Global Information Assurance Certification Paper

Efficiently Deducing IDS False Positives Using System Profiling

GCIA Gold Certification

Author: Michael Karwaski, mkarwaski@gmail.com

Adviser: Brent Deterding

Accepted: <>

Abstract

It is all too often modern day security analysts are plagued with security events that are irrelevant to a targeted host. Current applications and technologies attempt to eliminate these events by means of manually disabling and altering IPS/IDS rulesets. While this technology works, it does not provide an automated process for distinguishing the higher priority events from the low/irrelevant security risks. This paper is aimed at describing how to create a simple, static inventory database, then comparing security alerts to see if they relate to the host in question. This will allow for greater visibility into which alerts are actually relevant to the end users network.

© SANS Institute 2009, Author retains full rights.

1. Introduction

This paper is aimed at providing an in-depth look at system profiling, and how correlating this information with intrusion detection platforms will decrease the chance of witnessing irrelevant network alerts, and increase the chance of escalating true positive network alarms, while assigning a low risk threat to the alerts that do not relate to the host in question. We will look at how keeping records of every system on a network can help in identifying whether activity from the host is normal or abnormal. These records will include everything about the system that could generate network traffic. This will be done by generating a sample corporate lab environment, containing several server configurations and workstations, along with networking devices. We will look at identifying normal and abnormal communications between these hosts using a combination of intrusion detection platforms and profiles of all devices on the network.

In order to prove the profile correlation concept, several open source tools will be used to create network traffic and monitor the behavior. We will also use a combination of operating system/hardware configurations to show how each device, though similar in application, can produce very different results. We will then look at how using a profile can assist a security analyst in determining if a certain host is infected or whether the traffic produced is normal behavior. This will be done by comparing three separate databases and inserting results based on the elimination of irrelevant network alarms.

2. Background

Security analysts know all too well how noisy a networked environment can be these days. It is all too often that we are required to dig deep into our analytic skills and research traffic patterns for abnormal (unexpected) behavior when we don't have a solid standing point on where to start the analysis. With the ever expanding collection of applications, hardware configurations, network appliances, and bandwidth, we are witnessing noisy environments like none before.

When we apply an intrusion detection system to a network, we all have one common goal in mind; increase visibility into abnormal activity and respond in an immediate fashion once detected. This is becoming a chore to the common network security professional, as he is faced with an increase in intrusion alerts that fire on behavior that was not expected by the intrusion detection system (IDS), but possibly expected by the other applications residing on the network.

New installments of IDS devices with little to no signature tuning tend to be quite noisy right out of the box. Most of the alerts that get triggered are irrelevant to the network traffic, or are low threat. Analysts need a method for extracting the pertinent alarms and taking action to quickly address and remediate them. This requires attention to the comparison of normal network behavior to the addition of noise and abnormal activity. Being able to decipher and extract the differences has been an ongoing issue to security professionals around the globe.

While tuning a system over time is done, one will need an efficient method to process and rule out the large amount irrelevant alarms that get triggered and review them for possible signature tuning or removal from the IDS altogether. While there are many techniques for dealing with this, one of the greatest tools (in which this paper will discuss) is system profiling, and the correlation to IDS events to affected applications and services. By correlating generated IDS events to the relevant (installed) products, one can eliminate the high irrelevancy rate, and increase the true positive network alarms that get displayed to the end user.

3. Proof of Concept – Lab Design

3.1. Lab Overview

This brings us to our experiment and proof of concept where we will see the benefit of such a system. In the following section we will set up a simple network with several lab machines and an IDS to monitor the network. We will focus on internal network traffic only (as this is where the profiling will help us out the most) and identify what normal vs. abnormal behavior is. We will then attempt to correlate the detected activity with system profiles to determine the validity of the alert. From this determination we can then proceed to adjust the signature, or eliminate it from the system. The following is an outline of what type of information we can expect to gain from both a.) IDS events and b.) A store of system profiles. The goal here is to provide a correlation mechanism and to prove its usefulness in the networking arena.

- What detail does the IDS alert give us?
  - Source IP(s)
  - Destination IP(s)
  - Suspected activity
  - Possible payload
  - Timestamps
  - CVE Information
  - Type of traffic
    - Protocols

- What do profiles tell us?
  - Hostname
  - IP address
    - If available in a static environment
  - MAC address(s)
  - Authorized applications

- Authorized protocols
- Location
  - Physical location of the device
- Environment
  - what is the device used for
- Operating system
  - Type
  - Version
- Possible services running
  - Authorized ports

What we would like to accomplish, is to eliminate the alerts that do not apply to a system, based on its profile. In order to accomplish this, we will need one more piece of information; a vulnerability database. This third set of information will provide us what applications and service versions are actually vulnerable.

3.2. Database Design

Suppose we receive an alert from our IDS device telling us that an attacker is trying to exploit host x on our network. Within the IDS alert, we are provided with the CVE (Common Vulnerabilities and Exposures) ID that we may reference to pull all the details about a vulnerability, including affected products and related versions. From here, we can compare the affected products to our system profiles, and see whether the product is actually vulnerable, and if the alert generated is a true positive. If the alert is deemed as non-applicable, the alert can be disregarded and suppressed from the view of the end user.

Until recently, the vulnerability database would be constructed by in-house contributors, working to populate a list of applications and their associated threats. The end database would take lots of time and effort to maintain and keep current. Luckily for us, the Open Source Vulnerability

Database (OSVDB) was created, and is being actively maintained by over 4,600 researchers, spanning 25,683 products, and covering over 58,000 vulnerabilities. (Open Source Vulnerability Database, 2008)

The OSVD can be downloaded from their website and imported into several different databases, including MySQL, which we will use in the following experiments to tie in the correlation of alerts vs. the affected profile database store. The OSVDB schema can be seen in the following figure 1.2. As you can see, it lists references to vulnerabilities, and links to the affected products. This is a crucial step in our quest for eliminating false positive IDS events.

In figures 1.1 and 1.3, you can see the other two databases, constructed in MySQL and used in the experimental design. The Snort database schema is the default database used in the Snort db output configuration, while the inventory database is a simple user-defined database created for the purposes of this proof of concept design.

The Inventory Database (profile stores)

```
+----------------------
| Tables_in_inventory
+----------------------
| servers
| workstations
| network_devices
| applications
| application_map
| device_criticality
| criticality_map
+----------------------
```

The Open Source Vulnerability Database

```
+----------------------
| Tables_in_osvdb
+----------------------
| authors
| classification_items
```

```
| classification_types
| classifications
| credits
| ext_reference_types
| ext_references
| object_affect_types
| object_correlations
| object_links
| object_products
| object_vendors
| object_versions
| vulnerabilities
+----------------------
```

Snort Alert Database

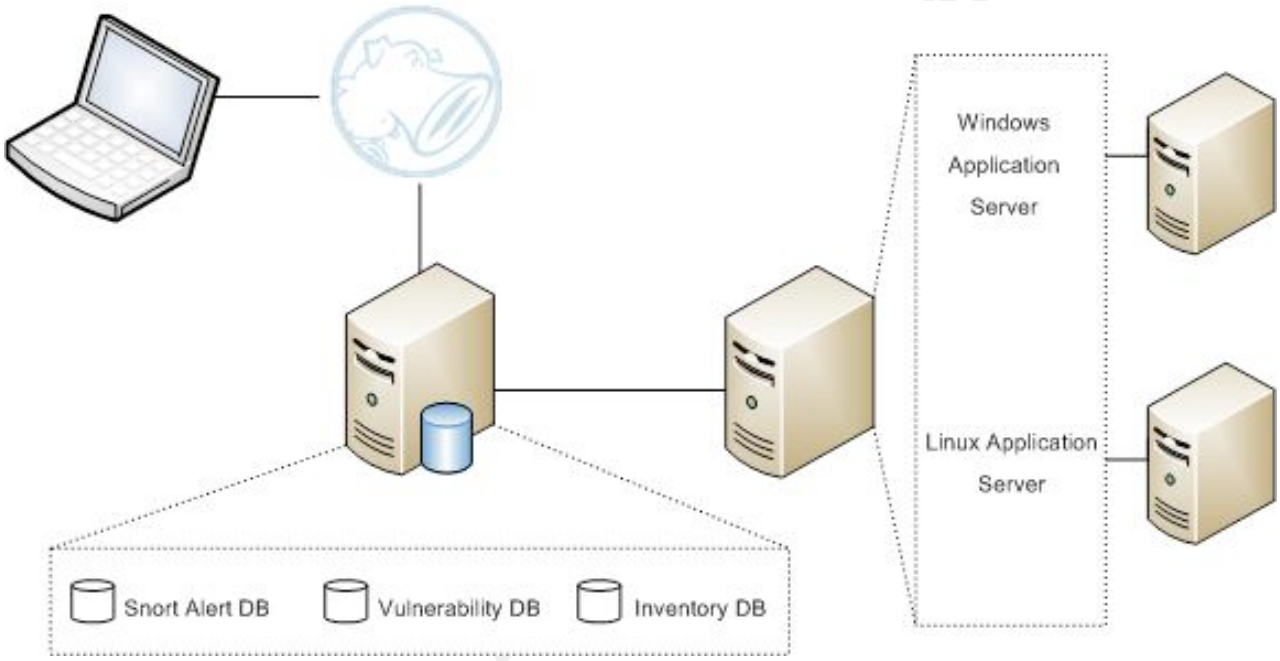
```
+------------------
| Tables_in_snort
+------------------
| acid_ag
| acid_ag_alert
| acid_event
| acid_ip_cache
| base_roles
| base_users
| data
| detail
| encoding
| event
| icmphdr
| iphdr
| opt
| reference
| reference_system
| schema
| sensor
| sig_class
| sig_reference
| signature
| tcphdr
| udphdr
+--------------------
```

3.3. Lab Setup

Now that we have an idea of what will be correlated, we can start generating events and looking to see if they apply to the targeted system. In order to accomplish this task, a lab environment has been setup containing the following devices and applications:

1) A scanning host running GFI LANGuard
    a. This will be used to scan and trigger IDS events
2) A Snort IDS box / Database store
    a. This host will store our three databases
        i. Inventory database (profiles)
        ii. Snort alert database (snort alerts)
        iii. Vulnerability database (correlation to profiles)
3) A virtual machine host serving two hosts
    a. Windows Application Server
    b. Linux Application Server

The entire network topology can be visually described by looking at the following network topology.



In order to ensure the scanning host would generate and trigger IDS events, services were enabled on the application servers, so that the scanning engine attempted to communicate with the open ports. The following is a description of each of the targeted hosts, and what service/ports are opened on them.

Windows Application Server

Windows XP Professional
Service Pack 2
Version 2002
Hostname: windowsLab
IP Address: 192.168.1.111
Open Ports:

```
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-term-serv
```

Linux Application Server

Ubuntu Server 4.3.2-1
Hostname: ubuntuLab
IP Address: 192.168.1.113
Open Ports:

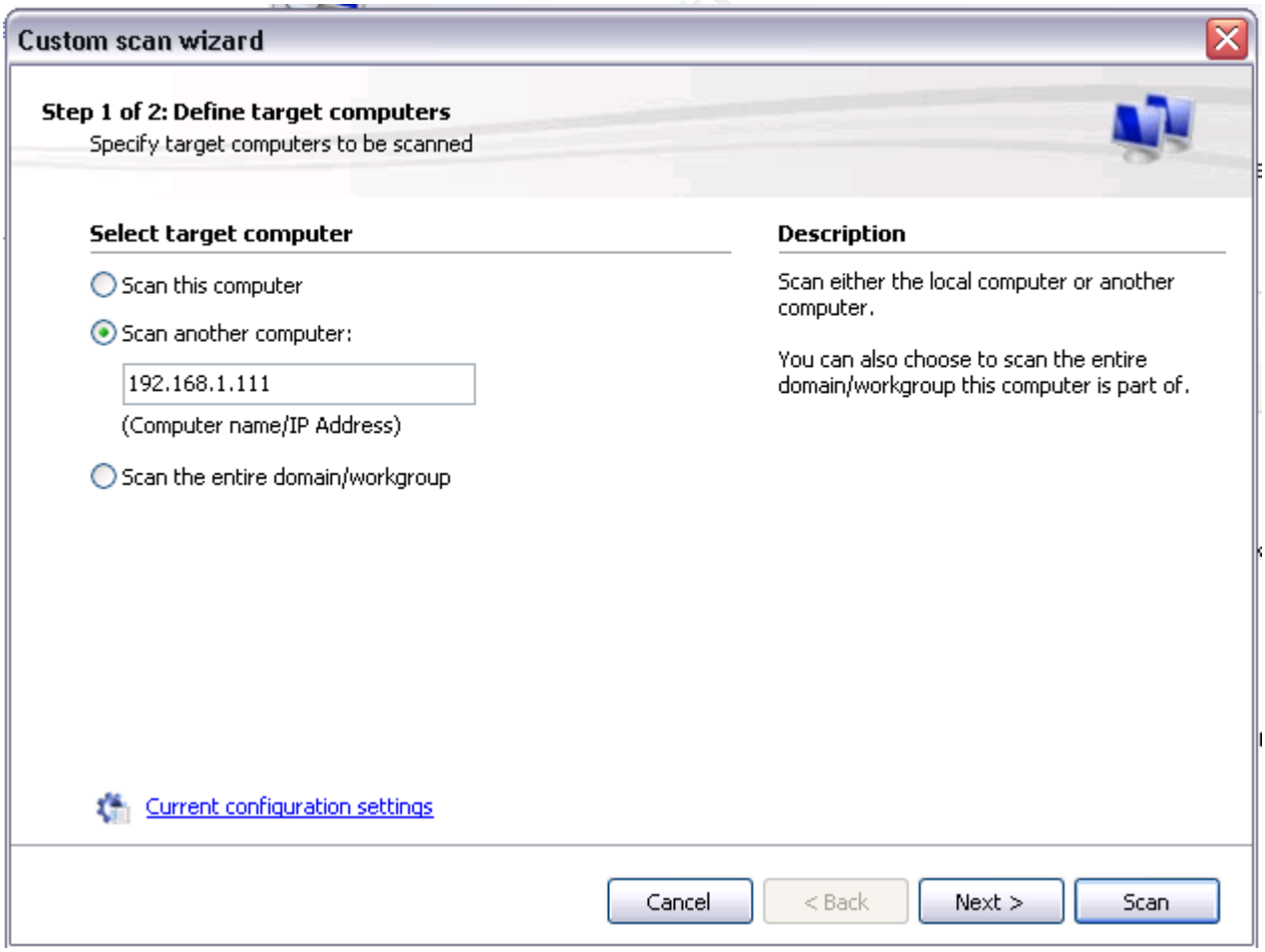
```
PORT     STATE SERVICE
22/tcp   open  ssh
53/tcp   open  domain
80/tcp   open  http
110/tcp  open  pop3
139/tcp  open  netbios-ssn
143/tcp  open  imap
```

445/tcp open microsoft-ds
993/tcp open imaps
995/tcp open pop3s

3.4. Generating Results from a Vulnerability Scan

3.4.1. Scan Setup

With the hosts setup and configured to respond on the network, we can now begin the scanning with our GFI Languard scanning software. We will first scan the Windows application server, followed by the Linux host. Once the scanning is complete, the Snort alerts will be checked via the Basic Analysis and Security Engine (BASE) web-based tool for ease of viewing the Snort alert database.



3.4.1. Windows Server Results

The following screenshot shows what alerts were triggered from the vulnerability scan from GFI Languard. The alerts are taken from the BASE display.

#14-(1-5132)[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP request tcp	2009-10-17 16:53:58
#15-(1-5133)[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP trap tcp	2009-10-17 16:53:58
#16-(1-5124)[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP private access udp	2009-10-17 16:53:41
#17-(1-5134)[bugtraq] [snort] TFTP NULL command attempt	2009-10-17 16:54:02
#18-(1-5135)[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP trap udp	2009-10-17 16:54:02
#19-(1-5125)[arachNIDS] [snort] ICMP L3retriever Ping	2009-10-17 16:53:43
#20-(1-5126)[arachNIDS] [snort] ICMP L3retriever Ping	2009-10-17 16:53:45
#21-(1-5127)[arachNIDS] [snort] ICMP webtrends scanner	2009-10-17 16:53:45
#22-(1-5131)[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP trap tcp	2009-10-17 16:53:58
#23-(1-5130)[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP request tcp	2009-10-17 16:53:58
#24-(1-5116)[url] [nessus] [cve] [icat] [bugtraq] [snort] MISC MS Terminal server request RDP	2009-10-17 16:35:32
#25-(1-5117)[url] [nessus] [cve] [icat] [bugtraq] [snort] MISC MS Terminal server request	2009-10-17 16:35:32
#26-(1-5118)[url] [nessus] [cve] [icat] [bugtraq] [snort] MISC MS Terminal server request	2009-10-17 16:35:33
#27-(1-5119)[url] [snort] MISC MS Terminal Server no encryption session initiation attempt	2009-10-17 16:35:33
#28-(1-5120)[url] [nessus] [cve] [icat] [bugtraq] [snort] MISC MS Terminal server request RDP	2009-10-17 16:42:11
#29-(1-5121)[url] [nessus] [cve] [icat] [bugtraq] [snort] MISC MS Terminal server request	2009-10-17 16:42:11
#30-(1-5122)[url] [nessus] [cve] [icat] [bugtraq] [snort] MISC MS Terminal server request	2009-10-17 16:42:11
#31-(1-5123)[url] [snort] MISC MS Terminal Server no encryption session initiation attempt	2009-10-17 16:42:11
#32-(1-5128)[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP request tcp	2009-10-17 16:53:57
#33-(1-5129)[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP trap tcp	2009-10-17 16:53:57

In looking at the results of the first scan against our Windows server, some of the immediate alerts that stand out, are the MS Terminal Server events related to the CVE 2001-0540:

"Memory leak in Terminal servers in Windows NT and Windows 2000 allows remote attackers to cause a denial of service (memory exhaustion) via a large number of malformed Remote Desktop Protocol (RDP) requests to port 3389."

(CVE Editorial Board, 2001)

This particular vulnerability applies to the following applications per the MS:MS01-040 security bulletin from Microsoft:

Affected Software:

- Microsoft Windows NT 4.0, Terminal Server Edition
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Datacenter Server

(Microsoft Corporation, 2003)

Now, if you remember the inventory details for this Windows host (on the previous page 13), it is running "Windows XP Version 2002 Service Pack 2". This means that the alerts generated from the scan do not apply and can be eliminated (or considered low priority) from the presentation of events to the end user. We will discuss on how to do this later. For now, we can keep in mind that seven alerts can be disregarded from the scan (relating to the Terminal Services vulnerability), and do not need to be presented as high severity. Really, the only observation we can take from this alert, is that a host is attempting to scan us (recon activity observation only). No action would be needed, as the host is not vulnerable to this and cannot be exploited

.

3.4.2. Linux Server Results

Let's take another look at a second alert, this time generated from the scanning host, directed at the Linux application server. We expect to be able to eliminate any events that do not apply to this host as well.

#0-(1-5225)	[bugtraq] [snort] TFTP NULL command attempt	2009-10-18 12:57:14
#1-(1-5221)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP request tcp	2009-10-18 12:54:00
#2-(1-5222)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP trap tcp	2009-10-18 12:54:00
#3-(1-5223)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP request tcp	2009-10-18 12:54:00
#4-(1-5224)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP trap tcp	2009-10-18 12:54:00
#5-(1-5220)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP trap tcp	2009-10-18 12:53:59
#6-(1-5219)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP request tcp	2009-10-18 12:53:59
#7-(1-5218)	[arachNIDS] [snort] ICMP webtrends scanner	2009-10-18 12:53:47
#8-(1-5217)	[arachNIDS] [snort] ICMP L3retriever Ping	2009-10-18 12:53:46
#9-(1-5216)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [bugtraq] [snort] SNMP private access udp	2009-10-18 12:53:43

Here, we can see that the Snort IDS engine had picked up several traffic patterns related to SNMP, specifically the ones for CVE 2002-0012:

"Vulnerabilities in a large number of SNMP implementations allow remote attackers to cause a denial of service or gain privileges via SNMPv1 trap handling, as demonstrated by the PROTOS c06-SNMPv1 test suite. NOTE: It is highly likely that this candidate will be SPLIT into multiple candidates, one or more for each vendor. This and other SNMP-related candidates will be updated when more accurate information is available."

(CVE Editorial Board, 2002)

But wait, per our profile in the inventory database, this host is not even running SNMP at all. In fact port 161 or 162 are not even open on the Linux host. Therefore, we can safely disregard the SNMP events altogether. Again, we will see how this correlation will work in the later part of this document.

4. Correlating Lab Results

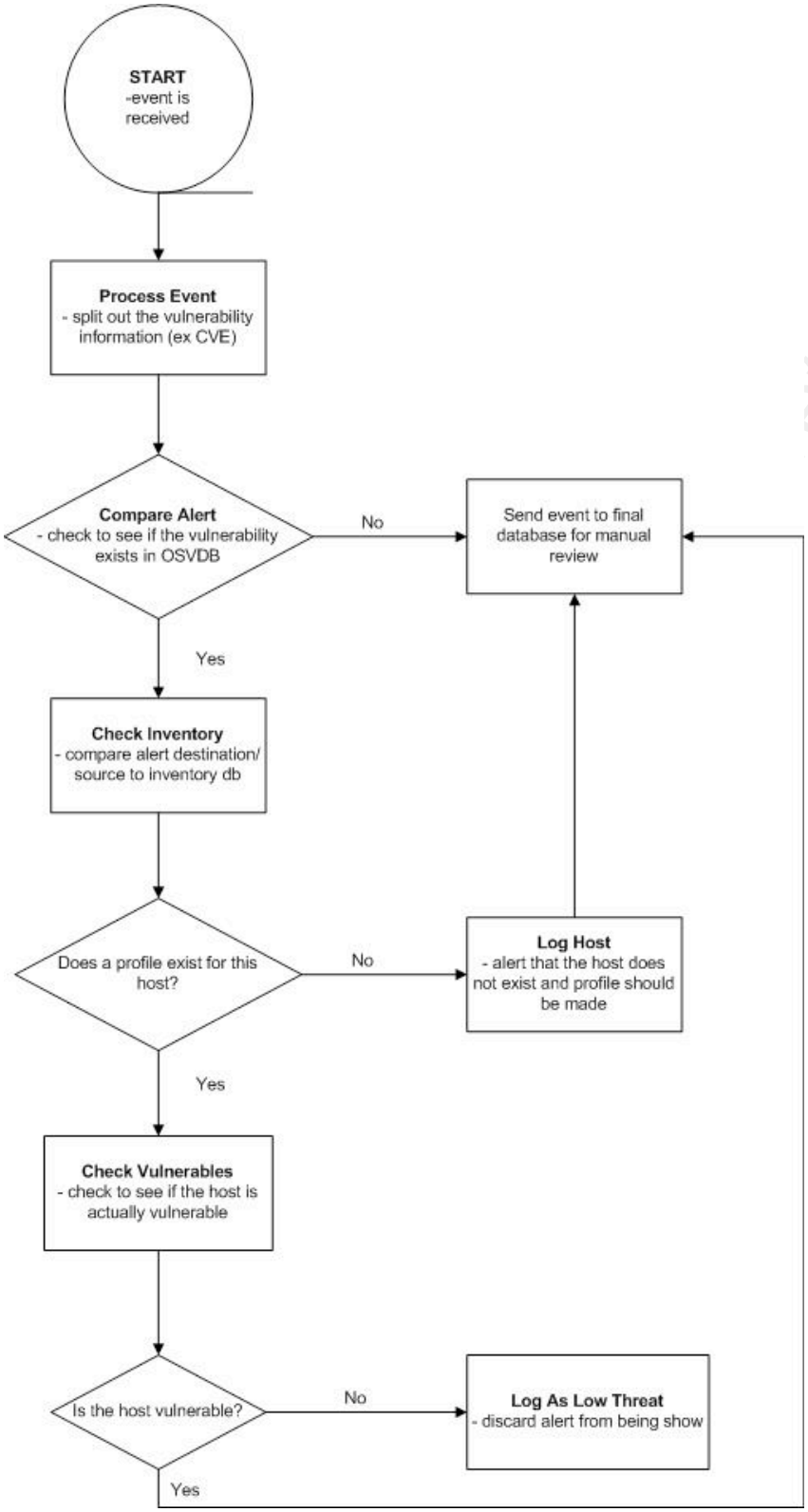
4.1. Lab Result Overview

Now that we have a couple solid events to base our research off of, we can start processing them and comparing to the two databases, showing how to efficiently eliminate the false positives. This will involve taking an alert, parsing through it to obtain the CVE information, looking up the CVE in our local OSVDB, then seeing if the vulnerability is applicable to the targeted host. From here, the results can be sent to a final database, where the pertinent alerts are presented to an analyst.

We also must keep in mind that, even though the alert may not be applicable to the host at hand, the alert needs to be treated as an "attempted" action against the network. While we do not foresee any immediate threat, the pure fact that someone is conducting malicious behavior should not be ignored. One way to factor in these low threat events, is to create some sort of criticality index, giving a higher rating to the number of occurrences some action took place between the source and destination.

Internal hosts that generate outbound IDS alerts (possible worm, Trojan, or virus infections) may also need to be given a higher priority in the criticality index, as these would indicate post-infection. It may be possible that the infected host contracted the malware by an undetected method, and the system does not alert based on the profile. Given the malware had changed system settings (ports, services, applications), the profile would no longer apply, as the system becomes unaware of the actual vulnerable system.

A basic algorithm is shown in the following diagram.



4.2. Applying the Correlation Algorithm

Now that we know the theory behind correlating events to the inventory and vulnerability databases, we will go into an outline of the program responsible for parsing the output and correlating the events. The following pseudo code will outline the process in the Perl programming language.

```
#!/usr/bin/perl

#keep reading the snort alert database forever until killed
While(1) {
        #create a connection to the snort alert database
        My $snort_db = DB->new('snort alert database');

        #extract the alert from the snort database
        My $alert = $snort_db->query('select alert from database');

        #parse out the cve information from the alert
        If($alert =~ m/'cve information '/) {
                #parse out the alert and CVE information
                My $cve =  cve_number;
                My $target = target;
                My $alert_id = snort_alert_id;
        }

        #open connection to the osvdb
        My $vuln_db = DB->new('vulnerability database');

        #grab all vulnerability information related to the cve above
        My @vuln_software = $vuln_db->query(' select affected_software from osvdb
                                                where cve = $cve');
```

```
        #open a connection to the inventory database
        my $inventory_db = DB->new('inventory database');

        #extract actual software installed on the host
        my @host_software =  $inventory_db->query('select software from inventory
                                                    where hostname/source = $target'
        #loop through the installed software on host x and compare it to the found
        #vulnerable software above and insert the alert into final db if found

        foreach my $host_app (@host_software) {
                foreach my $vuln_app (@vuln_software) {
                        if ($vuln_app eq $host_app) {
                                connect to final alert db and insert;
                        }
                }
        }
Sleep .5;
}
```

4.3. Result Display to End User

Now that the end database is populated with our relevant alerts, we can read from it and grab our pertinent security events. All of the irrelevant events have either been discarded, or prioritized as low threat. The end display will read from the final database, where the above Perl script writes to. These end alerts can be inserted in the same format as the original Snort alert, or altered to meet the needs of the end user. This is one of the great features of the correlation process, as it will work in almost every environment. The end security analyst will now only see the events of relevance.

5. Technology Drawbacks

5.1. Drawback Overview

While we are now eliminating many of the irrelevant alerts generated from vulnerability scans, we also have to be cognizant of several drawbacks to this correlation technology. Having an understanding of what these drawbacks are is essential in grasping the entire picture of our secured network. We also have to understand that this implementation may not suit the needs of every organization, nor may it work in every networked environment. The tools used in this paper may not necessarily be the ones used in your environment, but the general idea still holds true; creating an aftermarket automated correlation process for IDS events, and eliminating non-applicable positives through inventory correlation will greatly reduce the load on security analysts while increasing visibility into actual security threats.

5.2. Database Upkeep

As you may have figured out already, managing the three databases is essential in the successful use of this particular correlation technology. While the Snort alert database is pretty easy to maintain (since its already defined and updated accordingly using Snort), the other two (Inventory and OSVDB) need to be updated on a regular basis. Without a current inventory, there wouldn't be anything to compare the targets to.

5.2.1 Inventory Database Maintenance

Marinating a current inventory is beyond the scope of this document, but one might opt to create an automated process for this as well. This may include creating a policy, tracking all application installs and changes, and then inserting them into the inventory database. An inventory system could be developed and automated application mapping tools could keep the list up-to-date. We must also keep in mind that the goal of this particular correlation technology (as opposed to other passive monitoring tools), relies on a static inventory, which allows for the greatest correlation accuracy. While it does not matter how this list gets populated, it is essential that is be accurate.

5.2.2 Vulnerability Database Maintenance

The Open Source Vulnerability Database is a great, well maintained set of vulnerabilities that the common user is free to use. While the database is great for well-know vulnerabilities (provides CVE information, as well as affected products and applications), some of the lesser known ones either do not exist, or do not contain the appropriate information required to compare the event to our inventory database.

Maintaining the vulnerability database will involve alerting it to suit the needs of your network. For example, if the vulnerability is not referenced by a CVE number, one may want to add the number to the database, so that the automaton process can pick up on it and compare it to the generated IDS alert. A second example may be the fact that the applications are not listed for a particular CVE in the vulnerability database. This can be resolved by adding the pertinent applications to the list.

5.2. Ignored Event Visibility

As mentioned in previously, we may not want to ignore all the irrelevant events for a specified target. In doing so, we may not be visible to an attack. Instead of ignoring the irrelevant events, we can tag them as low threat, and wrap them up into a single event. For example, say the events from the SNMP scan against the Linux server (in the conducted lab) are not relevant due to the absence of SNMP on the server. Just because they are irrelevant does not mean someone was not trying to scan us. We may want to know that someone is "knocking at our door". The automation program can create a criticality index of the event and wrap all the SNMP events into a single "recon" event. This event will show up in the final alert database as low threat, and tell us that someone is trying to scan us (with irrelevant vulnerabilities).

6. Existing Correlation Technologies

Probably the most recognized relevant technology that attempts to profile systems in a passive matter is that of Sourcefire's RNA (Real-time Network Awareness).

"Sourcefire RNA is an innovative, passive sensing technology that provides real-time network intelligence to the Sourcefire 3D® System. RNA enables organizations to confidently protect their dynamic networks through a unique, patented combination of passive network discovery, network flow analysis, and targeted vulnerability assessment technologies."

(Sourcefire, 2009)

RNA attempts to profile systems in real-time by passively monitoring the addition of new hosts on the network and analyzing the traffic created by them. The RNA system proceeds to create a "baseline" for the normal traffic, and then compares this to all future traffic. It generates alerts based on anomalies from abnormal traffic.

The main difference between the two technologies, is the process by which the inventory databases get updated. Sourcefire RNA processes use automated passive mapping technologies to assign installed applications and services to a database. The RNA process also develops a baseline of expected traffic, and alerts on anything that does not match the baseline and matches an RNA rule in the Snort IDS engine configuration. The static entry method, as discussed within this paper, relies on manual insertion of the inventory records.

SC Magazine had done a review on the Sourcefire 3D system, a combination of the Snort IDS, and RNA processes. In their review, they had identified that the RNA setup is about 80% accurate in identifying the applications and services installed on a host. Other services are hard to identify without sufficient data and observations. The static entry mechanism, while harder to maintain on large network with many hosts, will provide better accuracy in the alerting mechanism.

(SC Magazine, 2005)

7. Conclusion

As we have seen through the correlation of IDS events generated from a vulnerability scan, compared to an inventory database, we can efficiently and effectively eliminate false positive from our system. This allows us focus on actual security events, and provides better incident response times, as we do not waste time analyzing false positives.

We have proved through the lab that there are many irrelevant signatures that fire on a vulnerability scan that do not need to be investigated. This can be done by applying a simple correlation algorithm, comparing the applications and services installed on a particular host to the vulnerability database to see if an event applies. The end result gives only events that are pertinent to the host.

We have discussed the drawbacks to the technology, but still have proven that, given the networked environment, the system is worth implementing. Even is an event does not apply to a particular host, it can still be logged and tracked, but given a lower priority. If the targeted host does not exist in the inventory database, the alert can be sent to the end user for manual analysis. All in all, correlation technology will save lots of time, research, and resources if properly maintained.

8. References

CVE Editorial Board, . (2001, July 25). Cve-2001-0540. Retrieved from http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0540

CVE Editorial Board, . (2002, January 1). Cve-2002-0012. Retrieved from http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012

Microsoft Corporation, . (2003, June 13). Microsoft security bulletin ms01-040. Retrieved from http://www.microsoft.com/technet/security/bulletin/ms01-040.mspx

Open Source Vulnerability Database, OSVDB. (2008). The Open source vulnerability database. Retrieved from http://osvdb.org/about

Timm, K. (2001, September 11). Strategies to reduce false positives and false negatives in nids. Retrieved from http://www.securityfocus.com/infocus/1463

SC Magazine, . (2005, June 15). Sourcefire 3d system. Sourcefire 3D System, Retrieved from http://www.scmagazineus.com/Sourcefire-3D-System/Review/213/

Sourcefire, . (2009, October). Network monitoring with real-time passive network intelligence. Retrieved from http://www.sourcefire.com/products/3D/rna