Beyond the Anomaly: The Quest for the Underlying Cause

Version 1.0

Todd Heberlein
Net Squared, Inc.

23 Mar 2005

In the paper "Why Anomaly Detection Sucks" we looked at an alert to an unexpected event (an anomaly) on one of our systems. To our frustration, we could not identify the underlying cause of the event, and we were left with many lost hours and a very uncomfortable feeling that something might be wrong but we could do nothing about it. This report follows our continuing quest to identify the underlying cause of the alert. Having exhausted the leads using the original alert, we set up a system to allow the apparent attack to continue. This activity led us back search for new evidence on the original host – some dating back a year and a half. Eventually we traced down the cause of the alert, and it was not from any attack.

1 Introduction

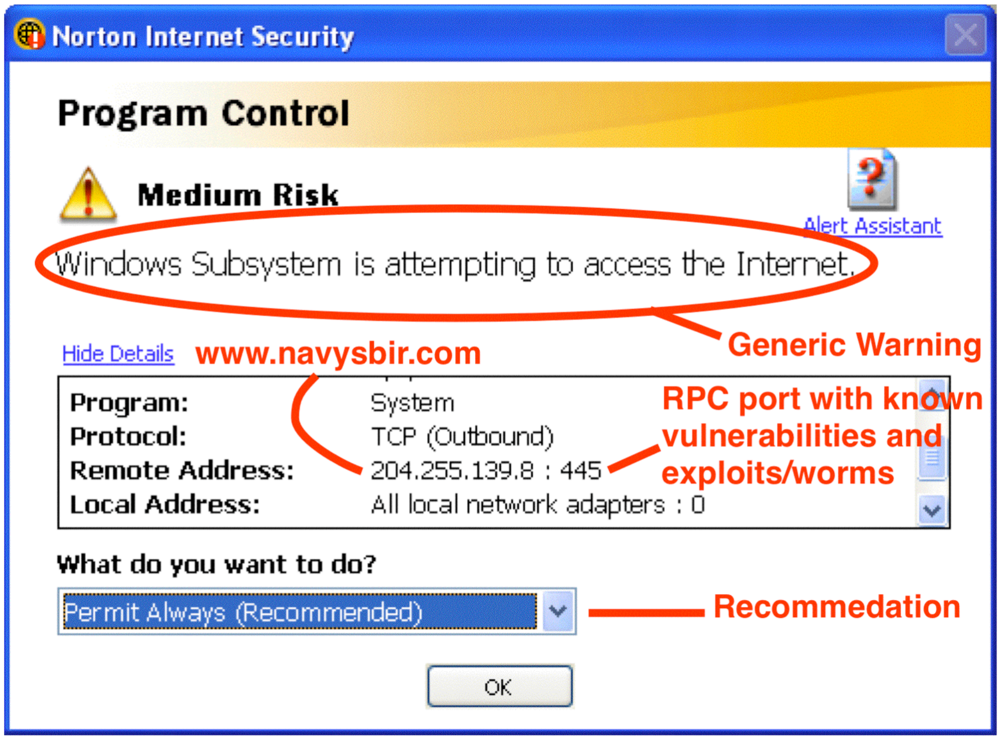
The paper "Why Anomaly Detection Sucks" highlighted our frustrations with an unexpected network event (see Figure 1). The problem is that such an alert leads to more questions. What is the underlying cause of the alert? Is it something I should be concerned about? If it is something I should be concerned about, what should I do about it? Attempting to answer these questions can be a very time consuming and potentially fruitless exercise. In the end you may only be left with an uneasy feeling that something is wrong but you don't know what. In such cases, never having been notified about the activity may have been a better solution – ignorance is bliss.

However, being supposed computer security experts we felt that we had to continue investigating the problem until we knew one way or another what the problem was and what to do about it. This paper covers our experiments and investigations that spanned several weeks, and we are now confident that we know the underlying causes behind the activity that caused the alert.

Section 2, The Fear, describes why we were concerned about this event and why we devoted so much time investigating it. Section 3, Preliminary Forensics and Environment Concerns, describes our initial forensics and our fear about carrying out an investigation in a changing environment. Section 4, Router, describes the changes we made to our network in order to gather additional details about the event. Section 5, Honeypot, describes the honeypot we deployed to gather more data about the event. Section 6, looks at the data we collected using our modified router and honeypot. Section 7, Host Evidence, returns to the offending host to continue the investigation. Section 8, Repeatability of Events, describes how we could on demand create the activity that initially led to the alert. Section 9, Probable Scenario, describes our strongest theory for the source of the alert. Section 10, Lessons Learned, enumerates the lessons we learned through this investigation. Finally, Section 11, Conclusions, summarizes this paper.

2 The Fear

Figure 1 shows the original alert from Norton Internet Security software that triggered our investigation. Actually, the original message did not show the details in box in the middle of the window; we needed to click on Show Details to see this additional information. Without the details box, we only know that "Windows Subsystem is attempting to access the Internet" and Norton recommends we always let this activity occur. The details, however, shows that the host is trying to connect to a remote host via port 445.


Norton Internet Security

Program Control

Medium Risk

Windows Subsystem is attempting to access the Internet.

Program:	System
Protocol:	TCP (Outbound)
Remote Address:	204.255.139.8 : 445
Local Address:	All local network adapters : 0

What do you want to do?

Permit Always (Recommended)

Figure 1: The Alert That Triggered The Investigation

The reference to port 445 raised the biggest concern because there are a number of security issues associated with this port. There are at least 11 vulnerabilities associated with this port (CVEs 2003-0352, 2003-0528, 2003-0533, 2003-0605, 2003-0715, 2003-0812, 2003-0813, 2003-0818, 2004-0206, 2004-1154, and 2005-0045), 116 Snort rules to detect activity on port 445, and numerous worms and worm variants (e.g., Sasser, Deloder, Bolgi, Spybot, Slinbot, and Lioten) that travel over port 445. If our system was trying to perform an outbound connection to this highly vulnerable port to a remote system, we feared that our system had been compromised and was attempting to spread the attack.

3 Preliminary Forensics and Environment Concerns

We performed preliminary forensics, including running security scans of the disk, making sure all patches were applied, and checked telltale signs of known malicious programs. Later we tried to correlate times of the attempted connection with the user logged in (one of the normal users or administrator); it turns out that Microsoft does not provide this simple capability or has hidden it very well. We also checked all the programs started automatically (using msconfig) by comparing their names to known programs (www.sysinfo.org/startuplist.php). These efforts turned up no useful information.

As we continued our search for the cause of the activity we were concerned that the activity might eventually stop and deny us the ability to track it down. During our investigations Microsoft released at least a dozen vulnerability patches, and even Symantec, whose security products raised the alert, issued security patches for their products. Today's networks are continuously changing with a continuous stream of patches and signature updates for a wide range of system software and applications. Ultimately this makes it very difficult to conduct controlled experiments in order to track down the source of the problem.

Indeed, when we inserted a new router in the network (see Section 4), Symantec's software noticed that the router's MAC address had changed (even though the IP address remained the same), and automatically applied a new set of security rules. The new rules stopped alerting us to the outbound requests to port 445 and simply allowed the traffic to continue. In this particular case we had deployed additional sensors to detect and capture the activity, but in other circumstances this could have stopped our investigation cold.

4 Router

Having apparently reached our limit in understanding the cause of the suspicious activity, we decided we needed to allow the traffic to proceed and watch the activity. Assuming the administrators of the target system were not keen on allowing an apparent attack to proceed against their system, we needed to modify our network in order to capture the activity as it proceeds against a system that we controlled. Figure 2 shows the modifications we made to observe the potential attack in action.

On the left of Figure 2 is the design of our original network. Our internal network uses the address block 128.120.56.0/24, and the gray circle at the bottom represents the Windows XP system that is attempting to connect to the target on port 445. Our internal network uses a Linksys firewall/NAT box to connect to the Internet, and the external network (204.255.0.0/16) and target host are in the upper right.

On the right of Figure 2 is the modified network. We inserted a Cisco router between the internal network and our external Linksys gateway. The Cisco router has three configurable interfaces: Ethernet0 (labeled E0), FastEthernet0 (F0), and VLAN1 (Vlan1). Our original network was connected to interface E0; normal Internet access would go into Cisco interface E0, out Cisco interface F0, and through the Linksys router. However, traffic bound to the target system would go into Cisco interface E0 and out Cisco interface Vlan1. On our 204.255.0.0/16 network we added a honeypot host at IP address 204.255.139.8 (see Section 5) to accept the connection request from our Windows XP system.

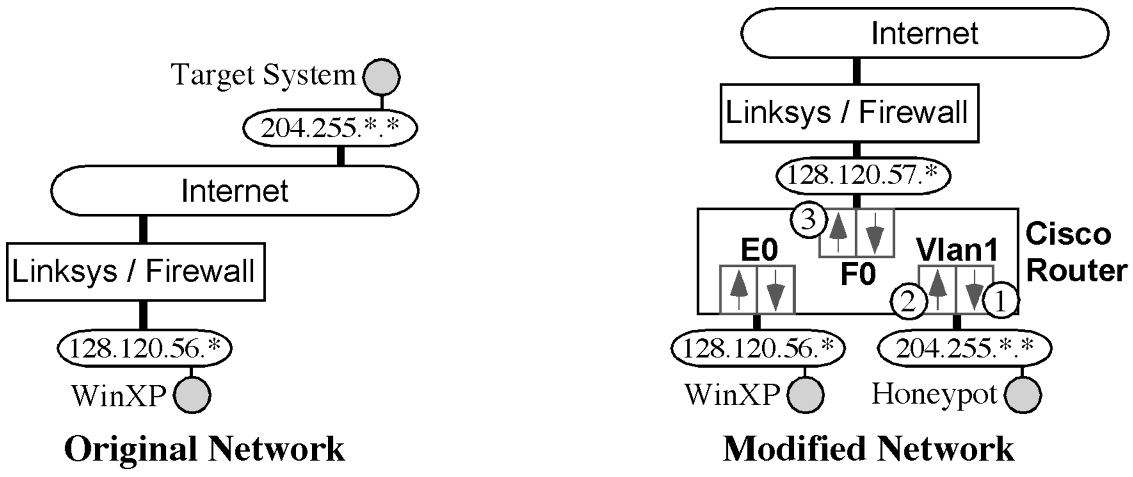


Figure 2: Outbound Honeypot

In case the attempted activity was really an attack, we wanted to take additional steps to contain the spread of the attack. To address this we added two layers of containment. First we “wrapped” our honeypot network with the Cisco router. Cisco supports access control lists (ACLs) for both inbound (into the router) and outbound (out of the router) directions for each interface. Thus, our three-interface router supports up to six ACLs. We specified three ACLs, labeled (1), (2), and (3) in Figure 2. ACL (1) controls access into the honeypot network; ACL (2) controls access out of the honeypot network; and ACL (3) controls access out of interface F0 and to the rest of the Internet.

As shown in Figure 3, ACL (1) allows both TCP and UDP packets into the fishbowl network, but records the sessions in the data structure *into_fishbowl_table*. ACL (2) allows outbound UDP traffic for Network Time Protocol (port 123) and DNS (port 53). This allows the honeypot to set its clock and resolve any IP addresses to IP names. The third rule in ACL (2) allows outbound packets if they were part of a previous inbound session request identified in the into_fishbowl_table data structure.

Together, ACL lists (1) and (2) allow any inbound connections to the fishbowl network but block any outbound traffic not associated with a previous inbound connection request (with the exception of outbound NTP and DNS requests). This prevents a penetrated server from spreading an attack.

① **Into Fishbowl**

```
permit tcp any any reflect into_fishbowl_table
permit udp any any reflect into_fishbowl_table
```

② **Out of Fishbowl**

```
permit udp any any eq 123
permit udp any any eq 53
evaluate into_fishbowl_table
```

③ **Out to Internet**

```
deny tcp any any 135      permit tcp any any
deny tcp any any 137      permit icmp any any
deny tcp any any 138      permit ip any any
deny tcp any any 139
deny tcp any any 445
```

Figure 3: Cisco Access Control List Rules

The final Access Control List, ACL (3), is designed to block the potential spread of an attack from the Windows XP machine in our network to other hosts. (Note: The ACL rules on the right side in ACL (3) in Figure 3 actually belong below the rules on the left.) While we had only observed outbound connections to the one target IP address

(204.255139.8), we were concerned that once we allowed the connection request to go through, the (potential) attack code would then try subsequent IP addresses. The first five rules block outbound TCP connections to commonly attacked Windows ports (135, 137, 138, 139, and 445) (Note: Although we had only observed TCP traffic, to be safe we should have also blocked UDP traffic). The last three rules allow additional IP traffic out to the Internet – the order reflects our gradual opening up of outbound access, as the last rule “permit ip” subsumes the previous two rules.

Together, these three access control lists (and implicit routing rules) direct the suspicious traffic into our honeypot network, bottle up the honeypot in case the attack succeeds and tries to attack additional systems, and prevents the suspicious Windows XP machine from attacking any other IP addresses on the Internet.

5 Honeypot

With the router in place to redirect the suspicious traffic to our own honeypot network, we needed to set up a honeypot host to accept the suspicious traffic. To be extra safe, we implemented our first honeypot on a Mac OS 10 system running our own *Packet Sucker* program. The *Packet Sucker* program simply accepts a connection, reads the first set of data from the client, and closes the connection. Since many attacks occur in the first packet, this extremely simple honeypot has been effective at capturing many attacks on many server ports.

Unfortunately in this case no attack was apparent in the first packet. The suspicious host activity only sent a legitimate initial SMB dialect negotiation request in the first packet. If the source of the activity was indeed an attack, we needed a higher fidelity honeypot that could provide the correct response to the SMB negotiation request. We set up a second honeypot that was an unpatched Windows XP system with file sharing turned on. This system would provide the correct SMB negotiation response and had a high probability of the system being vulnerable to the attack.

This second honeypot allowed the suspicious activity to proceed further, but still no obvious attack was observed. The suspicious Windows XP system attempted to mount the directory “NAVYSUMMARY” on the target system, but our honeypot system responded with an “access denied” message.

For the third experiment we added the directory NAVYSUMMARY to the honeypot system. Unfortunately, even with this directory the target system issued an “access denied” message when the suspicious activity tried to mount the NAVYSUMMARY directory. No obvious sign of an attack was observed. Had we not set up the honeypot correctly? Was a higher fidelity honeypot needed before the attack would proceeded?

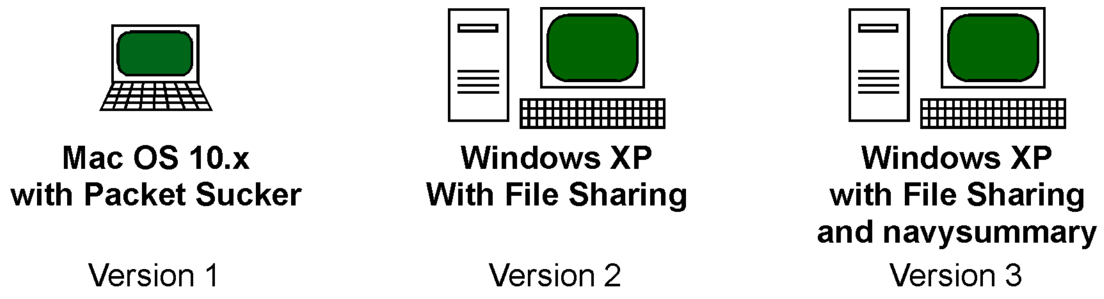


Figure 4: Progression of Honeypots

6 Packet Trace

Figure 5 shows a simplified version of the packet traces for the three versions of the honeypots described in Section 5. An arrow from left to right indicates that the message is sent from the client (the suspicious machine) to the server (the apparent target), and an arrow from right to left indicates that the message is from the server back to the client. The SMB protocol is complex, and the traces are somewhat simplified. For example, the session request and response messages involve more than two messages.

Version 1 on the left only observed the first SMB packet from the client – the client sends a list of SMB dialects that it understands and requests the server to pick one. Version 2, with a full Windows XP system and file sharing turned on, allows the SMB activity to continue, but the session fails when the client tries to connect to the directory NAVYSUMMARY. Version 3, with the directory NAVYSUMMARY added to the honeypot, also fails when the client tries to mount the directory.

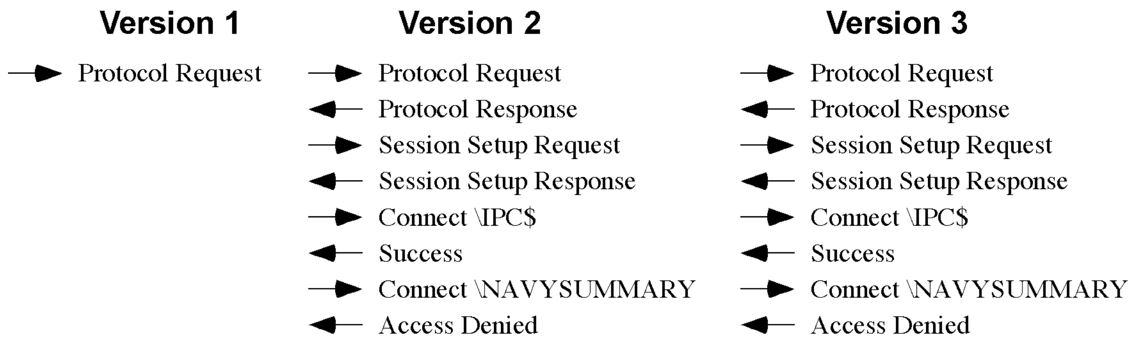


Figure 5: Packet Traces for Honeypot Versions

While we could not get the client to successfully mount the requested directory on the server, the packet traces at least gave us some additional information: the activity was specifically looking for the directory NAVYSUMMARY.

7 Host Evidence

With the information gathered from the packet trace, we returned to our Windows XP system and searched for “NAVYSUMMARY”. As shown in Figure 6 we discovered a mount point for the directory navysummary on the host www.navysbir.brtrc.com in the folder “My Network Places”. Pulling up additional information on the mount point shows that it was created on June 23, 2003 (see Figure 7).

Further searching on the Windows XP system uncovered an email sent by the Navy telling us to go to host www.navysbir.com (the names www.navysbir.com and www.navysbir.brtrc.com resolve to the same host) (see Figure 8). Furthermore, the email arrived on June 23, 2003.

Further searches on the Internet showed that NAVYSUMMARY to be a legitimate directory on the target web site. In this directory are several Microsoft Word templates for creating reports for the Navy. Furthermore there are several approaches using HTML (including the SMB URI) to create mount points in a client’s “My Network Places” directory. Thus, we have a mechanism for creating the mount point on our system.

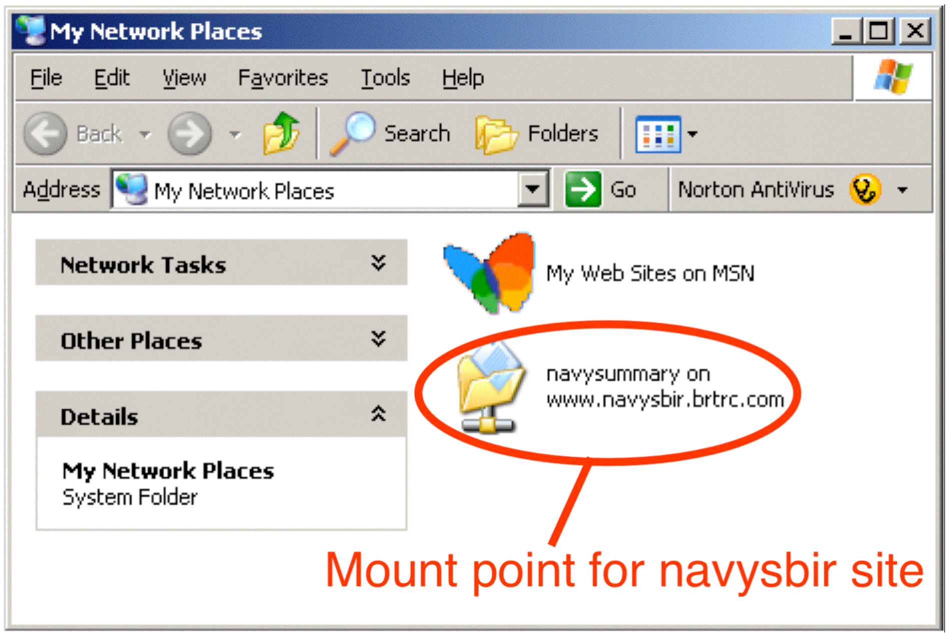
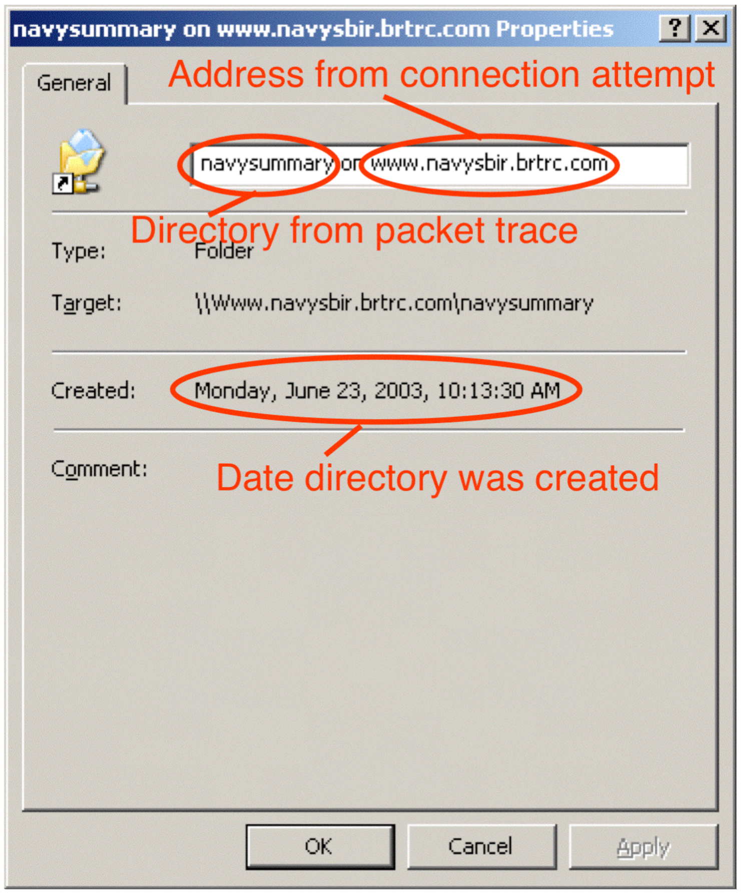

My Network Places
File Edit View Favorites Tools Help
Back Search Folders
Address My Network Places Go Norton AntiVirus
Network Tasks
Other Places
Details
My Network Places
System Folder
My Web Sites on MSN
navysummary on www.navysbir.brtrc.com
Mount point for navysbir site

Figure 6: Mount point - first proximate cause


navysummary on www.navysbir.brtrc.com Properties
General
Address from connection attempt
Directory from packet trace
Date directory was created
OK Cancel Apply

navysummary on www.navysbir.brtrc.com

Type: Folder

Target: \\Www.navysbir.brtrc.com\navysummary

Created: Monday, June 23, 2003, 10:13:30 AM

Comment:

Figure 7: Details, including creation date, of mount point

```
From: "John Williams" <williajr@onr.navy.mil>
Subject: RECOMMENDATION FOR PHASE I AWARD UNDER DOD 2003 STTR
SOLICITATION

...
Since you submitted your proposal electronically I will need
you to fax me a signed copy of appendix A/B (by all three
parties) at (703) 696-4884 by 2 July 2003.  I have attached a
copy to this email.  If you can not make this date please
contact me.  I would also like you to start working on your
Intellectual Property Rights Agreement between your firm and
the Research Institution.  Once this is completed please
forward those to me by mail, fax, or e-mail.  My mailing
address can be found at www.navysbir.com under POC's & Links,
then POC's.
...
```

Figure 8: Email from June 23, 2003

8 Repeatability of Events

The mount point discovered on our Windows XP system showed why the Navy's host and the specific directory NAVYSUMMARY were targeted, but it still did not indicate what caused the network traffic. On the assumption that something was "tickling" the mount point in the "My Network Places" directory, we started testing the AntiVirus software. As it turns out, we can force the Windows XP system to generate the exact same suspicious network traffic by requesting Symantec's latest AntiVirus software to scan the hard disk.

Interestingly, while running the AntiVirus software causes the attempt to mount the NAVYSUMMARY directory to be rejected by our honeypot (now observed under controlled conditions), double clicking on the mount point in the "My Network Places" successfully mounts the NAVYSUMMARY directory. In other words, the honeypot was set up correctly, and the mount request by the Windows XP client was slightly different when generated by the AntiVirus software (causing a rejected mount) and the double click (causing a successful mount).

9 Probable Scenario

The evidence suggests the following scenario: On June 23, 2003 we received email from the Navy directing us to visit the targeted system. The user on the Windows XP visited the site and at some point visited the NAVYSUMMARY directory, probably to pull down templates for submitting reports for the new contract that was about to start. Visiting this web site caused an SMB mount point to be added to the "My Network Places" folder. A year and a half later the new AntiVirus software automatically (and in the background) occasionally scanned the "My Network Places" folder and the mount point in the folder causing the attempted outbound connection to the target system, and this network activity (generated by the AntiVirus software) was detected and blocked by the host-based firewall (from the same company!).

Thus in the final analysis, the activity we have been observing is *not* associated with an attack.

10 Lessons Learned

The following summarizes some of the lessons learned from this series of experiments:

- **Anomalous events are not necessarily malicious.** In this case, what initially appeared as a very suspicious event turned out to have an innocent explanation.
- **Anomaly reports do not identify the underlying cause of the anomaly.** There may be many reasons for the cause of an anomalous event (in our case it was an interaction between an old visit to a web site and the two security software systems), but to track it down may require substantial investment of time and resources.
- **Session event records are insufficient.** The original anomaly report contained session-type information (source and destination addresses and destination port), and on the surface this looked very suspicious. A full packet trace showed no specific attack activity, and it provided additional clues to resolve the issue. The simple session information was misleading.
- **Outbound honeypots are needed.** Most honeypots are designed to capture inbound traffic, but this event showed the need for honeypots to capture suspicious outbound traffic. Outbound honeypots require tight integration with the routing fabric to redirect the specific suspicious activity.
- **High fidelity honeypots are needed.** The simple packet sucker honeypot was insufficient to resolve the problem. A high fidelity honeypot, including the same IP address, host name, and directory structure, was necessary to capture the relevant packet flow data.
- **Knowledge of the user's activities was important.** We were familiar with the old Navy contract, and this led us to search the email archives which included critical information.
- **Host information is critical.** While the simple host session information (address and port data) was clearly insufficient, even the full packet trace information was insufficient to resolve this problem. The discovery of the mount point on the host's file system, the old email, and the knowledge of the change of security software (new AntiVirus software and firewall) were all critical to identifying and understanding the underlying cause of the suspicious activity.
- **We are losing network visibility.** Much of today's deployed intrusion detection is based on network traffic analysis, but we losing our ability to perform meaningful analysis. In addition to encrypted traffic such as ssh and https, an increasing amount of traffic is flowing over switched networks as opposed to broadcast networks, especially for internal traffic. This is largely driven by economics. When building our honeypot network we visited the local electronics store to buy an Ethernet hub, and they only had switched hubs. Even in our small network, maintaining visibility of all host activity may soon become impossible without moving sensors onto every host.

11 Conclusions

This paper follows up on a previous paper titled “Why Anomaly Detection Sucks”. Security software on one of our Windows XP systems alerted us that the system was attempting to connect to port 445, a port with many known vulnerabilities and many known attacks and worms. This report of unexpected activity (which the security system recommended we allow to proceed) left us feeling that our system had been successfully penetrated, but we could not find definitive proof of the problem. Furthermore, without specific information, we had no way to clean up our system.

This paper describes our additional research to identify the underlying cause of the activity that generated the alert. To gather more information we needed to allow the apparent attack to proceed, but we needed to do this safely and in a controlled environment. We did this by building a honeypot to capture the outbound traffic. Section 4 describes how we modified our network infrastructure to create a network for the targeted system and route the suspicious traffic to this network. Furthermore, we added access control list rules to prevent the potential attack from spreading beyond the penetrated honeypot or even our own Windows XP system.

Section 5 described our honeypot as we revised it to capture additional details about the potential attack. Section 6 describes the packet traces that provided the key evidence that helped “break the case” – namely the NAVYSUMMARY directory that the suspicious activity attempted to access. Section 7 describes our return to the host for additional evidence on the original Windows XP system. Using the NAVYSUMMARY name we found a mount point on the system for that directory on the target host, and using the date of the creation of the mount point we discovered an email that directed the user to visit the targeted web site. Section 8 describes how we could reliably repeat the suspicious activity by running the AntiVirus software.

With all this evidence in hand, Section 9 describes our hypothesis for the cause of the underlying activity that generated the original alert. The first piece of the puzzle began with an email message in June of 2003, and the suspicious alert was generated by the interaction of two security software systems running on our host and the residue left on the file system after visiting the Navy’s web site. Finally, Section 10 summarizes the lessons learned from this exercise.