Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at http://www.giac.org/registration/gcih

Responding to Zero Day Threats

GIAC (GCIH) Gold Certification

Author: Adam Kliarsky, adam.kliarsky@gmail.com
Advisor: Antonios Atlasis

Abstract

Used with increasing sophistication, 0day attacks have been essential in successful Advanced Persistent Threat (APT) style attacks making headlines recently. The problem is evident; incident handlers and response teams struggle to identify and respond to unknown threats. This is an issue that plagues organizations of all sizes that rely on signature-based detection mechanisms. Attempting to handle unknown threats without a systematic plan will fail. It is imperative that incident handlers and response teams have a methodology to be able to respond to unknown or unidentified threats to protect the critical assets and data that businesses rely on. While some vendors claim their product is the solution to identify unknown issues, relying on a single solution creates a single point of failure. With complex attacks and sensitive data, this single point of failure could be detrimental. This paper will discuss integrating specific techniques into the preparation, identification, and containment phases of incident response to address the current problem.

1. Introduction

The internet has become a pervasive threat vector to organizations of all sizes. As new technologies are adopted to keep pace with business trends, surreptitious sources lurk in the shadows to exploit the weaknesses exposed. Sophisticated, targeted attacks such as Aurora, APT, Stuxnet, and Night Dragon have been making headlines, with goals of monetary gain and intellectual property theft. Zero-day threats have been essential success factors in some of these attacks. One example of this is the Aurora attacks on Google et al. in 2010. "On January 14, 2010 McAfee Labs identified a zero-day vulnerability in Microsoft Internet Explorer that was used as an entry point for Operation Aurora to exploit Google and at least 20 other companies" (Operation Aurora, 2010). As SANS Incident Handler Marcus Sachs stated in a related Internet Storm Center diary post, "we need to start rethinking how we are going to defend our networks in the coming years and decades" (Sachs, 2010). Organizations are empowering their employees with mobile devices, tapping into the business potential of social networking, and taking advantage of the scalability and virtualization of cloud computing. But while these technological advancements increase business productivity, they also increase exposure and subsequent risk to the organization. Mobile computing and smart phones, for example, expand corporate borders beyond safeguards of the perimeter and internal controls. And more malware is being seen targeting these devices. According to Kapersky Labs, "In January 2011, Kaspersky Lab recorded 154 different mobile malware families with 1,046 strains, two per cent of which are already targeting Android Mobile" (Kaspersky Lab: sensitive corporate information is increasingly at risk from mobile malware, 2011). Social networking is another popular platform that facilitates a variety of threat vectors. With over 500 million active users, half of which are logging in each day (Facebook, 2011) it is clear why attacks target these users: the odds are good. From friend requests to viral videos, the user base loves to engage by clicking enticing links. Popular client-side attacks that take advantage of unsuspecting users facilitate malware propagation, bypassing perimeter controls and installing key loggers, bots, and other malicious software. Sophisticated attacks targeting inherent human and technological

Adam Kliarsky, adam.kliarsky@gmail.com

weaknesses combined with advanced 0day exploits will play important roles in successful attacks going forward.

Undetectable and for the most part unknown, the 0day threat presents an increasing new front on which incident handlers have to fight. The term zero-day (0day) refers to, for the most part, the amount of time the community has to respond to a newly discovered and/or disclosed threat. The community includes both home and corporate users, as well as security vendors. To better understand the 0day, it is imperative to understand vulnerability research and corresponding ethical disclosure practices. This topic is subjective both in definitions and ethics. Different researchers subscribe to different opinion, and handle the subject accordingly. Additionally, there exists some ambiguity due to the diverse group and decentralized governance involved. Security researchers of all calibers work diligently to discover new bugs in software products or network protocols. While motives range from ethical to malicious, the end goal is the same: discover vulnerabilities that expose risk. As figure 1 illustrates, the vulnerability research lifecycle is a process that starts with identifying the software vulnerability through static analysis, fuzzing, etc, establishing a 'proof of concept' (PoC) to demonstrate the existence of exploitability, then disclosure to the vendor, and subsequently the public. When a proven vulnerability (proven by existence of PoC or other exploit code) is released to (or used in) the public without prior vendor engagement, it is referred to as a 0day.
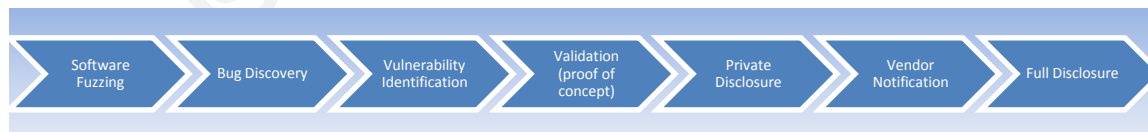

Figure 1: Vulnerability Research Lifecycle

The challenge that 0days present to information security teams is the gap in detection and identification capability. Since vendors have no prior knowledge of the 0day, signature-based systems, such as intrusion detection/prevention and anti-virus will not identify the threat. As incident response teams get inundated with signature based alarms daily for known threats, trying to identify a 0day is almost impossible. As a result, an incident may go undetected for some time. It is therefore a necessity to establish a

Adam Kliarsky, adam.kliarsky@gmail.com

solid, phased incident response plan and corresponding measures that can efficiently detect and identify a 0day, so that it can be mitigated as quickly as possible.

2. Incident Response: The Zero Day Approach

The traditional (and successful) incident response program is typically implemented using a phased methodology. This allows the lifecycle of incident response to be broken down into separate manageable components. While implementation varies and is contingent on the needs of the organization, there are two popular methods: one from the SANS Institute and another from the National Institute of Standards and Technology (NIST). The one taught by SANS (Figure 1) uses six phases that consist of 1) Preparation, 2) Identification, 3) Containment, 4) Eradication, 5) Recovery, and 6) Lessons Learned (Murray, 2007). The NIST version uses four phases that similarly consist of 1) Preparation, 2) Detection and Analysis, 3) Containment, Eradication, and Recovery and 4) Post-Incident Activity (Scarfone, Grance, & Masone, 2008).

SANS:

| Preparation | Identification | Containment | Eradication | Recovery | Lessons Learned |

NIST:

| Preparation | Identification & Analysis | Containment, Eradication, & Recovery | Post-Incident Activity |
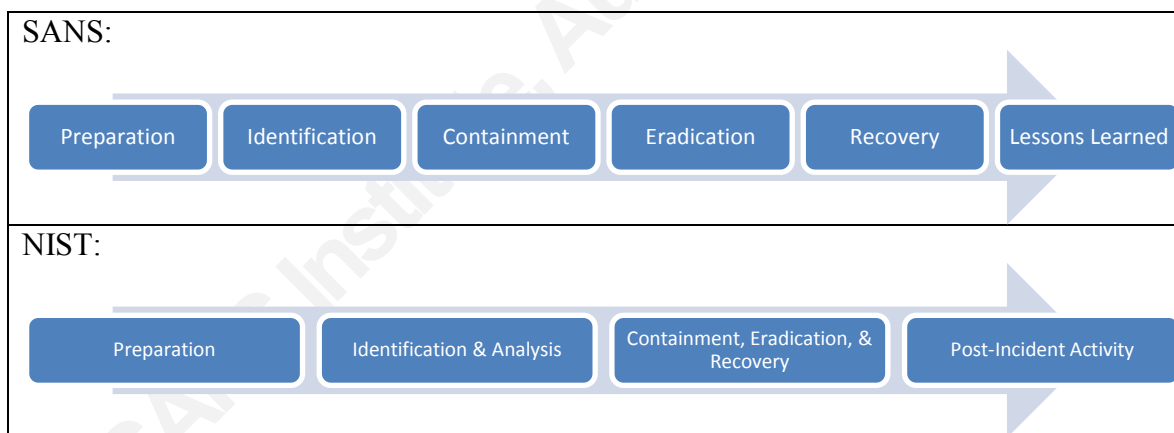
Figure 2: SANS and NIST Incident Response Models

For handling incidents where 0day exploits were used, the IRT may need a slightly modified approach, an approach that applies specifically to 0day based incidents. Such an approach, which actually enhances the typical incident handling methodology with some additional actions, is presented in this paper.
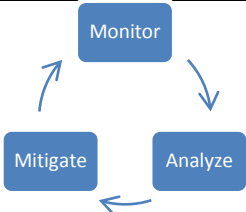
The phases that will have the most impact for 0day incident response will be preparation, identification/analysis, and containment. The preparation phase will situate the organization and response team to be able to respond effectively to a 0day. The identification phase is where the response team identifies the 0day as an incident and

Adam Kliarsky, adam.kliarsky@gmail.com

analyzes it to fully understand how to mitigate it. Containment is where the mitigation will be applied, in the form of host and network intrusion prevention, and other controls that limit the impact of a 0day threat.

The incident response team (IRT) should have a methodology to deal with these threats both proactively and reactively. The proactive response should focus on external threats; 0days that are announced to the public but haven't impacted the organization. The reactive response should focus on responding to a 0day compromise; an actual incident. The following methodology addresses both proactive and reactive in the preparation, identification, and containment phases.

Table 1 below illustrates the methodology for responding to externally announced threats to mitigate the potential threat before it becomes an actual incident:

Table 1: External Response: Proactively Mitigating Impact to the Organization
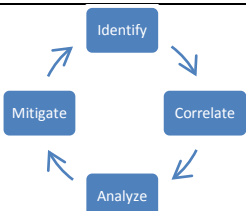
| | 1. Monitor
2. Analyze
3. Mitigate |

The proactive approach consists of the following:

1. Monitor – refers to the ongoing monitoring of public resources to identify 0day threats.
2. Analyze – refers to the analysis of the PoC (or weaponized exploit) in a lab environment to identify potential threat vectors and targets that impact the organization.
3. Mitigate – takes the information gathered from the analysis to build and implement mitigation mechanisms (IDP signatures etc).

Table 2 below shows the suggested steps to take with respect with regards to internal 0day incident response.

Adam Kliarsky, adam.kliarsky@gmail.com

Table 2: Internal Response to 0day Incidents

	1. Identify 2. Correlate 3. Analyze 4. Mitigate

The reactive internal response kicks off when suspicious activity is detected on or to corporate assets (reactive implies the team is responding to a compromise). This could be an influx of calls to the help desk about a seemingly common problem on similar hosts, intrusion detection system (IDS) alerts triggering around the same time as servers appear to have issues, or other related troubleshooting calls. The IRT will play a significant role in assembling the pieces of the puzzle together to identify an incident.

1. Identify – refers to the correlation of disparate logs, alarms, or other events that could signify an incident.
2. Correlate – takes the next step to further correlate host and network activity and isolate the malicious process.
3. Analyze – occurs once the threat has been identified and isolated, and provides the basis for mitigation.
4. Mitigation – the goal of this process is to identify the malicious activity and implement controls to contain the incident, and prevent further exploitation.

Understanding the steps involved in responding to 0day threats, the IRT can apply the steps in these methodologies to the phased response plan to maintain continuity within the organization.

3. Preparation

The preparation phase has two primary goals; to ensure incident response team readiness, and to ensure sufficient controls to mitigate security incidents (Scarfone, Grance, & Masone, 2008). To be ready, the IRT will need to be have 'eyes on' the

Adam Kliarsky, adam.kliarsky@gmail.com

internet at all times to see what's happening. It will also have to be able to react accordingly to ensure risk is mitigated. Furthermore, the IRT will need to ensure sufficient controls are in place to prevent and detect possible compromises. Success factors to augment existing controls will include anomaly detection mechanisms, as well as malware collection and analysis capability. To be effective, the incident handler will need to asses this phase from all sides, understand what and where to monitor and protect. This means having monitoring the public domain, and implementing detection mechanisms in place at the host and network levels. An internal compromise will have a few characteristics that incident handlers can look for; initial exploitation threat vector, backdoor or other covert communication back to the origin, and possibly a propagation component.

3.1. External Response: Handling 0day Advisories

Analyzing external advisories helps the IRT prepare for potential attacks. By understanding how a 0day works, what the target is, how it is exploited, the IRT can ensure controls are in place to prevent attacks (incidents) from occurring. This is accomplished by configuring a lab to mimic the production environment, then following the monitor → analyze → mitigate methodology which involves monitoring public resources for 0day announcements, replicating the attack, and deducing mitigation strategies.

3.1.1. Building an Incident Response Lab

To successfully respond to posted threats, the IRT must have a lab environment that simulates, as much as possible, the production environment it protects. The lab should consist of systems to simulate the role of the attacker, the victim, and a monitoring tool. This lab could be physical machines, or alternatively virtual machines. The benefit to a virtualized lab is the IRT can revert a virtual machine to a known state during the testing process, thus saving time in rebuilding. As for the configuration of the lab, the attacking machine should have tools, interpreters, and compilers to accommodate a variety of source code files associated with the 0day. The victim machines should reflect what is deployed within the organization (Windows XP, 7, MAC OS etc). The monitoring system should be able to capture traffic between the attacker and victim to

Adam Kliarsky, adam.kliarsky@gmail.com

analyze traffic traversing the network. This host should be some Unix/Linux system with both sniffer and IDS capability.
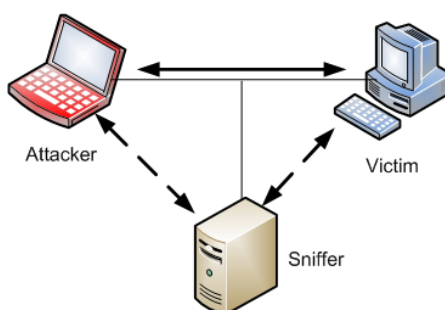
Figure 3: Basic Lab Configuration

3.1.2. Monitoring Public Resources

One essential component to being prepared is monitoring what's happening on the internet on a daily basis. As the internet threat landscape changes from day to day, the IRT needs to be constantly monitoring the internet 'weather'. By keeping an eye on public internet resources and observing new trends, types of attacks and vulnerabilities, the IRT will be aptly ready to respond to 0day threats. Websites, mailing lists, blogs/micro-blogs (Twitter), and vendor notifications are all vehicles for 0day notifications. The SANS Internet Storm Center (http://isc.sans.org) for example, is a great resource for such notifications. This site is manned by one of their volunteer incident handlers daily. The Internet Storm Center (ISC) monitors a variety of public resources, including logs received through DShield, "a distributed intrusion detection system for data collection and analysis" (About the Internet Storm Center). These logs come from a variety of internet-connected devices used by businesses and home users. "These devices feed the DShield database where human volunteers as well as machines pour through the data looking for abnormal trends and behavior" (About the Internet Storm Center). Another great resource is the Full Disclosure mailing list (http://lists.grok.org.uk/full-disclosure-charter.html). Created by Len Rose in 2002 (Cartwright) and hosted by Secunia, this list informally serves as a point of notification for a variety of security notifications, including 0day threats. Exploits can be published in a number of places, and typically with the announcement of a 0day, the associated proof of concept (PoC) or exploit will be linked in the announcement. Two common places to find exploits are Offensive Security's Exploit-DB (http://www.exploit-db.com/) and Packet Storm

Adam Kliarsky, adam.kliarsky@gmail.com

(http://packetstormsecurity.org/). The IRT should be able to monitor sources, identify a 0day when released, and be able obtain the PoC or exploit code for analysis.

3.1.3. Analyze the Threat: Replicating the Attack

Once a 0day is posted to a public resource, and the IRT confirms exposure to the organization, the team needs to be able to reproduce this in the lab environment to see what the potential impact would be. The first step in this analysis is to review the target software/application, version, and operating system the PoC was written for. The target needs to be configured so that it reflects the victim used in the PoC. Then the team needs to modify the PoC so that it is applicable to the environment. If the PoC uses hardcoded IP addresses, for example, this would need to be changed. If need be, the code is compiled to produce an executable exploit. Alternatively, if code is interpreted (Python, Ruby etc), the permissions need to be set to ensure the exploit is executable. The last step in attack replication is the monitoring system. The system should be running a sniffer to capture all packets in the exchange.

Once everything is setup, the exploit is launched against the target. The IRT team should verify the exploit works as expected. Confirming the attack was successful, the IRT can begin to identify unique characteristics of the attack to implement mitigation. Things like tcp/udp ports, packet payload, payload size, unique patterns within the payload, and byte offset of identifiers within packets are all useful.

3.1.4. Mitigation

Once the threat is analyzed, the IRT has enough information to start working on mitigation. If a particular tcp/udp port was used, this can be checked on perimeter firewalls to ensure it is blocked. Unique patterns in the network capture can be used to create signatures. Most buffer overflow exploits will use NOP commands embedded in the shellcode to ensure control of EIP, so those NOPs can be used in IDS signatures (assuming the attack isn't encrypted). Appendix A illustrates a hands-on example where an Easy FTP server exploit is analyzed to derive a signature for the popular open-source IDS Snort (http://www.snort.org).

Adam Kliarsky, adam.kliarsky@gmail.com

3.2. Internal Response: Preparing for a 0day Incident

The methodology for internal response follows the 'identify → correlate → analyze → mitigate' process. The preparation phase is critical to the success of this methodology, since identification relies on certain tools and monitoring to be in place (and subsequent steps rely on success of the predecessor). One thing the IRT should have ready to go is an incident response toolkit. This should be a read-only disc containing known trusted binaries, and ideally this would be bootable. There are some Linux based distributions, such as Helix (https://www.e-fense.com/store/index.php?_a=viewProd&productId=11) and Deft (http://www.deftlinux.net/) that can be used, or alternatively the IRT can create a custom disc.

Additionally, some things that should be configured to ensure successful identification will be system logging, network monitoring, host monitoring, the ability to collect malware, and application white listing – just to start. The incident handler and IRT need to be continually reviewing what could augment their ability to identify 0day threats.

3.2.1. Internal Log Monitoring & Aggregation

One of the important factors in securing a network is to setup a log monitoring mechanism. At the bare minimum, this should be a simple unix-based syslog server, but ideally a more intelligent security incident and event management (SIEM), capable of correlating information from those logs. Organizations can leverage enterprise SIEM solutions like Gartner's 'Magic Quadrant' candidates, Arcsight and RSA enVision (Therrien, 2010) if their budget permits. Or alternatively, go with an open source platform such as AlienVault's OSSIM (http://www.ossim.net) which stands for Open Source Security Information Management. All devices capable of sending logs to a remote system should be configured to do so. This will give an incident response team visibility across multiple connected devices at any given point of time. On the note of time, synchronized time is also critical to incident response, especially with regards to establishing an accurate timeline. If logs are captured out of order, critical incident response evidence may be missed. All systems in the environment should be configured

Adam Kliarsky, adam.kliarsky@gmail.com

to use a centralized time source such as network time protocol (NTP), ensuring consistency across disparate systems.

3.2.2. Monitoring Suspicious Network Activity

Identifying a 0day relies on system and network visibility. As malicious process traverse the network towards the intended target, network activity logged can provide crucial information. Malware propagation, command & control communication, and target proliferation are examples of network activity to watch for. Regardless of how or why, network activity can and will identify malicious behavior. There are a few tools that can augment existing network security systems, to identify anomalies associated with 0day activity.

Ourmon (http://ourmon.sourceforge.net/) is a Unix-based network monitoring and detection system that uses flow based collection and analysis to identify anomalies. Ourmon functions as a sniffer, using promiscuous interfaces to collect traffic flows between client and server. It analyzes data using Berkely Packet Filters and top talkers, then presents data as needed, in graphs, reports etc. As Ourmon attempts to separate traffic of interest from the rest, it employs a different method of flow analysis, extracting information of interest only. One benefit of Ourmon, is the ability to log DNS responses on a network. As some popular botnets have used Fast Flux DNS to avoid detection and blacklisting, Ourmon could pick it up. According to the Ourmon project page on Sourceforge (http://ourmon.sourceforge.net/), other capabilities include catching unknown mail relays, botnets, and best of all "spot infections with random 'zero-day' malware thingees" (ourmon - network monitoring and anomaly detection system). The basic Ourmon architecture consists of two components; a probe (like a packet collector) and the back end graphics engine which handles the reports. Both components can sit on the same box, depending on the organization deploying Ourmon – or probes can be deployed to monitor segments, feeding data back to the back end engine.

Netflow (http://en.wikipedia.org/wiki/Netflow) uses statistical data about client/server IP based flows to detect anomalies. Unlike traditional intrusion detection/prevention systems (IDP), flow based detection does not analyze payload (which might be moot in the case of a true 0day). Instead, layer 3 devices collect

Adam Kliarsky, adam.kliarsky@gmail.com

unilateral flows based on several fields in an IP header. One flow is established and tracked from client to server, and another in reverse from server to client. The advantage of this is the disposition the collecting device has – "Because flow data is coming directly from the router, a core element of any large network, NetFlow is capable of providing a unique view on the entire traffic of a network at the infrastructure level" (Gong, 2004). Netflow would enhance the ability to detect anomalies and potential 0day activity on the network.

BotHunter (http://www.bothunter.net/) is an application that was developed by SRI International to monitor communication between internal hosts and the internet with the purpose of identifying compromised machines. Funded "through the Cyber-Threat Analytics research grant from the U.S. Army Research Office" (About BotHunter), the application is free to download and licensed through SRI. What makes BotHunter an ideal tool to have for 0day identification, is the proprietary algorithm it uses in conjunction with a modified snort package to detect infections. The unique algorithm BotHunter uses is called network dialog correlation, and classifies flows between clients/servers as potential attack sequence steps. These steps, or dialog events, are then run through a correlation engine which builds a host profile and compare it to a malware infection lifecycle model; the closer the match, the higher the probability of infection. To add to the credibility of this application, it can be noted that "BotHunter first recognized Conficker data-exchange patterns back in November 2008, well before other security vendors picked up on the threat" (Vamosi, 2009).

The host profile BotHunter builds is calculated using an attack sequence consisting of the following elements: Infection I = <A, V, E, C, P, V', {D}>

Infection I:

A	Attacker
V	Victim
E	Egg Download Location
C	C&C Server
P	Peer to Peer Communication Points
V'	Victim's Next Propagation Targets
{D}	Set of Dialog Sequences

Adam Kliarsky, adam.kliarsky@gmail.com

The infection lifecycle uses correlated ingress/egress flows, applying weights to events. The host profile along with the malware infection lifecycle determines the situation to which a malware infection exists, whether other signature based security monitoring systems are aware or not (AV, IDP). "BotHunter is capable of declaring a host infected when either of three dialog sequence combinations is observed" (About BotHunter):

Condition 1:	Evidence of a local host infection, and evidence of outward malware coordination or attack propagation, or
Condition 2:	At least two distinct signs of outward bot coordination, attack propagation, or attacker preparation sequences are observed.
Condition 3:	Evidence that a local host has attempted to establish communication with a confirmed malware control host or drop site.

Darknets can also be used to identify anomalous or unauthorized traffic. Darknet is the term given to "IP address space that is routed but [contain] no active hosts and therefore no legitimate traffic" (Schiller & Binkley, 2007). Ingress traffic to a darknet would be an indication of anomalous activity, whether a configuration issue or malware propagation, and something the incident team would flag and investigate. Team Cymru, an internet security firm, sponsors 'The Darknet Project', providing guidance on deploying a darknet successfully (http://www.team-cymru.org/Services/darknets.html). Some components of a successful darknet include a sniffer, an upstream router configured for SNMP (for traffic statistics), Team Cymru recommends using a class C network, though "The more address space you allocate to the Darknet, the greater your visibility and statistical sampling" (The Darknet Project). Another option Team Cymru recommends, depending on the architecture (and related overhead this might cause), is route an organization's entire network to the darknet. Specific legitimate prefixes would be excluded and routed to proper destinations using an interior gateway protocol. This way any traffic not specifically destined to legitimate hosts will be picked up. Whatever log aggregation and correlation used, alerts should be configured to notify the IRT when ingress/egress traffic is detected. The incident response team will also want to ensure that

Adam Kliarsky, adam.kliarsky@gmail.com

the darknet isn't easily detectable, as most malware – and especially targeted 0days – will do some detection checking to avoid early analysis. Networks using bogon IP space or that are completely devoid of assets will be easy identifiers if ingress traffic is detected, but can be detected my malware and thus lose sustained compromised activity required for analysis. It is for this reason honeynets are useful.

Honeynets are networks consisting of multiple honeypots, designed to offer attackers a buffet of targets to choose from. From the perspective of network monitoring, the incident handler will see odd activity; scanning for targets, initial compromise, then subsequent pivoting and/or propagation. Honeynets work well to simulate an actual production network with several resources on the network. With one honeypot, the incident handler has a method to attract and capture malware, but it doesn't properly simulate the actual network environment, and thus the incident handler risks losing that key visibility gained with honeynets. The Honeynet Project (http://www.honeynet.org/) is a great resource for the development and research involving honeynets. As stated on their website, "The Honeynet Project is a leading international security research organization, dedicated to investigating the latest attacks and developing open source security tools to improve Internet security" (About The Honeynet Project). The Honeynet Project sponsors such popular projects as Honeywall, Sebek, Honeyd, and Nepentheses, among others. Honeywall (https://projects.honeynet.org/honeywall/) is a flagship project based on Fedora Linux used to build honeynets "for capturing, controling and analyzing attacks" (Spitzner, 2008). Used in conjunction with both high and low interaction honeypots, Honeywall acts as a layer two bridge/gateway for the honeynet. Individual honeypots can then be built and used as bait for attackers, while Honeywall captures all activity which can be used by the incident response team for analysis.

3.2.3. Monitoring Host Activity

In addition to monitoring the network, monitoring activity on individual systems will be critical to identify a 0day. Host monitoring is important for both detection and identification, as without it attacks can and will go unnoticed. There are a few technologies/products that can be used to identify anomalous activity. File level

Adam Kliarsky, adam.kliarsky@gmail.com

monitoring, host intrusion detection/prevention (HIDS/HIPS), and system logging are some examples.

Tripwire (http://www.tripwire.com) is a product designed with many system monitoring features including rules, policies, and the ability to customize as needed. The file system rules option allows a system to be baselined against a known good state, and alert on violations. Violations indicate unanticipated changes to files and/or directories. This could be unscheduled patches or upgrades, but can also indicate a compromise, exploit, rootkit, or other malware.

AIDE (http://aide.sourceforge.net/) stands for Advanced Intrusion Detection Environment, and is an open source file/directory integrity monitoring system, similar to Tripwire. Instead of using specific rules, AIDE stores hashed values of files and directories in a database, and runs tests against those known values. As with Tripwire, unexpected changes to the files/directories trigger alerts to notify the IRT that a potential compromise is taking place.

OSSEC (http://www.ossec.net) is a popular open source HIDS (host intrusion detection system). As stated on the website, "It performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response" (Welcome to the Home of OSSEC). A typical limitation of IDS in identifying a 0day is the reliance on signatures; however OSSEC adds a few features to compensate. These include "log analysis, integrity checking, Windows registry monitoring, rootkit detection, real-time alerting and active response" (Cid).

3.2.4. Malware Collection & Analysis

In order to respond to any type of malware, there needs to be a method to collect it. The incident response team needs to ensure they have the capability to capture malware, and to analyze it. Honeypots are systems designed to emulate servers of value which tend to be targets of malicious users and software. Honeypots are used to track hackers, identify new types of attacks, and collect malware. Computers built with the sole purpose of being a honeypot are considered high-interaction honeypots. They have a full OS installed, as well as applications needed to fulfill their role. Low interaction

Adam Kliarsky, adam.kliarsky@gmail.com

honeypots are systems designed to emulate the bare minimum, typically used with malware analysis.

Honeyd (http://www.honeyd.org/) is a lightweight honeypot daemon that can emulate a variety of virtual targets, running a variety of services. Using layer 2 weaknesses, it can listen for requests to neighboring hosts, and claim those addresses, directing the request to the honeypot. Configuration files allow Honeyd to emulate as needed; "any type of service on the virtual machine can be simulated according to a simple configuration file" (Provos, 2004). This type of monitoring is greatly beneficial to an IRT looking to identify unknown threats.

Dionea (http://dionaea.carnivore.it/) was developed as a replacement solution to Nepentheses, a solution used for malware collection and analysis. Dionea can be described as a system designed to "…trap malware exploiting vulnerabilities exposed by services offered to a network, with the ultimate goal of gaining a copy of the malware" (dionaea - catches bugs). Dionea uses libemu to not only identify shellcode in an exploit, but let the exploit run in a chrooted environment, allowing multi state exploits to reveal their true actions.

3.2.5. Application Whitelisting

Another popular 0day mitigation strategy that has gained momentum in recent years is application whitelisting. This is where the organization permits all known safe production applications to run, and blocks all others. This ideally would prevent arbitrary remote code execution, but requires some work up front to ensure business continuity. The primary benefit of application whitelisting, is that only known trusted applications will be permitted to run. The limitation on the other hand, is that if malware injects itself into whitelisted process memory, it will run within the space of the trusted/allowed application (assuming the trusted list uses application names/locations versus hash values).

Adam Kliarsky, adam.kliarsky@gmail.com

4. Detection and Analysis of 0day Incidents

What differs in the 0day incident response scenario is the level of difficulty of detection and identification. Whereas known threats trigger alarms on intrusion detection/prevention systems, 0days are technically unknown threats. Some vendors are fast at developing signatures to publicly announced 0days (as is the open source community), but what about the 0day that no one knows about? The incident handler and response team need to be able to take action based on other indications. Following the 'identify → correlate → analyze → mitigate' methodology, the team will be in a situation to do that. Detection and analysis will identify, investigate, and analyze the threat. Mitigation will occur in the containment phase.

4.1. Identify

In order to identify an incident, the IRT needs to gather events, identify potential signs of compromise, and investigate the events to correlate actionable items. Once the events have been correlated, the IRT will need to analyze them and determine first if there is an incident and then how to mitigate it.

The first step in identifying an incident, is analyzing events for potential signs of compromise. Odd log entries, network activity, calls to the helpdesk, or other anomalous activity can all lead to an incident. The IRT will need to take a step back, and analyze other events across disparate systems at that given timeframe. It's also important for the IRT to be in communications with the help desk, as certain issues may be evident via ticketing or troubleshooting. If a troubleshooting call takes place for a system down issue, this would be a good time to check other systems. What are the IDS showing? What about firewalls? Identifying different possible indications of compromise is like hunting for Easter eggs. Maybe there are more, maybe not. Monitoring tools setup in the preparation phase will be critical in identifying odd activity. The IRT needs to check device logs and help desk tickets on a regular basis to identify potential signs.

Device and system logs contain a plethora of data. The logs an IRT should be concerned with will be logs showing odd communication, such as outbound IRC attempts, connections to blacklisted IPs or domains, server based communication on workstation subnets, and successive failed login attempts. Anything suspicious that could

Adam Kliarsky, adam.kliarsky@gmail.com

have malicious outcome should be checked. If logs are aggregated (via a SIEM or other centralized logging mechanism) for all devices, such as firewalls, proxies, domain controllers, essential applications, databases, and other critical systems, the IRT will have an easier time correlating events. If not, this process could consume precious time, and the team might miss some events. Either way, logs are essential to the incident response process, and invaluable to 0day identification. The IRT should identify hosts that show up in logs doing odd activity, and investigate those hosts further.

End users are also good indicators of suspicious activity. Users surf message boards, social networking sites, click suspect links, and respond to phishing emails. Client side attacks leveraging 0day threats can exploit workstations without antivirus picking it up. Tickets or calls from users complaining about odd activity from their PC, such as intermittent network connectivity, responsiveness of their PC, hanging programs, pop-up messages, or anything else that might seem odd to the user should be investigated.

Once a host has been identified, the investigation needs to continue on the local system to further identify signs of compromise. The IRT should take steps upon initial contact with the host to preserve volatile data. In the book *Malware Forensics: Investigating and Analyzing Malicious Code* (Aquilina, Casey, & Malin, 2008), the authors outline a methodology for volatile evidence collection:

Adam Kliarsky, adam.kliarsky@gmail.com

Table 3: Volatile Data Collection Methodology (Aquilina, Casey, & Malin, 2008)

- On the compromised machine, run trusted command shell from an Incident Response toolkit

- Document system date and time, and compare to a reliable time source

- Acquire contents of physical memory

- Gather hostname, user, and operating system details

- Identify users logged onto the system

- Inspect network connections and open ports

- Examine Domain Name Service (DNS) queries and connected hostnames

- Examine running processes

- Correlate open ports to associated processes and programs

- Examine services and drivers

- Inspect open files

- Examine command line history

- Identify mapped drives and shares

- Check for unauthorized accounts, groups, shares, and other system resources and configurations using the Windows "net" commands

- Determine scheduled tasks

- Collect clipboard contents

- Determine audit policy

While this methodology is quite comprehensive, the IRT will have to make the determination based on the incident at hand exactly what needs to be done. As the IRT is looking for suspicious evens that might indicate an incident, not all actions may be needed.

The first action the IRT should take is acquiring an image of the current running memory. This preserves volatile data that could be critical in further analyzing the incident, such as processes, network connections, passwords used, and other relevant information. A variety of tools such as Mandiant's Memoryze, FTK Imager, EnCase, and open source 'dd' (and its derivatives) can be used. Incident response toolkits such as Helix have a full suite of trusted tools and include imaging utilities.

Adam Kliarsky, adam.kliarsky@gmail.com

Once the RAM has been captured, the IRT can start extracting details about the host's activities that could help identify the incident. This includes open network connections, running processes, current logged on users, and any other information about the current state of the host.

4.2. Correlate

After suspect events are identified and information gathered, the next step is to correlate events to determine the source of suspect activity. Connections identified in network logs should be correlated on the local host with processes to determine what the source is. For Windows workstation analysis, the IRT can use native Windows utilities or third party utilities, so long as they are trusted binaries run from a read-only source. For example, one quick solution is to run 'netstat –ano | findstr <port>' with <port> being the outbound port in question. Findstr is the Windows 'grep' equivalent and can parse through the immense netstat output to show relevant network connections. The 'ano' switches of netstat will identify; all open ports (a), in numerical form (n), and the owning process id (o). The command 'netstat –ano | findstr :80' would display all current http based connections with process id. Then 'tasklist' can be run, piping output once again through 'findstr' to limit results, showing the owning process:

```
C:\Users\victim>netstat -ano | findstr :80
TCP    10.10.10.220:31658    10.10.10.15:80         ESTABLISHED    6908
C:\Users\victim>tasklist | findstr 6908
chrome.exe                6908 Console              1    133,384 K
```

Sysinternals (http://technet.microsoft.com/en-us/sysinternals/bb545021) is a great suite of tools used to gather system information (included with some incident response tools, such as Helix). TCPView and tcpvcon are tools from the Sysinternals suite that show open network connections and the owning process. Psfile and Handle are both tools from the suite that lists open file activity. ProcDump, Process Explorer, and Process Monitor are Sysinternal process information tools. AutoRuns will list all programs that

Adam Kliarsky, adam.kliarsky@gmail.com

are configured to launch at boot, which includes malicious programs wishing to remain persistent.

Volatility (https://www.volatilesystems.com/default/volatility) is a nice tool to use for extracting sensitive data from a RAM image. This tool can correlate network activity with running processes, and additionally extract those processes for further analysis.

4.3. Analyze

Once the process is identified, the IRT will need to analyze it. It would be wise for the IRT to use the memory dump previously acquired to identify processes hidden from Explorer.exe, to ensure the team isn't missing anything. When analyzing a suspect process, the IRT will want to obtain such information like the executable that spawned the process, other child processes created, and any other process context information.

One thing the IRT should keep in mind is that malicious programs often try to remain hidden. Rookits with Trojaned binaries can be mitigated with trusted tools on read-only media. In the case of memory analysis, the IRT needs to be able to identify hidden processes as well. To check a RAM memory dump for hidden processes, the IRT can use Volatility to search the memory dump. To show hidden processes, Volatility's "psscan –f" option can be used as it "methodically scans a memory dump for the signature of an EPROCESS data structure" (Aquilina, Casey, & Malin, 2008) and then carves that out for analysis.

When the IRT begins analysis on the specific process, the team can dump the process memory for a particular process. Tools such as Volatility can dump a single process from a RAM dump captured in the detection and analysis phase. Another useful tool is Microsoft's User Mode Process Dumper (http://www.microsoft.com/downloads/en/details.aspx?FamilyID=E089CA41-6A87-40C8-BF69-28AC08570B7E&displaylang=en). One of the benefits of this tool is that it can dump a process without killing it. Dumping the process alone can help the team do isolated analysis, such as running strings on the process memory to identify unique values to be used for anti-virus signatures.

Adam Kliarsky, adam.kliarsky@gmail.com

Next, IRT will need to identify the executable that spawned the process, so host protection mechanisms can be updated and therefore able to prevent it from executing. Additionally, the team will identify child processes launched, as well as open files, DLLs, as well as associated user information. PrcView (http://www.teamcti.com/pview/prcview.htm) is a process viewer for Windows that shows the process to executable path mapping the IRT will need. Another useful tool is CurrProcess, by NirSoft (http://www.nirsoft.net/utils/cprocess.html). CurrProcess shows process information, such as name, process id, priority level, program location on disk, and memory usage.

Understanding the process on the host that is associated with the suspect traffic and then identifying the executable on the host that spawned the process will help the IRT contain the incident. Appendix B illustrates a hands-on example using the steps with the associated tools.

5. Containment

The containment phase is where information gathered in the previous phases can be used to prevent further spread of an incident. As the NIST Computer Security Handling Guide states, "When an incident has been detected and analyzed, it is important to contain it before the spread of the incident overwhelms resources or the damage increases" (Scarfone, Grance, & Masone, 2008). A robust security program consists of a defense-in-depth strategy, with a multi-layered approach to securing systems. As such, the IRT can apply mitigation techniques from an analyzed incident to different layers to ensure defense-in-depth is maintained.

5.1. Network Level Containment

Containing the incident at the network level will involve implementing blocks on network devices. While the IRT identified one particular instance of a 0day, there is a chance that other systems might be affected too. It's important to implement containment strategies across the network to prevent the incident from continuing action or propagating. In the detection and analysis, the IRT should have

Adam Kliarsky, adam.kliarsky@gmail.com

determined what network communication was involved. If the 0day involved a bot based infection, command and control (C+C) communication will be seen outbound to C+C servers. This traffic can be null routed on network routers, blocked using access lists on firewalls. Additionally, analysis of the threat can be used to write custom signatures for intrusion prevention systems (IPS) to block related communication (see Appendix A for example custom Snort signature).

5.2.   Host Level Containment

The same information gathered in the detection and analysis phase can be used to apply mitigation techniques for host level containment. The first thing the IRT will want to do to contain the incident is terminate running processes associated with the incident analyzed. Next, take action to prevent other instances from spawning throughout the organization. Workstations can be configured with firewalls (such as the Windows Firewall), or host intrusion prevention systems (HIPS). Additionally, some anti-virus programs allow custom anti-virus (AV) signatures to be created. If the AV vendor used does not, the IRT can engage the AV vendor directly with the 0day information, so that the vendor can produce a supported signature. Domain tools (or scripts) can be used to check target workstations for the presence of malicious executables found in the detection and analysis section above, and subsequently delete them.

6. Conclusion

Zero day threats are a challenge that incident response teams have been facing as long as software vulnerabilities have been exploited. While these threats will continue to exist and challenge incident response personnel, having a solid plan to respond will provide a means to prevent and respond to these incidents. 0day threats are presented to us in a couple of ways; an advance notification that provides details and validates the existence of vulnerability (validation through proof of exploit), or the undetected threat that bypasses signature based security controls. As such, the IRT needs an approach for each – a methodology that applies

Adam Kliarsky, adam.kliarsky@gmail.com

specifically to 0day incidents, similar to the one presented in this paper. Responding to the public posted 0days requires the team to have the ability to reproduce the posted threat to mimic the threat potential, and mitigating it. Responding to the internal 0day threat requires the ability to detect anomalies, correlate across disparate systems, identify the threat, and mitigate accordingly. By having a methodology, the IRT has a steady state (repeatable) process in which to conduct incident response to 0day threats.

Adam Kliarsky, adam.kliarsky@gmail.com

7.   References

About BotHunter. (n.d.). Retrieved May 02, 2011, from BotHunter :

> http://www.bothunter.net/about.html

About The Honeynet Project. (n.d.). Retrieved 04 28, 2011, from The Honeynet Project:

> http://www.honeynet.org/about

About the Internet Storm Center. (n.d.). Retrieved June 2011, from DShield:

> http://www.dshield.org/about.html

About the Internet Storm Center. (n.d.). Retrieved April 11, 2011, from Internet Storm

> Center: http://isc.sans.edu/about.html

Aquilina, J. M., Casey, E., & Malin, C. H. (2008). *Malware Forensics: Investigating and*

> *Analyzing Malicious Code.* Burlington, MA: Syngress Publishing, Inc.

b33f. (2011, 06 01). *Easy Ftp Server v1.7.0.2 Post-Authentication BoF*. Retrieved 06 12, 2011,

> from Exploit-DB: http://www.exploit-db.com/exploits/17354/

Cartwright, J. (n.d.). *[Full-Disclosure] Mailing List Charter*. Retrieved April 10, 2011, from

> Secunia: http://lists.grok.org.uk/full-disclosure-charter.html

Cid, D. B. (n.d.). *OSSEC - Wiki*. Retrieved May 2011, 16, from OSSEC:

> http://www.ossec.net/wiki/Faq:Whatis#1.01_-_What_is_an_HIDS_.28Host-

> based_Intrusion_Detection_System.29.3F

dionaea - catches bugs. (n.d.). Retrieved 05 06, 2011, from carnivore.it:

> http://dionaea.carnivore.it/

Adam Kliarsky, adam.kliarsky@gmail.com

dookie2000ca. (2010, February 14). *Easy~Ftp Server v1.7.0.2 Post-Authentication BoF*.

Retrieved June 06, 2011, from Packet Storm:

http://dl.packetstormsecurity.net/1002-exploits/easyftp-overflow.txt

Exploiting Adobe Flash Player on Windows 7. (2011, April 4). Retrieved May 31, 2011, from

Abysssec Security Research: http://www.abysssec.com/blog/2011/04/exploiting-

adobe-flash-player-on-windows-7/

Gong, Y. (2004, August 15). *Detecting Worms and Abnormal Activities with NetFlow, Part 1*.

Retrieved April 20, 2011, from Symantec:

http://www.symantec.com/connect/articles/detecting-worms-and-abnormal-

activities-netflow-part-1

Kaspersky Lab: sensitive corporate information is increasingly at risk from mobile malware.

(2011, January). Retrieved March 03, 2011, from Kaspersky Lab:

http://www.kaspersky.com/news?id=207576290

Operation Aurora. (2010, 01 14). Retrieved from McAfee:

http://www.mcafee.com/us/threat-center/operation-aurora.aspx

ourmon - network monitoring and anomaly detection system. (n.d.). Retrieved 4 18, 2011,

from Ourmon: http://ourmon.sourceforge.net/

Provos, N. (2004, October 10). *Honeyd General Information*. Retrieved 06 16, 2011, from

Honeyd: http://www.honeyd.org/general.php

Sachs, M. (2010, November 14). *Stuxnet Analysis*. Retrieved from Internet Storm Center:

http://isc.sans.edu/diary.html?storyid=9934

Adam Kliarsky, adam.kliarsky@gmail.com

Scarfone, K., Grance, T., & Masone, K. (2008, March). *Computer Security Incident Handling*

   Guide. Retrieved March 1, 2011, from NIST Special Publications (800 Series):

   http://csrc.nist.gov/publications/nistpubs/800-61-rev1/SP800-61rev1.pdf

Schiller, C., & Binkley, J. (2007). *Botnets: The Killer Web App.* Syngress Publishing, Inc.

Spitzner, L. (2008, August 14). *Honeywall CDROM*. Retrieved May 15, 2011, from The

   Honeynet Project: http://honeynet.org/project/HoneywallCDROM

The Darknet Project. (n.d.). Retrieved 04 24, 2011, from Team-Cymru: http://www.team-

   cymru.org/Services/darknets.html

Therrien, L. (2010, May 19). *RSA in Leaders Quadrant for Security Information and Event*

   Management. Retrieved 04 14, 2011, from EMC Press Release:

   http://www.emc.com/about/news/press/2010/20100519-01.htm

Vamosi, R. (2009, August 24). *Is Your PC Bot-Infested? Here's How to Tell*. Retrieved 05 05,

   2011, from PC World:

   http://www.pcworld.com/article/170546/is_your_pc_botinfested_heres_how_to_tel

   l.html

Welcome to the Home of OSSEC. (n.d.). Retrieved May 15, 2011, from OSSEC:

   http://www.ossec.net/

Adam Kliarsky, adam.kliarsky@gmail.com

8. Appendix A: External 0day Advisory Response Example: Easy Ftp Server Post-Authentication Exploit
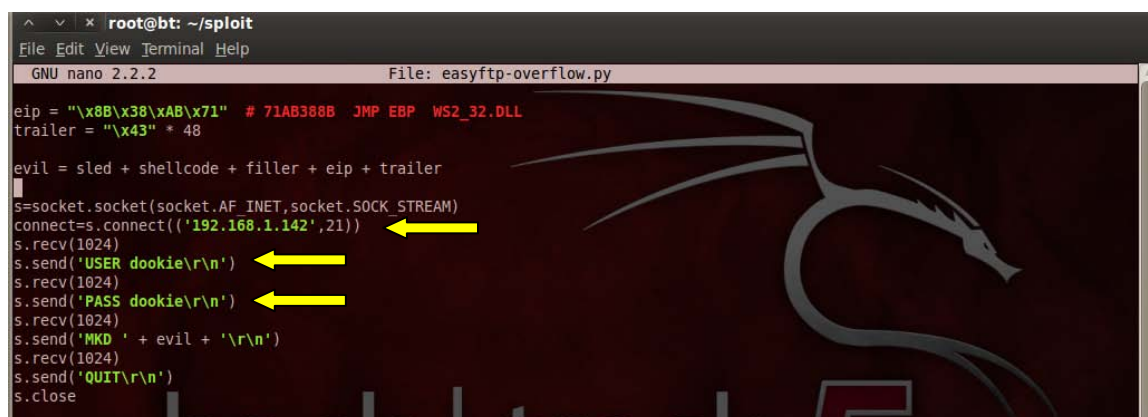
On February 14th, 2010 proof of concept for a vulnerability in Easy Ftp Server was published by Devon Kearns, aka 'dookie2000ca' (dookie2000ca, 2010). This post-authentication exploit was written for Easy~Ftp Server v1.7.0.2 and demonstrated code execution by running calc.exe on a Windows XP SP3 victim machine. For arguments sake, assume this was released as a 0day, with no vendor notification. As mentioned, the incident handler would need to step through the listed bullets; performing the initial assessment, then executing the code and observing the outcome.

In the initial assessment, the proof of concept is reviewed to make a determination of potential threat to the organization. If the organization is running this version of Easy Ftp Server is running on a Windows XP machine with service pack 3 installed, then it is assumed the threat exists. The next steps would be

- Obtain a copy of the code (PoC)

- Configure a machine to simulate an attacker

- Setup a victim Windows XP SP3 system running the vulnerable version of Ftp Server. Additionally, ensure the victim is running process, file, and registry monitoring utilities to identify when something happens to the vulnerable system.

- Configure a sniffer to monitor traffic between the attacker and victim.

Reviewing the PoC shows some static variables, such as target IP address and ftp user info. The incident handler will modify the code so that the code matches the victim. This will mean replacing the current IP address with that of the lab victim. The FTP

Adam Kliarsky, adam.kliarsky@gmail.com

server can be setup with the user/pass combination in the code.

Figure 4: Proof of Concept with set variables (IP Address, 'USER', and 'PASS')

Once the attacking machine is setup, the victim system needs to be configured as the advisory published. In this case a user 'dookie' with password 'dookie' is configured. This could be changed, so long as the PoC is changed as well. To keep it simple, it is recommended to follow the publisher's configuration as closely as possible.
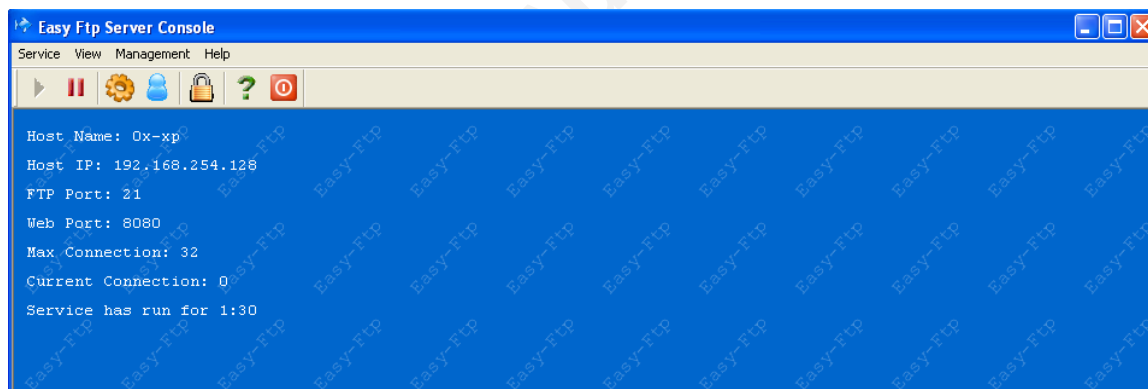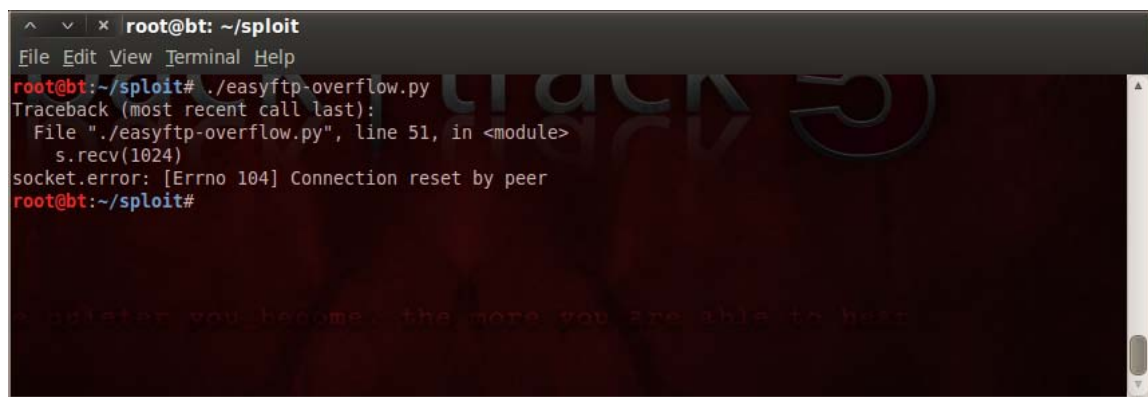
Figure 5: Victim Machine with Easy Ftp Server started

On the attacker machine, the PoC is modified with the victim FTP server's IP address and saved, then permissions are set to ensure it is executable. On the victim machine the server has been configured and is running. The sniffer is capturing traffic between the attacker and victim, with a specific capture filter to identify only the related traffic. The attack is setup and ready to go.
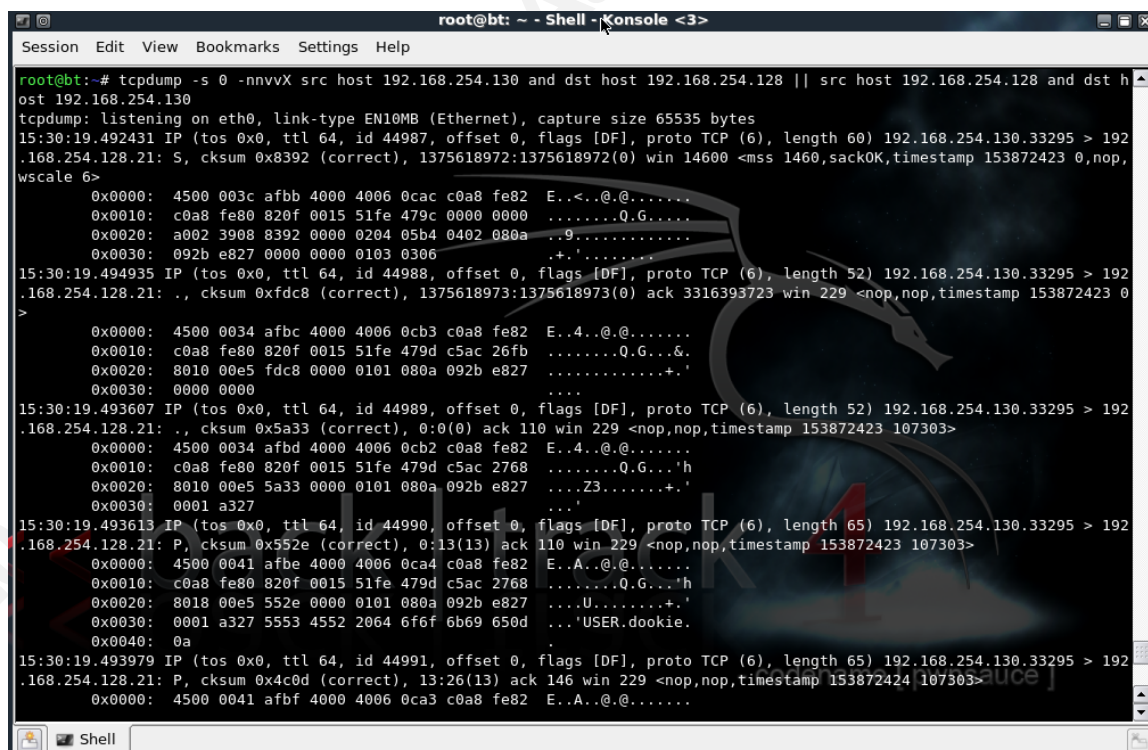
Adam Kliarsky, adam.kliarsky@gmail.com

Figure 6: Launching the Attack

The exploit executes, displaying a common error that happens when a buffer
overflow occurs and service stops. It is a good indication that the code executed as
expected.

Figure 7: Sniffer capture

The sniffer capture shows the initial exchange to login to the FTP service, followed by

Adam Kliarsky, adam.kliarsky@gmail.com

the MKD buffer overflow.
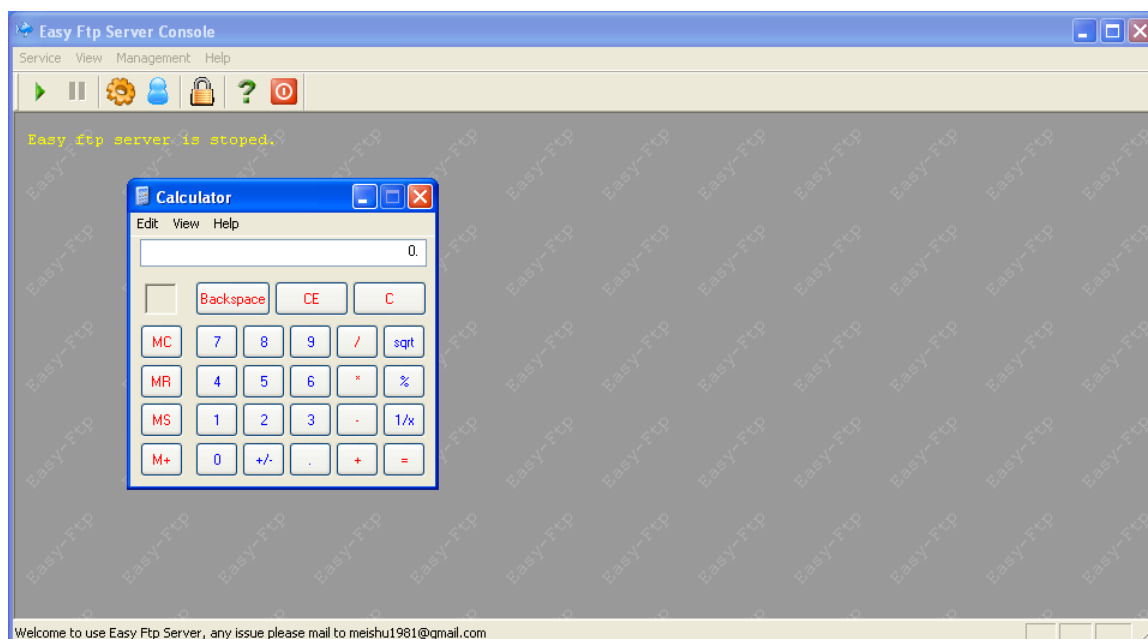
Figure 8: FTP service dies and calc.exe is launched

Finally we see the end result: the calc.exe payload that the author published.

This demonstrates that the vulnerability does exist, and will execute arbitrary code.

Looking at the PoC shows the encoder used and the payload generated:

```
# msfpayload windows/exec cmd=calc.exe R | msfencode -b
'\x00\x0a\x2f\x5c' -e x86/shikata_ga_nai -t c
# [*] x86/shikata_ga_nai succeeded with size 228 (iteration=1)

shellcode =
("\xd9\xcc\x31\xc9\xb1\x33\xd9\x74\x24\xf4\x5b\xba\x99\xe4\x93"
"\x62\x31\x53\x18\x03\x53\x18\x83\xc3\x9d\x06\x66\x9e\x75\x4f"
"\x89\x5f\x85\x30\x03\xba\xb4\x62\x77\xce\xe4\xb2\xf3\x82\x04"
"\x38\x51\x37\x9f\x4c\x7e\x38\x28\xfa\x58\x77\xa9\xca\x64\xdb"
"\x69\x4c\x19\x26\xbd\xae\x20\xe9\xb0\xaf\x65\x14\x3a\xfd\x3e"
"\x52\xe8\x12\x4a\x26\x30\x12\x9c\x2c\x08\x6c\x99\xf3\xfc\xc6"
"\xa0\x23\xac\x5d\xea\xdb\xc7\x3a\xcb\xda\x04\x59\x37\x94\x21"
"\xaa\xc3\x27\xe3\xe2\x2c\x16\xcb\xa9\x12\x96\xc6\xb0\x53\x11"
"\x38\xc7\xaf\x61\xc5\xd0\x6b\x1b\x11\x54\x6e\xbb\xd2\xce\x4a"
"\x3d\x37\x88\x19\x31\xfc\xde\x46\x56\x03\x32\xfd\x62\x88\xb5"
"\xd2\xe2\xca\x91\xf6\xaf\x89\xb8\xaf\x15\x7c\xc4\xb0\xf2\x21"
"\x60\xba\x11\x36\x12\xe1\x7f\xc9\x96\x9f\x39\xc9\xa8\x9f\x69"
"\xa1\x99\x14\xe6\xb6\x25\xff\x42\x48\x6c\xa2\xe3\xc0\x29\x36"
"\xb6\x8d\xc9\xec\xf5\xab\x49\x05\x86\x48\x51\x6c\x83\x15\xd5"
"\x9c\xf9\x06\xb0\xa2\xae\x27\x91\xc0\x31\xbb\x79\x29\xd7\x3b"
"\x1b\x35\x1d")

sled = "\x90" * 10
```

Adam Kliarsky, adam.kliarsky@gmail.com

```
filler = "\x90" * 30
eip = "\x8B\x38\xAB\x71"  # 71AB388B  JMP EBP  WS2_32.DLL
trailer = "\x43" * 48
```

A little research by the incident handler or response team could result in a payload that delivers a reverse shell back to the attacker.

Now that the potential impact has been understood, further analysis needs to be done to identify how to mitigate the threat. The sniffer capture shows the following packet as containing the buffer overflow exploit.

Figure 9: Tshark output showing authentication sequence followed by attack

What jumps out here is the MKD command in the FTP request and the string of C's at the end. These could be clear identifiers in an IDS signature. Other characteristics of this packet that could be used in an IDS signature would be the fact that it's TCP, destination port of 21, it is a post-authentication exploit, so it would follow in an established session after the 'USER' and 'PASS' parameters were sent. While it is always faster to save network captures via command line with such tools as tcpdump, Wireshark is better used to display and understand what is happening. The following displays the sequence of packets exchanged between the attacker and victim:

Adam Kliarsky, adam.kliarsky@gmail.com

Figure 10: Wireshark display of attack sequence

Figure 11 shows the sequence of packets and allows an analyst to step through each request/response to identify notable characteristics. The first three packets are the TCP handshake. This is followed by the FTP server banner, an acknowledgement, then login with 'USER dookie' followed by 'PASS dookie'. Line 135 begins the MKD buffer overflow, in the 1st byte of the FTP payload, which tshark displays cleanly:

Figure 11: tshark output showing exploit payload

So now an IDS signature can be built using the information gathered. Snort is a popular open source intrusion detection/prevention system that gets a lot of

Adam Kliarsky, adam.kliarsky@gmail.com

support from the community. Its open development platform allows analysts to create and modify IDS signatures, with global support from other users.

An example Snort signature would have the basic information:

Field	Value
Action	alert
Protocol	tcp
Source IP	192.168.254.130
Source Port	any
Direction	client > server
Destination IP	192.168.254.128
Destination Port	21
Message	Description
Flow	Established connection to server
Content	MKD

Figure 12: Snort signature options

Resulting in the following basic signature:

```
alert tcp 192.168.254.130 any -> 192.168.254.128 21 (msg:"FTP MKD
buffer overflow attempt"; flow:to_server,established; content:"MKD")
```

To make this effective, a full analysis should be done to identify patterns/behavior unique to this packet to correctly identify a future attack but limit false positives. Adding more to the content review, the following signature can be used to test for the exploit:

```
alert tcp 192.168.254.130 any -> 192.168.254.128 21 (msg:"FTP MKD
buffer overflow attempt"; flow:to_server,established; content:"MKD";
content:"253qCCCCCCCCCCCCCCCCCC")
```

By understanding the vulnerability and how an attack would exploit it, the incident response team can take preventative measures such as IDS rule development to alert and block when the attack signature fires.

Adam Kliarsky, adam.kliarsky@gmail.com

9. Appendix B: Internal 0day Detection and Analysis Example

The attack surface for workstations (users) is large, as users browse websites for a wide variety of reasons (both business and pleasure). This includes the use of chat and messenger programs, watching viral videos posted on message boards, and keeping up to date with friends and family on social networking sites. And let us not forget those phishing emails that entice users to click a link, or open a suspect attachment, like delivery confirmation emails:
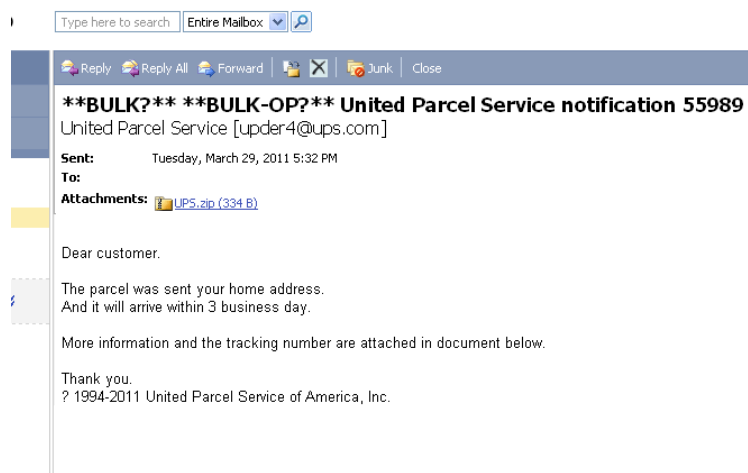
Figure 13: Sample email with suspect attachment

Given a situation where a user received an email, opened attachment, and subsequently opens a ticket for computer performance issues, the incident handler will want to investigate further. The immediate concern is, is existing protection and security monitoring doesn't alarm on this action. This is the behavior an IRT can expect with 0day incidents.

Before the IRT does any investigation on the workstation, the current state of the machine should be captured to ensure preservation of volatile data. The methodology of this paper recommends imaging RAM as a first step.

Adam Kliarsky, adam.kliarsky@gmail.com

Figure 14: RAM Acquisition with Helix

Next, the IRT will review any related activity to this workstation to correlate a specific process that will identify the origin. Reviewing network logs, the IRT can identify anomalous activity that the workstation was involved in. For example, some malware may use some specific method of communication to external servers, such as internet relay chat (IRC) channels. If the IRT sees network logs from the suspect workstation using tcp 6667 which is associated with IRC, that would be something to follow up on.

On the local system, the IRT can use tools to identify the source of this communication. As malware may use techniques to hide malicious processes, the IRT should run all commands from a trusted source. As previously mentioned, Sysinternals is a suite of system tools used for administration, forensic analysis, and

Adam Kliarsky, adam.kliarsky@gmail.com

other day-to-day tasks. TCPView is a tool from this suite that will let the IRT

correlate the suspect traffic to a process.

Figure 15: TCPView showing process associated with TCP 6667

The IRT will check the RAM dump to ensure there isn't additional processes

running that may have been hidden from Windows Explorer. Using Volatility, the

IRT will identify all processes:

Figure 16: Volatility with the 'psscan –f' option

Adam Kliarsky, adam.kliarsky@gmail.com

With the process name and ID confirmed, the IRT should conduct a mapping of process to executable on disk, to identify the origin. As mentioned, the process viewer PrcView.exe can be used for this task
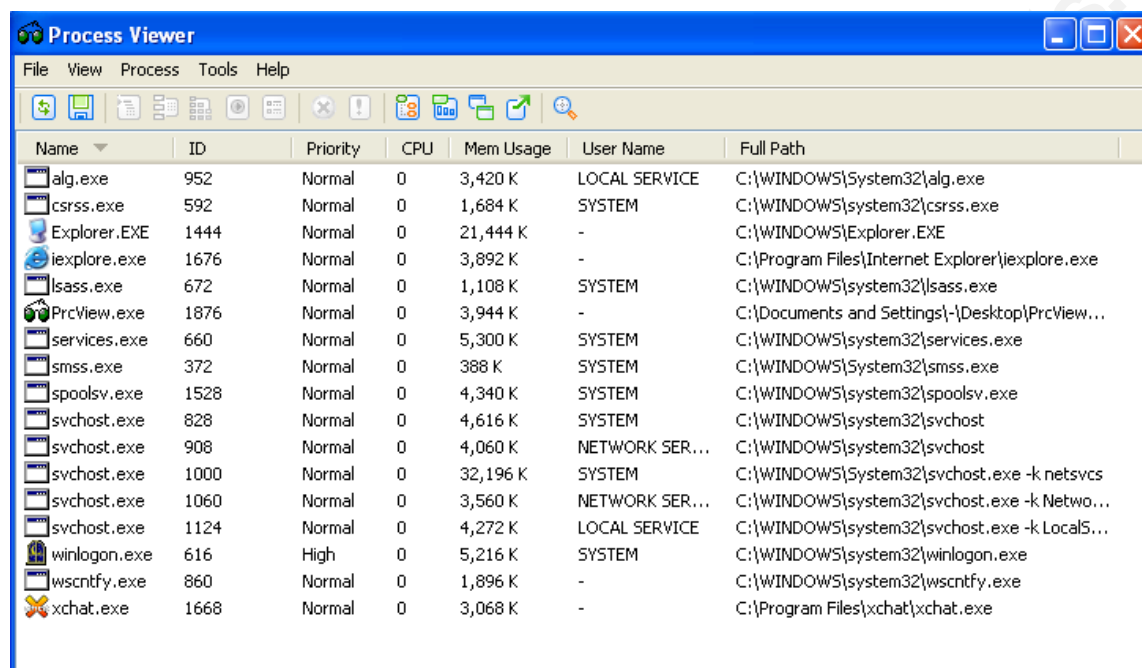
Figure 17: PrcView shows the executable path that spawned the xchat process

Additionally, CurrProcess can be used to identify other details of the process:

Figure 18: CurrProcess showing process details

Adam Kliarsky, adam.kliarsky@gmail.com

The information obtained during the detection and analysis can be then used to contain the incident and prevent similar incidents in the future or to other systems.

Adam Kliarsky, adam.kliarsky@gmail.com