Interested in learning more about security?

SANS Institute InfoSec Reading Room

A Single IDS Console Please: ManHunt 2.1 Pilot Test

Many companies have deployed a variety of network intrusion detection systems (NIDS) over time as their networks and security strategies have evolved. We certainly found ourselves in this position at the company I work for. We had deployed Snort1, Dragon2 and ManTrap3 on the network, not to mention Tripwire4 and all of the host system log files we have to audit. This created a piecemeal system that left us with several administration consoles and hundreds of events to sort through. We needed a way to bring them togethe...

A Single IDS Console Please: ManHunt 2.1 Pilot Test
Scott Reynolds
GSEC v1.4

Abstract

Many companies have deployed a variety of network intrusion detection systems (NIDS) over time as their networks and security strategies have evolved. We certainly found ourselves in this position at the company I work for. We had deployed Snort[1], Dragon[2] and ManTrap[3] on the network, not to mention Tripwire[4] and all of the host system log files we have to audit. This created a piecemeal system that left us with several administration consoles and hundreds of events to sort through. We needed a way to bring them together into a single console that would enable our security personnel to aggregate, correlate and analyze them. Without that we will be crippled by the sheer volume of events. Furthermore, we wanted to add more sensors to our network, and preferably sensors that were based on a different technology than the signature based systems we had already deployed. ManHunt[5], a protocol anomaly based NIDS offered by Recourse Technologies[6], seemed like it would fill our requirements. After a demo from Recourse we decided to initiate a pilot of the product, which I had the opportunity to coordinate and implement. Recourse has in-depth descriptions of all of the product features as well as white papers available on their website so I won't try to do that here. The paper will focus on the features that were evaluated in the pilot against the high level functional requirements specified here:

- Ease of installation
- Ease of configuration
- Ease of upgrade
- Ability to integrate a variety of other vendors' sensors and network devices
- Uses a detection technology not currently deployed at our site
- Ease of use
- Usefulness as a "master console"
- Quality of support

To summarize the findings, ManHunt was found to be reasonably easy to install and configure. Near the end of the pilot Recourse released the next version of ManHunt, version 2.11. They generously agreed to extend the evaluation period so that the new features could be tested. The upgrade went smoothly. Custom settings, like filters and response policies, were preserved through the upgrade process. Unfortunately, I will be able to get the rest of the new features evaluated in time to get them in this paper. ManHunt supports the integration of sensors and network devices from enough vendors to meet our needs. ManHunt is based on a technology, protocol anomaly detection, which is not currently deployed at our site. The GUI of the administration console was intuitive. It took little time to get comfortable using the product. Much more time could, and will, be spent

becoming more familiar with advanced features, but that is outside the scope of this document. ManHunt is useful as a "master console" depending on the definition of "master console" and how ManHunt is configured. For the purpose of this pilot, "master console" means; "ManHunt provides the ability to view base events generated by all of the sensors deployed at our site, and if necessary, to use a sensors native console for more detailed analysis". If a stricter definition were used, ManHunt would not be as useful. For example, if the definition were; "The ability to see all details of any event generated by any sensor and the ability to perform forensic analysis using that information all within a single console". Then this would not be the right product. In all fairness, I don't know if that product even exists yet. The reason for this is the fact that ManHunt only has access to the information which other vendors make available. In the cases tested here, only event summary information could be passed to ManHunt. For deeper analysis it was still necessary to use the native console of the sensor that generated the event. The support provided by Recourse was both timely and useful. Issues were quickly and professionally resolved. Overall, ManHunt met the high level functional requirements outlined for this pilot.

ManHunt Overview

Recourse describes ManHunt as a;

> "…threat management system [that] protects your information infrastructure with enterprise-wide high-speed gigabit detection, real-time threat analysis, and policy-based responses…Through the use of distributed sensors, and high-speed statistical correlation and analysis, ManHunt gathers intelligence from across the enterprise to quickly identify and respond to both known and unknown (or zero-day) attacks."

Suffice it to say, ManHunt is a robust and richly featured system that can accommodate integration of products from a variety of other vendors. The pilot was run with three ManHunt sensors, a ManTrap system, two Dragon sensors, and a Snort sensor. The two Dragon sensors actually send alerts to a Dragon Policy Manager (DPM) server which then feeds SNMP traps to ManHunt. ManHunt's native support for the Snort rule language was also tested by defining some custom events. The product supports several other sensors including ISS RealSecure, Cisco, Checkpoint, NetScreen, and the commercial version of Tripwire.
ManHunt is, at its core, a protocol anomaly detection system, but it does support the Snort rules language for the creation of custom rules. One advantage of adding a protocol anomaly detector to our existing signature based systems is its ability to detect what Recourse terms "Day Zero" attacks. If an attack violates one of the many protocols that ManHunt supports, an event will be triggered even though no signature has been developed. There is a white paper[7] available from Recourse which describes how ManHunt detected Nimda outbreaks before the new worm had been identified.

When it is time to respond, ManHunt supports a variety of configurable responses based on user defined policies. These are described later in this paper.

ManHunt also incorporates a fragmentation reassembly engine to protect against attacks like Fragrouter[8]. An attacker will use a tool like Fragrouter in an attempt to avoid detection by breaking the packets up into small fragments which are reassembled by the target. If an IDS, or firewall for that matter, does not incorporate fragmentation reassembly it will be blind to any attack launched using this type of evasion. ManHunt can also help to detect “low and slow” attacks with a user configurable threshold. Low and slow attacks occur when an attacker uses a long interval between packets in an attempt to avoid detection. When tuning IDSs to eliminate noise it is possible to miss an attack or probe if the interval between packets sent by the attacker is larger than the noise threshold. In ManHunt, by tuning the length of time an incident is considered active, an administrator can increase chance of catching multiple events in a single incident.

Licensing for ManHunt is sold by bandwidth. A "small" license provides 100 Mbps of traffic that can be monitored with ManHunt sensors. The small license starts around $25,000. ManHunt supports up to ten modular 10/100 Ethernet interfaces or two Gigabit interfaces per server. This means a single ManHunt node could monitor two Gigabit Ethernet pipes simultaneously. An independent lab test by Miercom[9] found that ManHunt yielded 100% detection on a Gigabit network at 60%, 80% and 95% utilization in three different traffic scenarios. The 100 Mbps on the "Small" license may not sound like much, but the three sensors deployed in this pilot didn't come close to using it all. Even though the capacity of the networks monitored was actually much higher than this. Also, the third party sensors don’t count against bandwidth on the license since they are processing traffic and then just sending the events to ManHunt.

ManHunt uses a Java powered cryptographic iButton from Dallas Semiconductor Technology[10] to validate the license and to verify the integrity of the database. The iButton is a hardware device that is preprogrammed with the amount of bandwidth purchased for the ManHunt node it is attached to. It will cause the product to clip if that amount of traffic is exceeded in aggregate across ManHunt sensors. In our case, the duration of the pilot was also encoded. ManHunt will cease to function the next time the server is rebooted after the end date programmed in the iButton. The iButton also generates a FIPS 140[11] compliant digital signature which is used to periodically sign the database to ensure its integrity. FIPS are Federal Information Processing Standards set by the National Institute of Standards and Technology (NIST). FIPS 140 is titled "Security Requirements for Cryptographic Modules" and outlines requirements for Security Level 1 through Security Level 4 systems, with Level 4 being the most secure. This has become a litmus test for determining the admissibility of electronically captured data as evidence.

Installation

Installation was fairly easy. For this pilot a Sun Ultra 10 was used with a quad Ethernet card and running Solaris 8 with current patches. The box has a single processor and 512 MB memory. ManHunt will also run on Intel platforms running the Solaris 8 Intel Edition. The installation of ManHunt further hardens the system by disabling UUCP, LDAP, service location, cachefs, SNMP, printing, sendmail, and the answer book manager in runlevels 2 and 3. Runlevel 2 is multi-user mode. Runlevel 3 extends runlevel 2 to make services available on the network. The use of another hardening tool such as Titan 4.0 for Solaris 8 could add further value but would require compatibility testing with the product[12].
Installing ManHunt simply involved running a script and modifying a couple of configuration files. One interface with a static IP address on the internal network for the administration console to connect to is required. The built-in NIC hme0 was used for this. An interface for each ManHunt sensor is also required. The quad Ethernet was brought up as qfe0, qfe1, qfe2 and qfe3 with IP's of 0.0.0.0 and was used for the sensors. With no IP address they can safely be connected to any location inside or outside the network. One was placed outside the firewall interfaced to our ISP, one in the firewall DMZ and the third roamed on mirrored ports on a Cisco Catalyst 6500 which handles several virtual LANs (VLANs). More detail about the roaming sensor in a bit.
The rest of the configuration was done through the GUI administration console which runs on Windows NT, 2000 or XP desktops. The only requirement for the console is that Java Runtime Environment (JRE) 1.3 be installed. The necessary files are available on the install disk if they are not already loaded.
The next step was to configure the main ManHunt node. This was all done in the administration console and simply involved creating and configuring the interfaces. Basically this involves creating an object and linking it to each of the interfaces in the box.

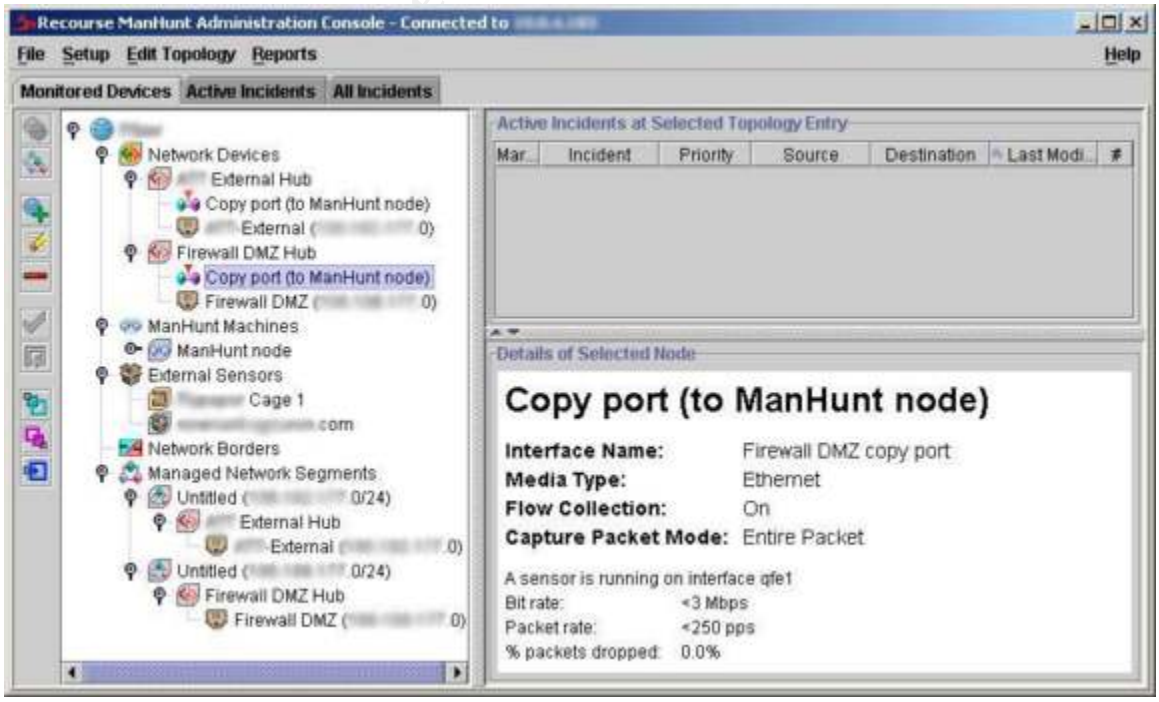

Figure 1: ManHunt admin console showing device hierarchy

Figure 1 shows the hierarchy of managed objects. Notice the various sensors (not all had been set up yet in this graphic). This snapshot was taken at night when traffic was light so the display reads <3 Mbps and <250 pps. When traffic exceeds these values the actual value will be displayed.

Upgrading

Near what would have been the end of the pilot, Recourse released the next version of ManHunt, version 2.11. They generously agreed to extend the evaluation period so that the new version could be tested. The upgrade went smoothly and took very little time. Custom settings like filters and policies were preserved through the upgrade process. ManHunt did need to be restarted to pick up the changes which created a small window of no detection.

Adding Sensors

There are two steps to add external sensors. First, install the ManHunt Smart Agent (MSA) for the type of sensor. This is a translator that takes the traps received from a Dragon sensor, for example, and converts them into ManHunt format. The MSA is only installed once for each type of external sensor. Second, add the sensor by clicking the type and defining the interface. Once it is added to ManHunt, the sensor must be configured to send traps to the ManHunt internal interface. I decided not to send every event from the Dragon sensors into ManHunt. It seemed like doing so would defeat the purpose of reducing information overload. Also, Dragon has a robust interface for doing detailed analysis of events. However, I wanted to test ManHunt as a centralized console. I settled on the "Trojans" and "Successful Compromises" signature lists to trigger alerts to be sent to ManHunt. This seemed like a good way to take advantage of the known signatures because I would definitely want to know right away if someone had a backdoor installed on a host on the network or had successfully compromised a host. I also added the TCP:Portscan event because it is easy to trigger for testing the alert. These alerts were fed through SNMP traps from the Dragon Policy Manager (DPM) server to ManHunt. The most difficult part of setting this up was figuring out what format the ManHunt MSA was expecting to receive so that the Dragon alert could be properly formed. This was not documented and required a call to Recourse technical support. It turns out that Recourse tested with an earlier version of the Dragon Alarmtool and the format has changed slightly in the version 5.2 that we are running. After analyzing some packet captures of the SNMP traps that the Alarmtool was sending, Recourse was able to specify the proper format for the alert. Once the Dragon alert format was set in the proper order, events began to show up in the ManHunt console as expected. Well, almost. At the time of this writing I am still working on a problem getting the source and destination ports into the alert in the console. Running

snoop on the ManHunt host shows that the packets arriving contain all of the information, so I suspect that there is still a problem with the format of the alert. Routers and switches can also be defined and then have ManHunt sensors attached to them. ManHunt supports routers from Cisco and Juniper and switches by Cisco, Foundry and Lucent (now Avaya). The interface for adding these is as intuitive as the others.
Now, some detail on the roaming sensor. Let's say you are using up all of the bandwidth included in your license and can't buy more, but you have not covered all of the networks you wanted to. One option is to set up what Recourse calls a "roaming sensor". Maybe you have a Cisco switch with four VLANs you want to monitor. If you aren't concerned with monitoring them 100% of the time, a mirrored port can be configured for each VLAN and ManHunt can automatically switch between them at some interval. Maybe ten minutes on each VLAN would be enough. At least you would have some coverage until more bandwidth can be purchased on the license.

Reviewing Events in the Console

ManHunt organizes data by incidents which are made up of events. For example, if an attacker probes port 21 on an ftp server a new incident is created containing a portsweep event. If that same attacker, five minutes later tries an anonymous login because he found the port open, the event will be added to the same incident, if it is still active. The user can define how long an incident remains active. As long as an incident is active ManHunt will continue to add events to it. After an incident is closed, a new event will cause a new incident to be created. This type of automatic correlation of events is one of the things that make ManHunt such a useful tool.
Another example is correlation across sensors. If an attacker triggers an event on two different sensors both events will be added to the same incident. A portsweep of one server is detected by a perimeter ManHunt sensor. Since the attacker appears to be sweeping an entire subnet for open SSH ports he hits the ManTrap system a few seconds later. This is picked up by ManTrap as an inbound connection attempt.

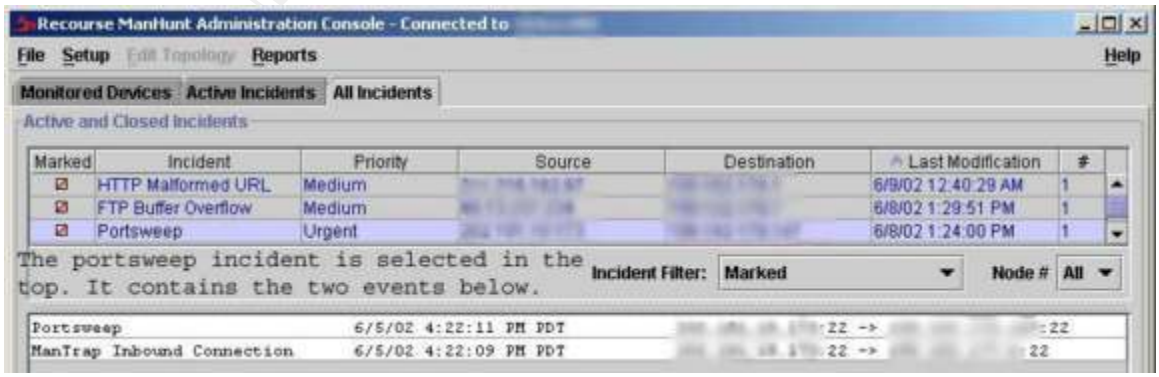

Figure 2: An attacker sweeps subnets for open SSH ports. Two sensors detect the probe and the events are correlated as a single incident.

Normally, the two sensors would report the event to their own management consoles. However, since ManTrap was added to the ManHunt console, the two events come in and are automatically correlated as one incident.
One area that could be handled better is the depth of information available in the ManHunt console for events generated by other vendors. For example, on the event displayed above it is possible to drill down to more detailed information. However, on any event generated by one of the Dragon sensors, the event description just says to go look at the Enterasys console and the raw packet is not included. This is not directly the fault of Recourse in that the description of the event, and the packet data, both exist in the Enterasys console; there is just no way to pass them to ManHunt with rest of the event. As Marcus Ranum and Robert Gleichauf point out in a recent issue of Information Security[13], the lack of data standards for IDSs is preventing better sharing of information between vendors and is hampering the development of more sophisticated analysis tools that could operate more efficiently if there was a standard data model for IDS output. When such a standard is developed we will see products that provide context sensitive information by tying the results of vulnerability scans to events. So if I have no servers running Microsoft's IIS, I won't see events related to the exploit of an IIS server; at least not as a high priority event. This will further reduce the burden of information overload placed on analysts today.

Protocol Anomaly Detection

For a brief discussion on the various types of IDSs, I direct the reader to "An Introduction to Intrusion Detection Systems" at SecurityFocus Online[14]. Protocol anomaly detection is based on the idea that many attacks violate the protocol which they are using. Therefore, if an anomaly can be detected, it should be possible to detect an attack without matching it to a known signature. One flaw in this concept that I have observed in this pilot is that some legitimate traffic violate protocols and therefore generate false alarms. Signature based detection, on the other hand, matches a known attack to a specific signature. In my experience it is desirable to have a system which incorporates both types of detection. This is because it can take more time to analyze what is going on in an anomaly event whereas a signature will just tell you what signature triggered the event. So, it can be faster to diagnose known attacks with a signature engine, but they will miss new attacks and possibly variants of existing attacks. That is where anomaly detection is very powerful. It has the ability to detect an attack before a signature has been developed.
While writing this paper, an event was triggered which is a perfect example of protocol anomaly detection in action. Figure 3 is the event detail, which brings up another feature. Each event is viewable in the GUI, but you can also click “To Clipboard” to easily paste the event into an email or report. Notice that the event type is HTTP Malformed URL and the base type is CRS/HTTP_BAD_REQUEST.

ManHunt detected this event because the packet violated the HTTP RFC. Anyone who was involved with intrusion analysis in 2001 will quickly recognize this as a system infected with the Code Red[15] worm trying to exploit and infect the target. The signature based sensors today would correctly identify this event as Code Red because of the signature of the payload. However, ManHunt's protocol anomaly engine would have recognized it as a possible attack on "Day Zero", before the worm was identified and signatures were developed and updated. Now that sounds great, but how useful is it to most people using this type of system? Would the appearance of a protocol anomaly have been enough to determine that an attack had occurred and to analyze the attack in depth? That depends on a variety of things. For one, the configuration of response policies for the event could play a part. Also, the frequency of the event could play a part in the ability to do analysis. Let's say that an event only occurs once. If there was not a policy in place for that event type to record more traffic, an analyst may not have enough information to determine the method of attack and whether or not it were successful. This may be the case in a buffer overflow attempt for example. A buffer overflow attack may involve just one specially crafted packet designed to exploit a specific vulnerability in a program that accepts connections on a host. It does this by sending more data than the program is prepared to accept. If the program does not check the size of the passed data before writing it to a buffer in memory, the passed data can be written beyond the end of the buffer. This causes executable instructions to be overwritten in memory. If the attacker crafts the attack properly he can insert his own instructions to be executed. For a detailed analysis of buffer overflows read Aleph One's "Smashing the Stack for Fun and Profit"[16]. If the attack were successful, it would likely be followed up by some type of connection. That could possibly be followed by some traffic associated with copying down a rootkit[17] to the compromised system. Rootkits are tools used by attackers to automatically cover their tracks, install back doors and even to patch the original vulnerability on the target system so that other attackers cannot gain access the same way. This will allow the attacker to "own" the system undetected for a longer period of time. If that traffic were not recorded by a triggered policy, the analyst would have a much more challenging time determining what had happened. So, it might be wise to set up a Record Traffic policy to be triggered on events that indicate a possible buffer overflow attempt. This will be shown in the next section "Creating Policies".

On the other hand, if an event is occurring repeatedly for an extended period of time, an analyst will have more information available to build a response in real time. This might be the case with the outbreak of a new worm that is traversing the network. For example, the Code Red worm would generate a lot of port scan events that could be tracked back to a particular system. The system could then be isolated for analysis. Based on the targeting of port 80 it could be assumed that this attack is aimed at a web server. If the infected machine were running an IIS server, and not an Apache server, it would be a safe bet that the attack is targeting IIS servers. Based on this information a response to quickly identify all IIS servers on the network could be developed and the levels of patching verified.

Event Type: HTTP Malformed URL
Base Event Type: RCRS/HTTP_BAD_REQUEST
Event ID: 3d0304ce6f684971:1
Start Time: 6/9/02 12:33:34 AM
End Time: 6/9/02 12:33:34 AM
Device: External Hub
Interface: -External-copy port
Source IPs: xxx.xxx.xxx.xxx:1795.
Destination IPs: xxx.xxx.xxx.xxx:80.
CVE Reference Number: (none)
Priority: Medium
Severity: 128
Reliability: 128
Aggregate Count: 1
Protocol: TCP
TCP Header Flags: PUSH, ACK
TCP Header Length: 20
IP Version: 4
IP Header Length: 20
Type of Service: ROUTINE
IP Total Length: 552
Time To Live: 113
IP Flags: DF
Source: xxx.xxx.xxx.xxx:1795
Destination: xxx.xxx.xxx.xxx:80

Packet:
=======
Byte Format: 45 00 02 28 13 67 40 00 71 06 05 95 D3 D8 A2 43 C6
B6 B2 01 07 03 00 50 91 98 0B 20 78 DB 87 C8 50
18 40 00 4B 0B 00 00 2F 64 65 66 61 75 6C 74
2E 69 64 61 3F 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
(…A bunch of bytes omitted for brevity…)
BD E8 FD FF FF B9 86 00 00 00 B8 CC CC CC CC
F3 AB C7 85 70 FE FF FF 00 00 00 00 E9 0A 0B 00 00
8F
ASCII Format: E..(.g@.q.....C.....P... x...P.@.K.../def
ault.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%
u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9
090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53f
f%u0078%u0000%u00=a HTTP/1.0..Content-type:
text/xml.HOST:www.worm.com. Accept: */*.Con
tent-length: 3569 ....U.....SVW.....
.....p.....
Alert Buffer:
=============
Byte Format: 3A 20 2A 2F 2A 0A 43 6F 6E 74 65 6E 74 2D 6C
65 6E 67 74 68 3A 20 33 35 36 39 20 0D 0A 0D
0A 55 8B EC 81 EC 18 02 00 00 53 56 57 8D BD

E8 FD
ASCII Format: : */*.Content-length: 3569 ....U.....SVW.
...

Figure 3: Sanitized event detail of Code Red attack detected by ManHunt as a violation of the HTTP RFC.

Creating Policies

Policies are also created through the administration console. Clicking in each field brings up a GUI that presents all the possible choices for that field. The first policy shown in figure 4 will send an email alert to a distribution list if someone starts a root process in the ManTrap cage. A process starting in the cage is a sure sign that someone has gained access to the system and we want to know about it immediately so that we can watch in real time. As mentioned in the previous section, this is where we could create a policy to record traffic following an event that indicates a possible buffer overflow attack. This is shown in the second policy in figure 4. If traffic is recorded it can then be played back as a session through the console for analysis.

Figure 4: Policy creation GUI.

The event type field can be one, some, or all of the hundreds of possible events. The priority field can be set to filter on less than, greater than or equal to one of the seven priority ratings which are; informational, low, medium, high, urgent, critical and fatal error. The last is a special type used only in the event of a failure of the ManTrap system and the rest are associated with event types.
The possible responses are; none, Email Alert, SNMP Alert, Allow Handoff, Trackback, Custom Response, Terminate, Traffic Record, Console Response and Suggest QoS ACL. Each is appropriately configurable. The Allow Handoff option allows secure handoff of an event between two ManHunt nodes. This could be useful if your ISP had a ManHunt node and you wanted to have certain events automatically passed off to them for investigation. Trackback traces an attack back to its source or entry point. The Traffic Record response can be configured to record traffic for a period of time or until a set number of packets have been collected. This traffic can then be viewed using snoop with the capture file as input. Terminate will cause ManHunt to send a TCP RST to the source and destination from its administrative interface. This response would not work against attacks like a SYN flood where connections are not required for the DoS to be effective.

False Positives

One problem inherent in any IDS is false positives. Indeed, when first brought online the number of events triggered by the ManHunt sensors was fairly high. Local filters can be used to filter out some of the noise. For example, the most frequent two event types initially triggered on our network were HTTP_BAD_MSGHDR_TEXT and HTTP_BAD_REQUEST. It didn't take too much digging to find out that they were caused by Instant Messenger clients from Yahoo, MSN and AOL that were violating the HTTP RFC. The way to solve the problem, (if you want to allow instant messaging in the first place, but that's a different issue) is to create a filter that ignores the event to or from specific servers. It is easy to do and looks like the following snippet. The IP addresses would be inserted and the * wild card may be used for IP or port. A single source/destination pair goes on each line after !matchiplist and a single event type on each line after !matchtype.

```
!begin_rule
!drop
!matchtype
HTTP_BAD_MSGHDR_TEXT
HTTP_BAD_REQUEST
!matchiplist
[insert source/cidr:port] , [insert dest/cidr:port]
[insert source/cidr:port] , [insert dest/cidr:port]
!end_rule
```

Recourse recommends performing a quarterly sanity check on filters by removing them and adding them back as needed to make sure you don't miss important events. Also, a value is tracked in the event database that will let you know how many events are dropped by each filter.

Reporting

ManHunt includes several reports and can also be configured to send a summary by email every night. The reports included are fairly limited and it is obvious that this version is intended to be more of a real time console than a reporting tool. Also, the queries seemed a little slow to run. This is probably due to the hardware used for the pilot. Since the pilot system is hosting sensors, as well as running analysis engines and generating reports, it should have at least one more processor and probably twice as much memory. It could also have to do with tuning the maximum size that the log files are allowed to reach before they are rotated. I observed that when the reporting performance gets slow, rotating the logs will return the system to normal. So the maximum log size should be reduced or more RAM added to the system.

Version 2.11 includes exporting to HTML format and support for real time exporting of events to an Oracle Database. Unfortunately, the database piece cannot be tested at this time due to the fact that I don't have an extra Oracle license available. These are the reports that can be run from the console:

- Top Event Types
- Top Event Destinations
- Top Event Sources
- Incidents Per Month
- Incidents Per Day
- Incidents Per Hour
- Incident List
- Events Per Month
- Events Per Day
- Events Per Hour
- Events by Classfull Source
- Events by Classfull Destination
- Events by Protocol
- Events by Vendor
- Event Types of Source/Dest
- Destinations of Source
- Sources of Destination
- ManHunt Login
- Devices with Flow Statistics

Let's take a closer look at one of the reports to get a feel for how they work. If an attacker is probing and attacking a network over a period of time it is likely that several incidents will be created containing many events originating from a single source. A likely thing that a security administrator would want to do is see every event that originated from that source. In ManHunt this is the "Destinations of Source" report. When selected this report allows a source IP address to be specified along with a date range. The query returns every destination IP address that the specified source has targeted and how many events have been triggered against each target. This can be displayed in table format as shown in Figure 5 below or can be switched to a bar chart.

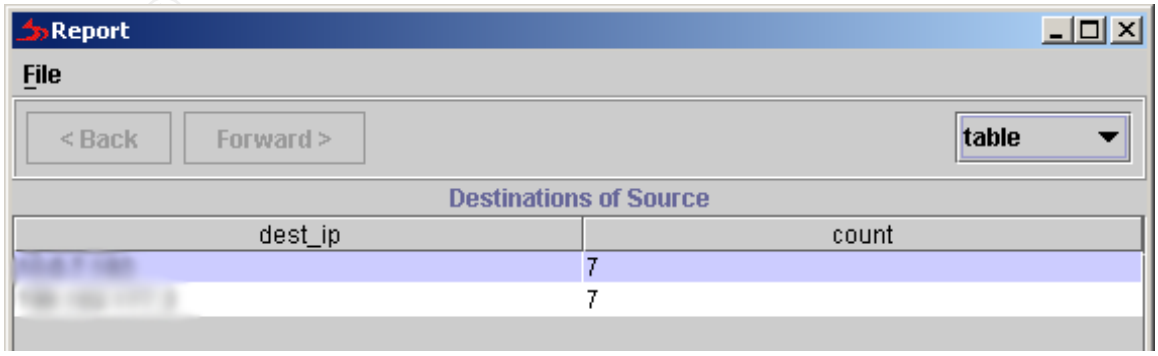

Figure 5: Sanitized Report Output for Destinations of Source report

Clicking on the count for one of the targets drills down further, displaying more detailed information about each event. Again, this can be displayed in table or bar chart format. The time of the event, the source and destination IPs, the type of event and the name of the sensor that detected the event are displayed as shown in Figure 6 below.

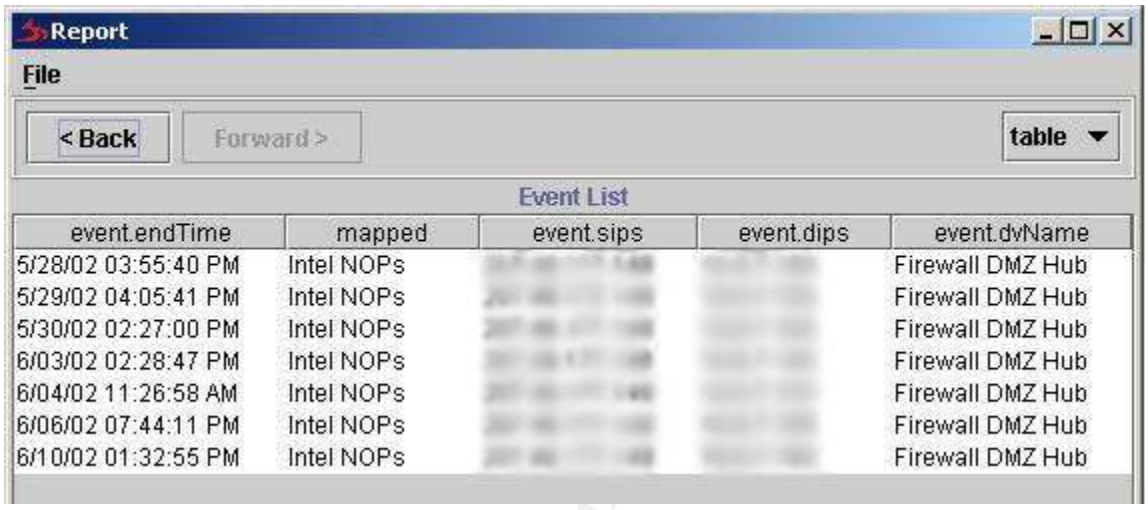

Figure 6: Report Output for Destinations of Source report

As you can see, this report will quickly give an administrator a good picture of the activity of an attacker over time. Most of the reports allow the analyst to drill down for more detailed information.

Product Support

The support from Recourse during this pilot has been outstanding. Both the field engineer and the technical support staff answering the phones were knowledgeable about the product and professional in their handling of issues. Calls to the support line were tracked and followed up until resolved. I even contacted the regular technical support line once, rather than the field engineer. I wanted to see if the post purchase support would be as good as the field support. I'm happy to say that they were very responsive and returned calls and emails quickly, and answered questions accurately.

Conclusion

This pilot was performed for the purpose of evaluating ManHunt as a strategic tool in our security architecture. The high level functional requirements were as follows:

- Ease of installation

- Ease of configuration
- Ease of upgrade
- Ability to integrate a variety of other vendors' sensors and network devices
- Uses a detection technology not currently deployed at our site
- Ease of use
- Usefulness as a "master console"
- Quality of support

Given that there is never a silver bullet solution for security it seems that ManHunt will work for the purpose which we intend to use it. It could be a strategic addition to our intrusion detection architecture. The product was reasonably easy to install, configure and upgrade. ManHunt supports the integration of sensors and network devices from enough vendors to meet our needs. ManHunt will provide a single console for security personnel to view events across sensors, although, this could be improved by better data sharing between vendors. In any case, the other vendors' consoles will still be available for follow on investigation as needed. ManHunt provides a detection technology that is different than those currently deployed through its protocol anomaly detection. The system integrated well into our existing environment and required very little time to get comfortable using. Support from Recourse was outstanding. In closing, I would just say that when evaluating an IDS it is very important to know what functional requirements must be met. A product that is right for one organization may be completely wrong for another. I highly recommend figuring out what is needed and coming up with a short list of candidates that may meet those needs. Then, if possible take them for a test drive. This will allow you to get past the hype and decide if the product really meets your requirements.

---

[1] www.snort.org – An open source signature based network intrusion detection system.

[2] http://www.enterasys.com/ids/ - A commercial signature based network intrusion detection system.

[3] http://www.recourse.com/product/ManTrap/ - A commercial deception based intrusion detection system also known as a honeypot.

[4] http://www.tripwire.org – An open source host based intrusion detection. Linux version detects changes to files and notifies user.

[5] http://www.recourse.com/product/ManHunt/ - A commercial protocol anomaly based network intrusion detection system.

[6] http://www.recourse.com

[7] http://www.recourse.com/product/white.php

[8] http://www.monkey.org/~dugsong/ - Fragrouter was released by Dug Song in 1999 to demonstrate the common vulnerability he discovered in many firewalls. By fragmenting packets it is possible to “see” beyond many firewalls. Fragrouter is an implementation of the exploit for this vulnerability. Dug has since censored his web site citing the Digital Millennium Copyright Act (DMCA). Fragrouter is still available from other sites like http://www.w00w00.org/files/sectools/

[9] Miercom. "Report 061101". Miercom. Novenmber 2001. http://www.mier.com/ (30 May, 2002) Note: Miercom independent lab test of ManHunt. Available for free download after free registration.

[10] http://www.ibutton.com/index.html - Hardware device for validating license and cryptographically signing database.

[11] Information Technology Laboratory, National Institute of Standards and Technology. "Federal Information Processing Standards 140-2, Security Requirements for Cryptographic Modules". May 25, 2001. http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf (15 June, 2002)

[12] Powell, Brad. "Titan 4.0 Beta 2". Powell, Brad. 31 August, 2001. http://www.fish.com/titan/TITAN_documentation.html (16 June 2002)

[13] Lindstrom, Pete. "IDS at the Crossroads". Information Security, June 2002. http://www.infosecuritymag.com/2002/jun/cover.shtml

[14] Innella, Paul and McMillan, Oba. "An Introduction to Intrusion Detection Systems". SecurityFocus Online. December 6, 2001. http://online.securityfocus.com/infocus/1520 (16 June, 2002)

[15] Permeh, Ryan. Maiffret, Marc. "AL20010717 .ida "Code Red" worm", eEye Digital Security. July 17, 2001. http://www.eeye.com/html/Research/Advisories/AL20010717.html - (2 June, 2002)

[16] One, Aleph. "Smashing the Stack for Fun and Profit". Phrack 49, Volume Seven, Issue Forty-Nine. http://www.cs.ucsb.edu/~jzhou/security/overflow.html (15 June, 2002)

[17] Altunergil, Oktay. "Understanding Rootkits". O'Reilly Network, Dec. 14, 2001. http://linux.oreillynet.com/pub/a/linux/2001/12/14/rootkit.html (15 June, 2002)

[18] Bace, Rebecca Gurley."Intrusion Detection", Macmillan Technical Publishing 2000. Referred to Ch. 10 for information on evaluating intrusion detection systems.

[19] Recourse Tecnologies. "ManHunt v.2.0 Administration Guide". Recourse Technologies, 2002.

[20] Enterasys Networks. "Alarm Tool Guide" Enterasys Networks, 2002.

Last Updated: August 21st, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS San Francisco Fall 2017	OnlineCAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced