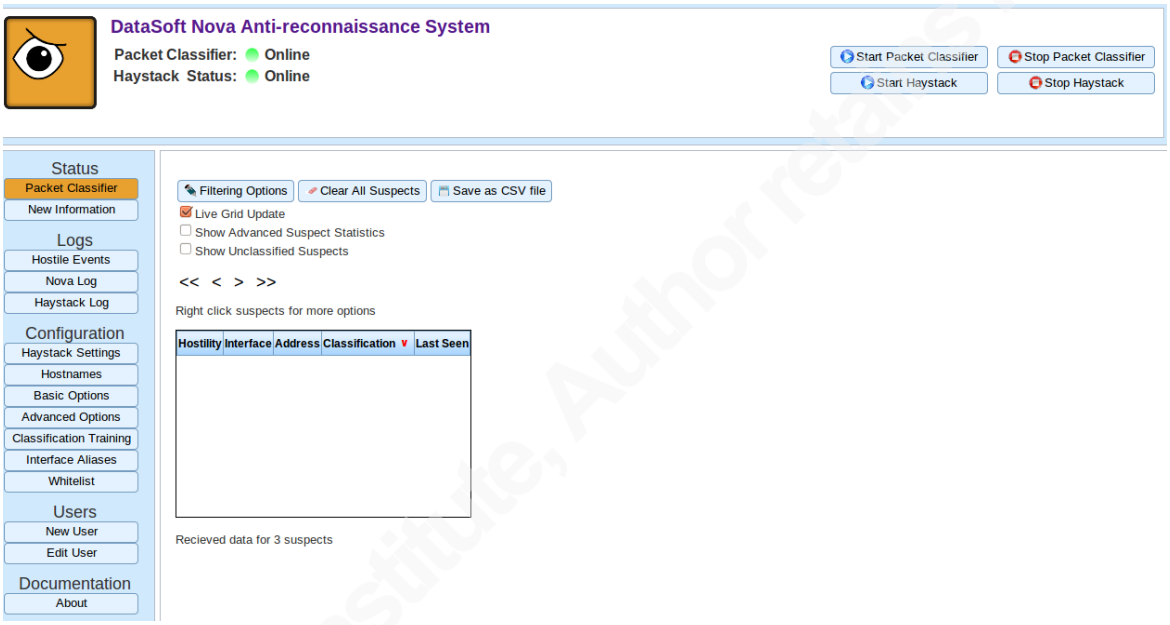


Figure 1. Nova Web Interface

2.2.3. Nova Haystacks

Nova's haystacks are comprised of modular profile components that make up the personality of the various honeypots. The profiles designate the operating system emulation, open TCP/UDP ports, and other configuration options which are assigned to the haystack nodes. Once deployed, a haystack is made up of several nodes, each with an assigned profile. As profiles are configured, Nova maintains the profile settings in the form of a configuration file passed to Honeyd when the haystack is started.

When deciding how to deploy Nova's haystacks, a few different options exist. First, a pre-built, default haystack can be used. This default option includes unique profiles assigned to 3 different nodes, each emulating a different operating system. The three profiles include a

Windows 2003, Linux and BSD Server. The emulated Windows server listens on several well-known Windows ports while the Linux and BSD servers also listen on several ports associated with common services. All three honeypots also emulate network services that are typically found running on these three operating systems. As a result, when an attacker performs active reconnaissance against the network on which Nova's haystack is deployed, these systems respond in a way which indicates to the attacker that they are live systems on the network running interesting services that may require further reconnaissance efforts.

Nova administrators can also build their own custom haystacks with honeypots that resemble real operating systems and services. Custom profiles can be created to deploy nodes which emulate the operating system of one of thousands of possible operating systems. Each profile can be configured to listen on any combination of user-defined TCP or UDP ports and even emulate services on some ports based on service scripts from the Honeyd project. These scripts, described in further detail below, imitate actual network services by presenting the same banners upon receiving connections and even handling user interactions like their real-world counterparts. Using custom haystacks, defenders can deploy a fleet of virtual honeypots exhibiting fine-tuned and specific behaviors when responding to probes from attackers. Figure 2 shows the web-based profile configuration interface available when building a custom haystack profile.

Josh Johnson, jcjohnson34@gmail.com

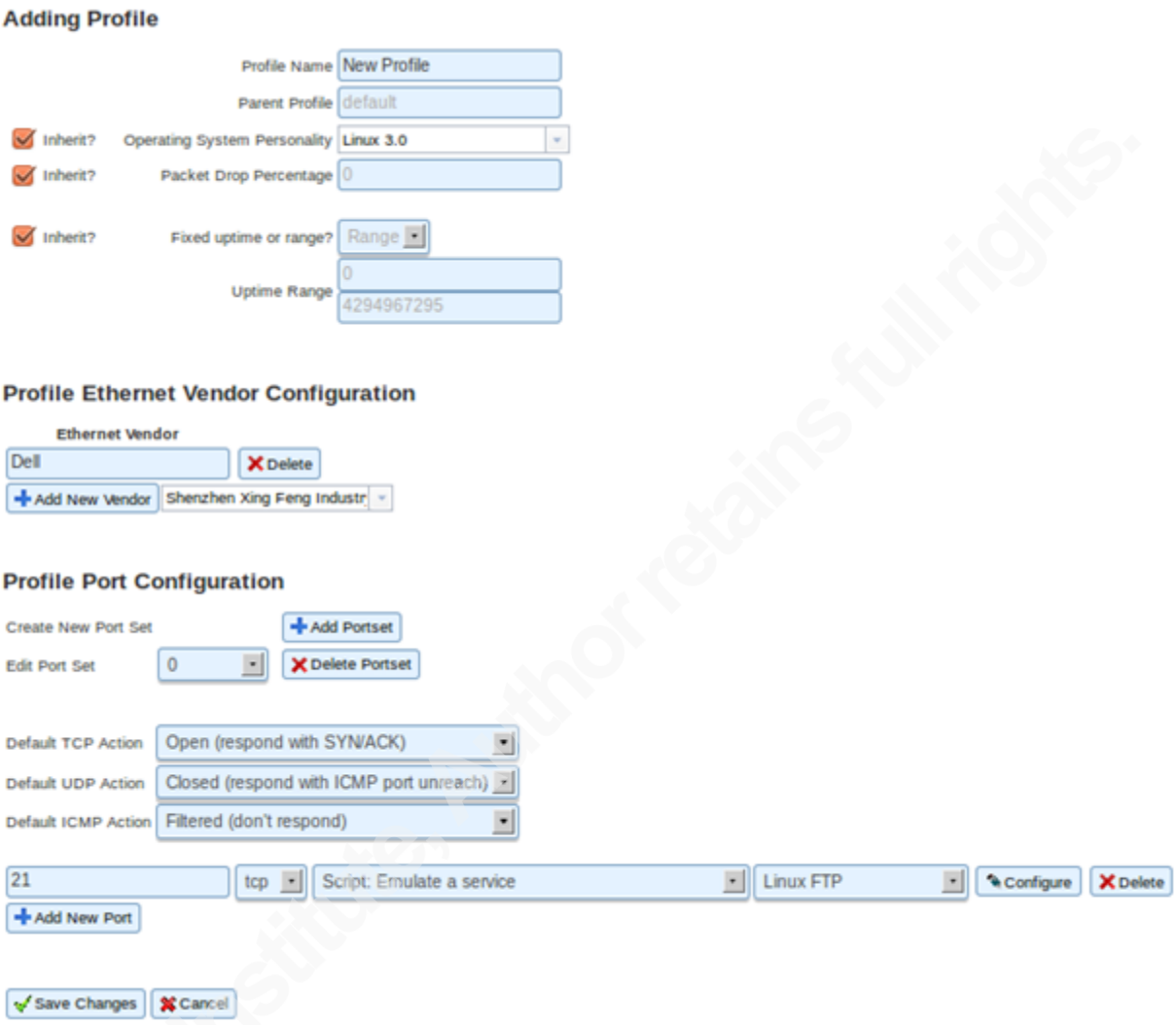


Figure 2. The available haystack profile configuration options

When building a haystack, administrators can also use the makeup of existing systems on their network to influence the composition of the haystack. Attributes of the real systems such as operating system, open ports and listening services can be imitated by honeypots. As a result, TCP/IP characteristics of the haystack nodes appear identical to the real systems from the perspective of an attacker performing network based reconnaissance. Nova's interface provides a simple mechanism to do this using Nmap, a popular and widely used port scanner and security utility. Through the web-based user interface, administrators can run Nmap and apply the results to a haystack. As a result, attackers must waste time going deeper than simple port scans and actually enumerating the services on each IP address in order to identify which systems on the network are real.

Josh Johnson, jcjohnson34@gmail.com

2.2.4. Honeyd Service Scripts

The service scripts available for use in Nova's haystacks are powerful tools to slow down and confuse attackers who are performing reconnaissance. The scripts allow for defenders to create their own network of Nova honeypots, all appearing to run specific versions of software that may be a tempting target for brute-force login attacks or other exploitation by an attacker. Furthermore, the Honeyd service scripts can cause attackers to waste time interacting with the service, thinking that it is a real server. In the meantime, network administrators are alerted to the attacker's presence on the internal network. This technique was popularized by the Deception Toolkit, whose author, Fred Cohen, insisted on the DTK's website that an active deception utility is successful when it affects how the attacker approaches reconnaissance. A full list of the possible services is shown in Figure 3. Each service script can be configured to run as specific version of that type of service. For example, a Linux FTP Service script offers the opportunity to appear as five different FTPD versions, also shown in figure 3.

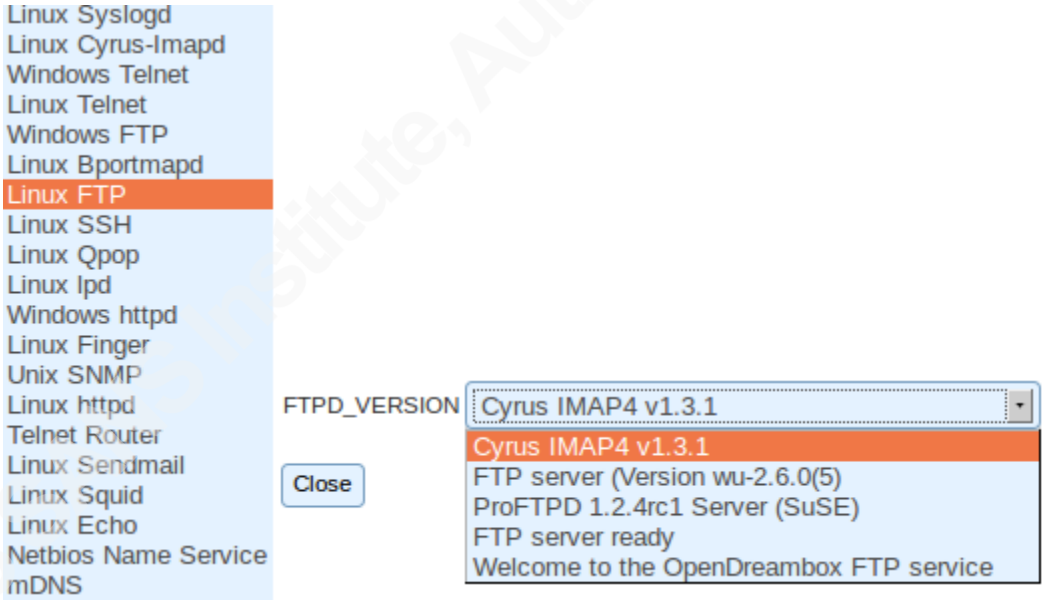


Figure 3. List of possible service scripts and the specific version options available

Nova also provides the ability to set up a "tarpit" on listening ports using Honeyd directives. Tom Liston popularized the term, tarpit, when he developed Labrea in response to the Code Red worm in 2001. His idea was that he could respond to malicious TCP SYN probes from infected hosts with TCP packets that included the SYN/ACK flags and set the TCP maximum segment size to a small number, but then never respond to further communications

from these hosts. (Haig, 2002) His technique was successful and the tarpit application caused malicious hosts to wait until the TCP session timed out before continuing their attack. As a result, the spread of the worm slowed and defenders had more time to isolate and eradicate the infections. Nova and Honeyd's implementation of tarpits serves the same purpose: slow down an attacker by delaying responses to connection attempts and data transfers. These tarpits can be applied to open ports when configuring haystack profiles and they can be utilized on both ports that simply allow TCP connections as well as ports that emulate services. Several tarpits applied to ports on a single node in a haystack can have a devastating impact on port scanners, causing the utilities to take a significant amount of time to complete. This defensive technique can provide defenders valuable time to contain and eradicate attackers from the network. Figure 4 shows the possible tarpit configuration options available for each port listening on a honeypot.

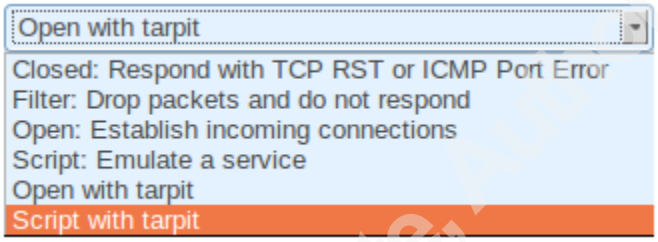


Figure 4. Tarpit configuration options

2.2.5. Nova's Classification

Since Nova's packet classifier is always sniffing traffic on available interfaces, whenever traffic is visible, the associated data is processed through a classification algorithm to determine if the connecting host is malicious. Nova uses the K-Nearest Neighbor algorithm to classify traffic based on a database of previously classified training data. The algorithm compares values such as average packet size, variation in packet sizes, distinct TCP and UDP ports contacted and several other data points to accurately identify attackers performing known methods of reconnaissance on the network. This implementation of the algorithm works by calculating several summary statistics for connections from a specific IP address, comparing the summary statistics to the training data, and using the results to identify whether the IP address is exhibiting malicious or benign behavior.

When Nova is running, every time a haystack node is contacted by a new source IP address, a SQLite database is populated with summarized data based on all connections from each unique IP address. That data is then compared to the training data using Nova's classification algorithm. User-defined thresholds, configured in the Advanced Options settings, allow users to adjust how closely traffic to the haystack must match the training data in order to be considered benign or malicious.

Although Nova is not designed to detect specific Nmap use on the network, since Nmap implements some of the most well-known, well-documented and widely used active reconnaissance techniques, Nova has great success identifying the use of this utility. For example, a TCP half-open scan, executed by using the "-sS" flag when running Nmap, is easily identified as malicious behavior due to the high percentage of packets with only the SYN flag set versus the number of packets containing the other flags (SYN+ACK, ACK, etc.) necessary for proper TCP communication to occur. Since Nova can be trained to identify reconnaissance activities based on statistical summaries of traffic known to be malicious, observing and summarizing Nmap scans provides effective training data that encompasses many port scanning methods. In fact, the default training data that comes packaged with Nova provides accurate identification of many popular Nmap scans with no further training needed.

2.2.6. Nova's Doppelganger

When used in non-technical conversation, the term doppelganger is often used to describe a body double or someone who looks very similar to another person. Nova implements the idea of a doppelganger as an active defense mechanism, allowing defenders to conceal the system running Nova from IP addresses it has deemed malicious. This feature provides administrators with confidence that once an attacker is identified on the network, the Nova interface and management console cannot be accessed by the attacker's IP address. Instead, an additional honeypot, the Doppelganger, is deployed and listens on a separate interface when the haystack is started. Once an attacker is identified and classified as malicious, all packets from the attacker's IP address to the Nova management interface IP address are forwarded to the Doppelganger. Using this feature, an attacker will not be able to directly attack the Nova console from any IP addresses that have been classified as malicious.

Josh Johnson, jcjohnson34@gmail.com

2.2.7. Nova, IDS and SIEM

Once Nova is installed, configured and running on a private network, administrators can create Snort rules to also identify when Nova's honeypots receive connections from unauthorized sources. IP ranges can be defined and classified in the Snort configuration file so that Nova-related alerts will not be triggered when connections are made to the real servers on the network. For example, if the Nova Haystack uses the 192.168.78.132-192.168.78.135 IP addresses, the following declaration could allow the haystack to be utilized in Snort rules:

```
ipvar NOVA_HAYSTACK [192.168.78.132-192.168.78.135]
```

Once the addresses of the Nova haystack are defined, a rule can be created to alert and tag sources coming from any IP address to the haystack. This provides defenders with a better picture of what an attacker is doing on the network.

```
alert tcp any any -> $NOVA_HAYSTACK any (msg: "Internal IP connecting to Nova Haystack node";
priority: 2;tag: host, 300, seconds, src; sid: 1000002;)
```

With the above declaration and associated rule, Snort can be used to tag all connections to nodes within the haystack. This results in packet captures for any subsequent network communications from the offending source address, even if the communications do not match another Snort signature.

Similar to Artillery, logs from Nova can be correlated with events from disparate systems in order to identify attackers on the network and build a better picture for defenders to analyze an attack. Nova has the ability to forward its logs to a SIEM using Rsyslog, and exactly what is forwarded can be configured in the "Basic Options" section within Nova. Logging options include 7 different classifications of events, so administrators can log events ranging from debugging notifications up to only critical incidents and alerts. Given the above Snort configuration and rules, SIEM correlation rules can be defined to alert whenever a Snort rule is matched and a corresponding Nova alert identifies the same source IP address as hostile.

Figure 5 shows a sample log event generated by Nova whenever an offending IP address is deemed as hostile based on the Nova's classification algorithm.

<81>Apr 29 15:42:09 ubuntu Nova[17268]: ALERT File ../src/Novad.cpp at line 850: Detected potentially hostile traffic from: Suspect: 192.168.78.1#012 Suspect is hostile#012 Classification: 1#012 Hostile neighbors: 2#012Classification notes: #012k=0:d=0.0756927:c=0:i=87#012:o 0.500375 0.109864 45.6108 18.095 2 1000 2 500.5 2 0.993171 0 0 0 0 #012:n 0.500375 0.109864 0.525279 0.391617 0.18718 1 0.149684 0.89996 0.159017 0.997144 0 0 0 0 #012#012k=1:d=0.0855165:c=1:i=185#012:o 0.504487 0.112332 46.0603 23.1512 2 1000 1 1000 1 0.996016 0 0.00298805 0 0 #012:n 0.504487 0.112332 0.526591 0.422808 0.18718 1 0.0944402 1 0.100329 1 0 0.00299105 0 0 #012#012k=2:d=0.0864056:c=1:i=191#012:o 0.504487 0.112332 46.1433 23.6369 2 1000 1 1000 1 0.996016 0 0.00298805 0 0 #012:n 0.504487 0.112332 0.526832 0.425451 0.18718 1 0.0944402 1 0.100329 1 0 0.00299105 0 0 #012#012#012IP Traffic Distribution: 0.516071#012Port Traffic Distribution: 0.0572391#012Packet Size Mean: 44.9643#012Packet Size Deviation: 5.89038#012Protected IPs Contacted: 2#012Distinct TCP Ports Contacted: 560#012Distinct UDP Ports Contacted: 1#012Average TCP Ports Per Host: 560#012Average UDP Ports Per Host: 1#012TCP Percent SYN: 0.979021#012TCP Percent FIN: 0#012TCP Percent RST: 0.0192308#012TCP Percent SYN ACK: 0#012Haystack Percent Contacted: 0.2#012TCP RST Packets: 11#012TCP ACK Packets: 0#012TCP SYN Packets: 560#012TCP FIN Packets: 0#012TCP SYN/ACK Packets: 0#012Total TCP Packets: 571#012Total UDP Packets: 18#012Total ICMP Packets: 0#012Total other protocol packets: 0

Figure 5. Example log event generated by Nova

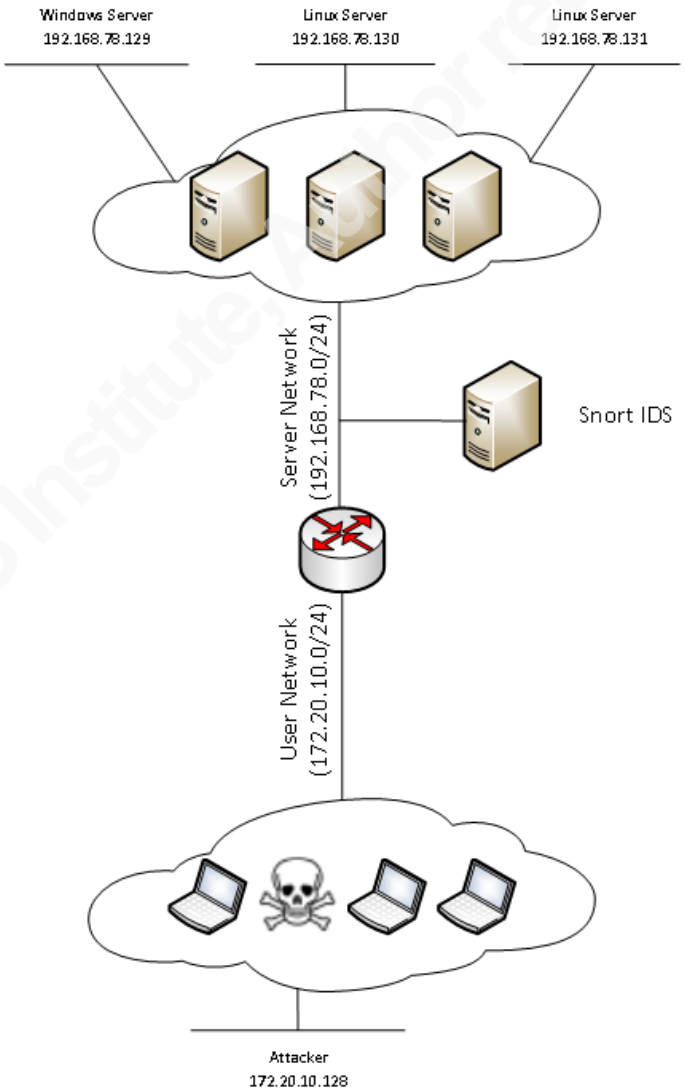
Several interesting facts about the traffic classified as malicious can be learned from this output. First, the "Suspect: 192.168.78.1" and "Suspect is hostile" sections show that Nova has indeed seen enough from this source address to apply the classification algorithm and flag this host as hostile. Also, in this case, the "Percent SYN," "Percent RST," "Percent FIN" and "Percent SYN ACK" section shows that the attacker sent mostly TCP packets with only the SYN flag set, and no packets with the SYN and ACK flags set. Based on this information, it can be assumed that the attacker performed a TCP half-open port scan against the haystack nodes since there was never a complete TCP connection built.

With this information, along with the Snort alert and additional tagged traffic, defenders may be able to better understand their adversaries' capabilities and intents. Nova, combined with strong IDS and SIEM implementations, can be a powerful tool to detect attackers performing reconnaissance on private networks.

Josh Johnson, jcjohnson34@gmail.com

3. Putting it all Together

The tools demonstrated below are configured based on the following network design and scenario. Consider the network layout where an attacker has used malware or a client-side attack to compromise a PC on a company's internal network. The internal PC user network (172.20.10.0/24) is not segmented from an internal server network (192.168.78.0/24) with a firewall, but a Snort IDS sensor is inspecting all connections between the two networks. The server network consists of several Windows and Linux-based systems. The attacker on the network has installed a Remote Access Trojan on the PC with the address 172.20.10.128 as shown in Figure 6.



Josh Johnson, jcjohnson34@gmail.com

Figure 6. Example network layout

In this scenario, network administrators have implemented Artillery on one of the Linux servers ending in .131 as well as three Nova honeypots using 192.168.78.132-192.168.78.134. The entire range of active IP addresses in this scenario is 192.168.78.129-192.168.78.134. IDS rules using Snort's tag option have been implemented to detect and log any connections to the Artillery ports or Haystack IP addresses. As such, once the attacker triggers a Snort signature, all future traffic from the offending IP address will be logged for the next 300 seconds. If an attacker runs a simple Nmap SYN scan against this range, the results will appear as the following:

```
Nmap scan report for 192.168.78.129
Host is up (0.023s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown

Nmap scan report for 192.168.78.130
Host is up (0.0018s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
3306/tcp open  mysql

Nmap scan report for 192.168.78.131
Host is up (0.0026s latency).
Not shown: 983 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
1433/tcp  open  ms-sql-s
1723/tcp  open  pptp
3389/tcp  open  ms-wbt-server
5800/tcp  open  vnc-http
5900/tcp  open  vnc
8080/tcp  open  http-proxy
10000/tcp open  snet-sensor-mgmt
44443/tcp open  coldfusion-auth

Nmap scan report for 192.168.78.132
Host is up (0.0049s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap scan report for 192.168.78.133
Host is up (0.0048s latency).
Not shown: 995 closed ports
```

```
PORT    STATE SERVICE
20/tcp  open  ftp-data
21/tcp  open  ftp
23/tcp  open  telnet
80/tcp  open  http
135/tcp open  msrpc
```

```
Nmap scan report for 192.168.78.134
Host is up (0.0015s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE
22/tcp open  ssh
23/tcp open  telnet
```

The Nova honeypots are implementing tarpit functionality on open ports, so the port scanning process was extremely slow from the attacker's perspective. Since the attacker has scanned IP addresses within the Nova Haystack, an IDS alert has been triggered. Furthermore, since the Snort tag option was used in this rule, all traffic from the attacker's source address traversing the IDS sensor will be logged for the next five minutes. Throughout the course of the port scan, Nova has examined enough traffic from this address to classify the source IP address as hostile as shown in Figure 7.

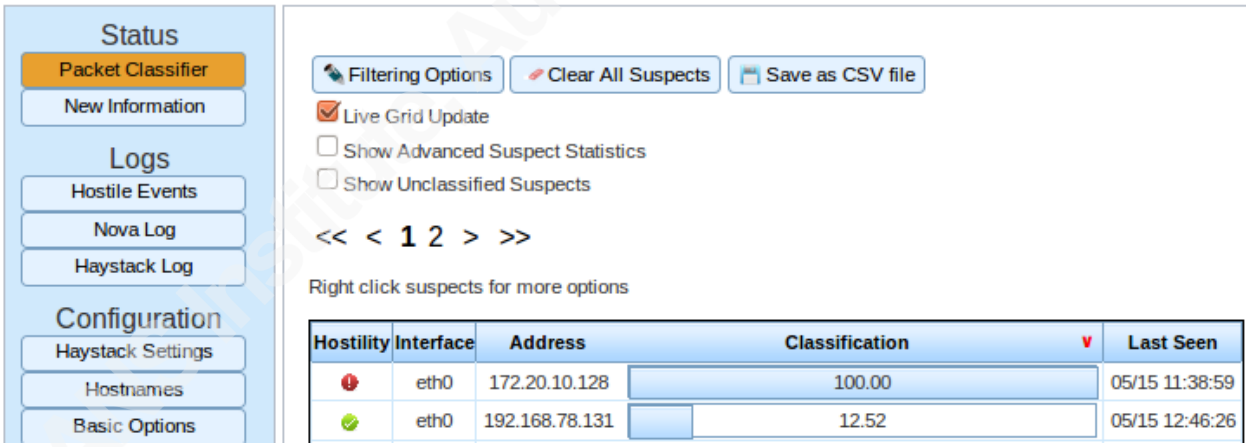


Figure 7. Nova classifying an IP address as malicious

Nova has also forwarded a log event to the log aggregator identifying this classification so the SIEM now has IDS and Nova alerts pointing to the attacker's IP address. The attacker has not yet been actively blocked for any of the reconnaissance activities, but several events of interest are available to defenders.

Now that the attacker understands which ports are open on these systems, complete TCP connections are needed in order to further identify which services are listening on these ports. Nova provides even greater value in this scenario, since the Honeyd service scripts and tarpits

can be used to slow down the attacker. For example, the attacker may decide to probe the honeypot emulating a Linux-based operating system running an FTP service on TCP/21, SSH service on TCP/22 and HTTP service on TCP/80. The service scripts respond to connections on each of these ports, emulating their respective services and allowing seemingly normal interaction, even though they're not really FTP, SSH and HTTP daemons. From an attacker's standpoint, running an Nmap service/version scan against live this live Nova host results in the following output:

```
root@kaliLinux:~# nmap -sV 192.168.78.132

Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-03 16:09 EDT
Nmap scan report for 192.168.78.132
Host is up (0.0036s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     WU-FTPD wu-2.6.0
22/tcp open  ssh     OpenSSH 2.1.1 (protocol 1.99)
80/tcp open  http    Apache httpd 2.2.20
MAC Address: A4:BA:DB:7D:27:5E (Dell)
Service Info: Host: 192.168.78.226; OS: Unix

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.58 seconds
```

Based on this output, an attacker may notice that the advertised service banners reflect versions of software which not been upgraded in some time and may be vulnerable. In fact, a Google search would yield several possible exploits to use against these versions of the services. If the attacker tries to interact with one of the services that are running a script, the Honeyd script will generate responses that appear to be consistent with the service they are probing. For example, in the case of the FTP server above, the attacker can connect and "log in" as an anonymous user to the FTP server:

```
root@kaliLinux:~# ftp 192.168.78.132
Connected to 192.168.78.132.
220 serv.local.mynet Cyrus IMAP4 v1.3.1 Fri May  3 00:42:57 EDT 2013) ready.
```

```
Name (192.168.78.132:root): anonymous
331 Guest login ok, send your complete e-mail address as a password.
Password:
230-Hello User at 172.20.10.128,
230-we have 62 users (max 100) logged in in your class at the moment.
230-Local time is: Fri May  3 00:42:57 EDT 2013
230-All transfers are logged. If you don't like this, disconnect now.
230-
230-tar-on-the-fly and gzip-on-the-fly are implemented; to get a whole
230-directory "foo", "get foo.tar" or "get foo.tar.gz" may be used.
230-Please use gzip-on-the-fly only if you need it; most files already
230-are compressed, and I will kill your processes if you waste my
230-ressources.
230-
230-The command "site exec locate pattern" will create a list of all
230-path names containing "pattern".
230-
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Also, if the web server is targeted, the attacker can interact with the "Apache" service script which returns a web page and even HTTP 404 errors if "unknown" files are requested:

```
root@kaliLinux:~# nc -v 192.168.78.132 80
Connection to 192.168.78.132 80 port [tcp/www] succeeded!
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Wed May  8 13:59:48 EDT 2013
Server: Apache/1.3.23 (Unix) (SuSE/Linux) ApacheJServ/1.2.2 mod_fastcgi/2.2.2 mod_perl/1.34 PHP/4.1.0
mod_ssl/2.8.7 OpenSSL/0.9.6c
Last-Modified: Wed, 25 Jan 2012 21:57:56 GMT
ETag: "581ea2-b1-4b7615b37f7aa"
Accept-Ranges: bytes
Content-Length: 263
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
```

```
<html><head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1"></head><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>


 root@kaliLinux:~# nc -v 192.168.78.132 80
Connection to 192.168.78.132 80 port [tcp/www] succeeded!
GET /index.php HTTP/1.0

HTTP/1.1 404 Not Found
Date: Wed May  8 14:01:14 EDT 2013
Server: Apache/1.3.23 (Unix) (SuSE/Linux) ApacheJServ/1.2.2 mod_fastcgi/2.2.2 mod_perl/1.34 PHP/4.1.0
mod_ssl/2.8.7 OpenSSL/0.9.6c
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /index.php was not found on this server.</p>
<hr>
<address>Apache/2.2.20 (Ubuntu) Server at 192.168.78.132 Port 80</address>
</body></html>
```

Further interrogation of these services will allow the attacker to realize these are not functional daemons. However, wasting time with actions such as brute-force password guessing or exploitation may trigger additional IDS alerts while allowing defenders more time to identify and eradicate the attacker.

If the attacker makes a connection to any of the honeypot ports on the Linux system running Artillery, the active defense system will permit the initial connection, send garbage data back to the attacker as shown in below, and then close the connection:

```
root@kaliLinux:~# nc -v 192.168.78.132  135□6□□□m
□□□LT/C□□@————————————————————————j□)
]~O"□□
```

Josh Johnson, jcjohnson34@gmail.com

!□u□□□9□□□Ŧ□□□I□□□Rn□+D%X□l1□□l□□□$${□P□uJ□□□□4□J□□□{9l□

Connection to 192.168.78.132 135 port [tcp/loc-srv] succeeded!

□a□i□□rLE□"□!□"&□_□□□□□□1□[I□□□□□□□□□□*□□xh□.□□□G□8□□
□E1□QP□P□□*□/□□ID□□<□□z□□□□□n□Z□yf□-
6□

u□c□#

□8T□Jb□□□□I□>□□□b}□)d□□n¯□□□}E□ iGC
%q^C□□□X□C□□□a□>□□□□y□□-□□□Sx□□□.□□q□1□0□□□9□□:SM?G□□

Artillery will also log the event and forward it to the remote log aggregator as well as create an iptables rule that drops any future connections from the attacker's IP address. Yet another IDS alert will be triggered because of this connection as well. Since Snort's tag option was used in the rule, future traffic from the attacker's address will be logged. If the attacker attempts to perform a vulnerability scan against the legitimate web server on one of the Linux servers within 5 minutes after triggering one of these alerts, defenders will be afforded the ability to see the additional malicious traffic within this window. For example, Figure 8 shows a packet capture, logged by Snort, of the attacker scanning the web application running on the Linux server.

Filter: ip.addr eq 172.20.10.128 Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.10.128	192.168.78.130	HTTP	172	GET / HTTP/1.1
2	0.091655	172.20.10.128	192.168.78.130	HTTP	178	GET /Uk4SuuSf. HTTP/1.1
3	0.094086	172.20.10.128	192.168.78.130	HTTP	180	GET /BgwxyzcW.do HTTP/1.1
4	0.100983	172.20.10.128	192.168.78.130	HTTP	181	GET /Czj9r4XA.gif HTTP/1.1
5	0.108316	172.20.10.128	192.168.78.130	HTTP	180	GET /ugzE2KBU.py HTTP/1.1
6	0.109854	172.20.10.128	192.168.78.130	HTTP	181	GET /1eepPgKI.asp HTTP/1.1
7	0.114325	172.20.10.128	192.168.78.130	HTTP	181	GET /BokOY9TH.htm HTTP/1.1
8	0.117953	172.20.10.128	192.168.78.130	HTTP	180	GET /glftT7hE.rb HTTP/1.1
9	0.124666	172.20.10.128	192.168.78.130	HTTP	181	GET /Hwc9xBUz.jsp HTTP/1.1
10	0.124736	172.20.10.128	192.168.78.130	HTTP	183	GET /ofYSYfct.htmls HTTP/1.1
11	0.149601	172.20.10.128	192.168.78.130	HTTP	181	GET /x4fVXz1A.cgi HTTP/1.1
12	0.152930	172.20.10.128	192.168.78.130	HTTP	181	GET /2Hvta880.php HTTP/1.1
13	0.155450	172.20.10.128	192.168.78.130	HTTP	183	GET /NlFBok4o.xhtml HTTP/1.1

Figure 8. Packet capture showing all traffic logged as a result of the Snort "tag" option

Since the attacker has triggered IDS alerts as well as alerts on both the Artillery and the Nova hosts, log review should allow incident responders to accurately identify the source of the

malicious traffic. The log events in Figure 9 include a sample of the events generated by Nova, Artillery and Snort during the course of the attacker's reconnaissance.

ALERT File ../src/Novad.cpp at line 850: Detected potentially hostile traffic from: Suspect: 172.20.10.128#012 Suspect is hostile#012 Classification: 1#012 Hostile neighbors: 3#012Classification notes: #012k=0:d=0.351803:c=1:i=197#012:o 0.995722 0.432286 59.881 1.53793 3 1000 0 1000 0 0.993723 0 0.00297324 0 1 #012:n 0.995722 0.432286 0.561797 0.123662 0.236194 1 0 1 0 0.997698 0 0.00297623 0 1 #012#012k=1:d=0.705239:c=1:i=198#012:o 0.888347 0.133096 59.863 1.65254 22 1000 1 949.048 1 0.993083 0 0.00377747 0 0.95 #012:n 0.888347 0.133096 0.561757 0.129527 0.53422 1 0.0944402 0.992438 0.100329 0.997055 0 0.00378126 0 0.95 #012#012k=2:d=0.779816:c=1:i=196#012:o 1 0.5025 43.9801 0.281436 1 1000 0 1000 0 0.994036 0 0.00497018 0 0.333333 #012:n 1 0.5025 0.52041 0.0329263 0.118097 1 0 1 0 0.998012 0 0.00497517 0 0.333333 #012#012#012IP Traffic Distribution: 0.830285#012Port Traffic Distribution: 0.830285#012Packet Size Mean: 43.8766#012Packet Size Deviation: 0.69156#012Protected IPs Contacted: 6#012Distinct TCP Ports Contacted: 164#012Distinct UDP Ports Contacted: 0#012Average TCP Ports Per Host: 136.167#012Average UDP Ports Per Host: 0#012TCP Percent SYN: 0.968009#012TCP Percent FIN: 0#012TCP Percent RST: 0.0308057#012TCP Percent SYN ACK: 0#012Haystack Percent Contacted: 1#012TCP RST Packets: 26#012TCP ACK Packets: 0#012TCP SYN Packets: 817#012TCP FIN Packets: 0#012TCP SYN/ACK Packets: 0#012Total TCP Packets: 843#012Total UDP Packets: 0#012Total ICMP Packets: 0#012Total other protocol packets: 0
[1:4000001:0] Snort Alert [1:4000001:0][Priority: 2]: {TCP} 172.20.10.128:43573 -> 192.168.78.132:135
10:12:01.312210 [!] Artillery has blocked (and blacklisted the IP Address: 172.20.10.128 for connecting to a honeypot restricted port: 25
[1:4000001:0] Snort Alert [1:4000001:0][Priority: 2]: {TCP} 172.20.10.128:43573 -> 192.168.78.132:5900
[1:4000000:0] Snort Alert [1:4000000:0][Priority: 2]: {TCP} 192.168.78.131:5900 -> 172.20.10.128:43573
[1:4000001:0] Snort Alert [1:4000001:0][Priority: 2]: {TCP} 192.168.78.130:135 -> 172.20.10.128:43573
[1:4000000:0] Snort Alert [1:4000000:0][Priority: 2]: {TCP} 172.20.10.128:43573 -> 192.168.78.131:5900
[1:4000001:0] Snort Alert [1:4000001:0][Priority: 2]: {TCP} 172.20.10.128:43573 -> 192.168.78.133:5900

Figure 9. Log events generated by Nova, Artillery and Snort

With this data, incident response teams should be able to clearly identify the compromised host on the 172.20.10.0/24 network. Several SIEM correlation rules could alert further based on these events, helping automate the start of an incident response process and prioritizing the event. For example, a SIEM could alert if logs are received from both Artillery and Nova classifying a single IP address as malicious. With this information as well as the captured Snort logs, incident responders could quickly review pertinent information to the attack. Reviewing all events and flows associated with the malicious IP address, other compromised systems may be identified through the resulting investigation. Furthermore, the indicators of

compromise identified in this investigation can be used to augment preventive controls to stop similar attacks from succeeding in the future.

A skilled attacker would not make as much clumsy "noise" on the network as shown in the above examples, but active defense systems limit the amount of reconnaissance and number of mistakes an attacker can make before being detected. They also slow down an attacker's ability to accurately map an internal network using active reconnaissance techniques after breaching perimeter defenses. Since these systems can be implemented on spare hardware and have limited to no negative impact on production networks, they can be a quick and easy win for network administrators to augment their defenses.

4. Conclusion

Although active defense techniques can be used on Internet-facing systems, their value may be limited since Internet-facing hosts are expected to be on the receiving end of continuous reconnaissance. On the other hand, active defense systems provide substantial value when placed in locations such as private networks where reconnaissance activities are less common, but are potentially more dangerous. Attackers using malware and client-side attacks do not need to assault public-facing systems directly, and once the initial exploit is successful, additional reconnaissance occurs within internal networks. Utilizing endpoints as a pivot point into private networks, attackers are able to use tunneling techniques in order to bypass egress filtering systems and conceal their presence. Acting as yet another layer of security, active defense systems can be implemented to specifically identify, alert on, and hinder this type of activity.

Internal systems providing active deception capabilities can increase the cost and time required for an attacker to successfully exfiltrate data. However, active defense, SIEM and IDS systems are significantly more useful when integrated together than when operating individually. When IDS alerts for honeypot IP addresses and ports are triggered, a single alert does not provide significant value and could be caused by a misconfigured system or a simple typo. On the other hand, when a SIEM can present one of these alerts with output from an active defense system as well as any other events associated with the potential offender, an analyst can quickly and efficiently begin researching and analyzing the event.

Josh Johnson, jcjohnson34@gmail.com

Applied to a kill chain model, active defense systems on private networks exist as an additional control within the phases of an intrusion in which attackers have already breached perimeter defenses. These systems provide defenders the additional ability to identify threats and prevent adversaries from taking actions on their objectives and exfiltrating data. Furthermore, the observation of internal reconnaissance activities can provide valuable intelligence on attackers' capabilities and motives as well as identifying the failures of other defenses and preventive controls. With this information, organizations can make better informed decisions on exactly where their security programs must improve in order to prevent similar attacks from succeeding in the future. The implementation of active defense systems and the integration of these systems into existing SIEM and IDS strategies can be the difference between a successful breach and a thwarted attack.

Josh Johnson, jcjohnson34@gmail.com

References

Bright, P. (2011, February 15). *Anonymous speaks: the inside story of the hbgary hack.* Retrieved from http://arstechnica.com/tech-policy/2011/02/anonymous-speaks-the-inside-story-of-the-hbgary-hack/

Deception toolkit. (n.d.). Retrieved from http://www.all.net/dtk/dtk.html

Haig, L. SANS Institute, (2002). *Labrea - a new approach to securing our networks*. Retrieved from website: http://www.sans.org/reading_room/whitepapers/attacking/labrea-approach-securing-networks_36

Holdaway, E. J. (2001). *Active Computer Network Defense: An Assessment* (No.AU/ACSC/055/2001-04). AIR UNIV MAXWELL AFB AL.

Hutchins, E.M., Cloppert, M. J., & Amin, R. M. Lockheed Martin Corporation, (n.d.). *Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains*. Retrieved from website: http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf

Kadrich, M. (2007). *Endpoint security*. (pp. 13-17). Upper Saddle River, NJ: Addison-Wesley.

Northcutt, S., & Novak, J. (2002). *Network intrusion detection*. (3rd ed., p. 159). Indianapolis: New Riders.

Robish, E., Johnson, K., & Strand, J. (2013, February 07).*Active defense harbinger distribution*. Retrieved from http://sourceforge.net/projects/adhd

Josh Johnson, jcjohnson34@gmail.com

Skoudis, E., & Liston, T. (2006). *Counter hack reloaded, a step-by-step guide to computer attacks and effective defenses*. (2nd ed., pp. 559-561). Upper Saddle River, NJ: Prentice Hall.

Anatomy of an attack from spear phishing attack to compromise in ten steps. (n.d.). Retrieved from https://www.mandiant.com/threat-landscape/anatomy-of-an-attack/

Developments of the honeyd virtual honeypot. (n.d.). Retrieved from http://www.honeyd.org/

Project artillery - the most advanced threat intelligence solution. (n.d.). Retrieved from https://www.trustedsec.com/downloads/artillery/

The nova project. (n.d.). Retrieved from http://www.projectnova.org/

Josh Johnson, jcjohnson34@gmail.com

Last Updated: September 23rd, 2017

Upcoming Training

SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS London February 2018	London, United Kingdom	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Northern VA Spring - Tysons 2018	Tysons, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced