Cleartext Passwords in Linux Memory

Sherri Davidoff
alien@mit.edu

July 26, 2008

Abstract

Upon examination, the memory of a popular Linux distribution contained many cleartext passwords, including login, SSH, Truecrypt, email, IM and root passwords. These passwords are retained by running applications and stored as plain text in memory for extended periods of time. The author investigated the source of these passwords and presents a proof-of-concept method for recovering passwords from memory. Recently, cold boot researchers demonstrated that memory is not as volatile as commonly expected, and that data from memory can be recovered with physical access to systems in a very short period of time. This has opened up a new class of attacks in physical IT security, and significantly raised the risk associated with cleartext passwords in memory.

1 Introduction

Memory contains a concentrated wealth of information, such as usernames, passwords, encryption keys, and personal data. This information can be used by attackers to gain access to related systems, or by forensic investigators to unlock encrypted files and partitions. The author of this paper examined the contents of Linux memory, identified cleartext passwords, investigated their source, and determined a proof-of-concept signature-based method for recovering them.

The Linux test system consistently left cleartext passwords in memory, including the login, SSH, email, IM, Truecrypt and root passwords. While cleartext passwords in memory have been a known, relatively low security risk for many years, the issue has taken on new importance in light of the recent research on cold boot data remnance in memory. In February 2008, a team led by Ed Felten of Princeton University discovered that DRAM is not as volatile as commonly expected. Felten's team studied the lifetime of data in DRAM, and demonstrated that, "[c]ontrary to popular assumption, DRAMs used in most modern computers retain their contents for seconds to minutes after power is lost, even at room temperature and even if removed from a motherboard." Felten's team leveraged this discovery to retrive encryption keys from memory. In their paper, they commented that "[r]esidual data can be recovered using simple, nondestructive techniques that require only momentary physical access to the machine."[1]

In other words, in little more than the time it takes to reboot a system, attackers can dump a system's memory. Memory can be dumped onto a USB key or other device, such as the iPod dumper demonstrated by William Paul and Jacob Appelbaum at the 2008 Cansec West conference. It can also be dumped over the network. When memory contains cleartext passwords, the attacker can retrieve these and use them to gain access to encrypted data, email accounts, servers and other networked systems. Cold boot attacks significantly increase the risk posed by cleartext passwords and other sensitive data in memory.

To demonstrate that cleartext password recovery is achievable, the author of this paper identified bytes which consistently surrounded the cleartext Truecrypt password in memory and used these as a signature to retrieve unknown Truecrypt passwords from memory on another Linux system. This was accomplished using the Memsniff memory searching tools, which were originally developed and released by Sherri Davidoff and Tom Liston at the 2008 Cansec West conference.[2]

[1] Felten, Ed et al. "Lest We Remember: Cold Boot Attacks on Encryption Keys." February, 2008

[2] Davidoff and Liston, "Cold Boot Forensics Workshop," Cansec West 2008,

The author hopes that this paper will provide forensic investigators with a more specific understanding of what information is commonly available for recovery, and help the Linux development community understand and appropriately reduce the presence of cleartext passwords in memory. Future work will include extending this cleartext password analysis to Windows and Mac operating systems, and identifying other signatures and methods for automatically extracting cleartext passwords from memory.

2 Methodology

Testing was conducted on a default installation of Ubuntu 7.10 (2.6.22) with no swap partition. The Coroner's Toolkit, memdump, emacs and build-essentials were also installed. The physical system was a Thinkpad T43 with 2G of physical memory. Two tools were used to capture memory: pcat (from The Coroner's Toolkit), and dd. Pcat was used to copy the memory of a specific process, whereas dd was used to create an image of memory, including areas not associated with a specific process.

During each iteration of the test, the researcher logged into the system and manually launched the following programs: Terminal, su, Thunderbird, Pidgin, GPG, Truecrypt, SSH (in that order). The programs were deliberately not configured to store passwords. Each password was typed manually every time.

After all programs had been launched, process memory was captured live using pcat, and redirected to an external USB drive. The testing time was not held constant, and ranged from five to forty-five minutes. The applications were active when the snapshots were taken. There were a total of twenty-five tests. Please see Appendix A for the detailed testing procedure.

The author also conducted separate experiments in which dd was used to image physical memory, instead of capturing process memory using pcat. However, on the Linux system examined, access to ZONE_HIGHMEM through /dev/mem is restricted, limiting the images to the lower 896M of physical memory (approximately half of the physical memory in use on the test system). Nonetheless, the results of ZONE_NORMAL memory analysis were instructive and have been included in this report. After the data was collected, the author wrote scripts which searched the memory snapshots for the known usernames and passwords. Based on these results, interesting sections of memory were manually examined using a hex editor.

Data	Number of occurrences	Process name
Login password	1	/usr/sbin/gdm
Encrypted login password	3	/usr/sbin/gdm
Email password	1	/usr/lib/thunderbird/thunderbird-bin
IM password	2	pidgin
GPG password	0	
GPG decrypted text	1	gnome-terminal
Truecrypt password	1	truecrypt
SSH password	1	Found in dd image; not specific process memory
Root password	2	Found in dd image; not specific process memory
Encrypted root password	3 3	su su

Summary of results.

3 Findings

This section includes detailed information regarding each of the passwords which were tested. When consistent location information or other patterns were observed, they have been reported.

http://sourceforge.net/projects/memsniff/

Testing was done in a specific, controlled environment, and therefore the location of passwords and surrounding data may differ on other systems.

3.1 Login Password

The Linux login password was found as Ascii text in the Gnome Display Manager process. When the login password was typed correctly on the first try, it appeared 558016 bytes into process memory. Multiple login attempts caused the location of the correct password to shift by approximately 5-10K.

Information from /etc/shadow and /etc/passwd, including the login shadow password, username, long name, UID, GID, home directory, and shell, was also found in the GDM process memory. The shadow password consistently appeared three times within GDM process memory. When the login password was typed correctly on the first try, the shadow password was consistently found at byte offsets 558396, 560684, and 561492.

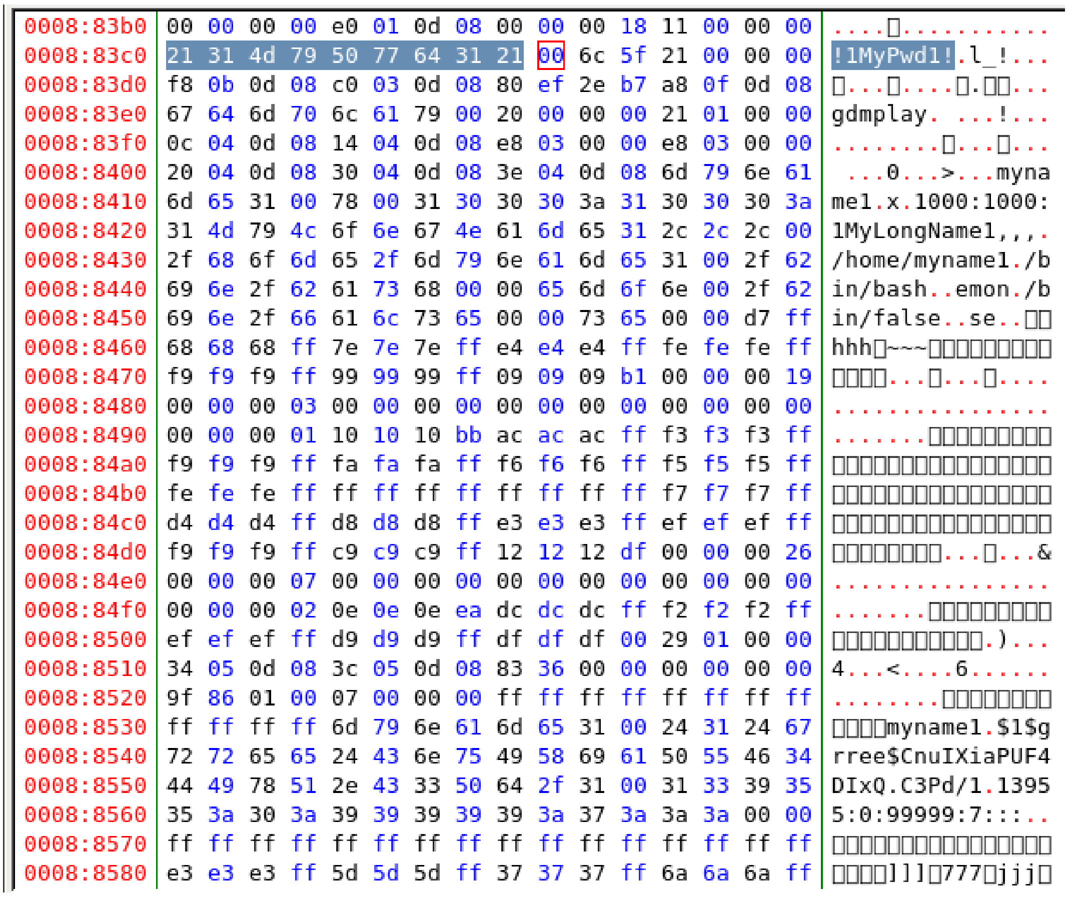

Figure 1: GDM Process Memory viewed in a hex editor. Note the login password (highlighted), as well as the encrypted password and other data from /etc/passwd and /etc/shadow.

3.2 Email Password

The "thunderbird-bin" process memory contained the user's plain text email password, name, email address, mail server and related information in Ascii format.

The email password was stored at least once, and sometimes twice, in thunderbird-bin process memory. It was not consistently found at a specific location, but was consistently found within 0.3M of the 12100000 byte offset (the total thunderbird-bin process memory was around 117M).

As expected, detailed information about the mail server connection, folders and current messages was also contained within the thunderbird-bin process memory.

```
00b6:5900 | 88 42 ee b7 01 00 00 00 14 d9 ba 08 08 00 00 00 | .B□□.....□□.....
00b6:5910 | 01 00 00 00 00 a9 a3 08 6f 7a 69 6c 6c 61 2d 74 | .....□□.ozilla-t
00b6:5920 | 68 75 6e 64 65 72 62 69 72 64 2f 64 32 70 6a 61 | hunderbird/d2pja
00b6:5930 | 38 38 30 2e 21 00 00 00 01 00 00 00 11 00 00 00 | 880.!...........
00b6:5940 | 21 6e 65 77 70 77 6f 6c 21 61 62 00 a0 3b ba 08 | !newpwol!ab.□;□.
00b6:5950 | 20 00 00 00 19 00 00 00 00 00 00 00 00 00 00 00 |  ...............
00b6:5960 | 00 00 00 00 00 00 00 00 18 00 00 00 39 00 00 00 | ............9...
00b6:5970 | c8 ef 35 b5 79 be 62 08 02 00 00 00 10 90 6e 08 | □□5□y□b.......n.
00b6:5980 | 21 00 00 00 d0 9b 9e 08 a0 f2 35 b5 6c f4 35 b5 | !...□...□□5□l□5□
```

Figure 2: Thunderbird-bin Process Memory viewed in a hex editor. The user's email password is highlighted.

3.3 IM Password

The IM password was stored twice in the "pidgin" process memory, as Ascii text. It generally appeared within the first 5M of process memory (total process memory was 62M). The IM username appeared 34 times within the Pidgin process memory.

3.4 Truecrypt Password

Although the graphical checkbox for "Cache passwords and keyfiles in memory" was NOT checked, the Truecrypt password consistently appeared as Ascii text within the Truecrypt process memory. It always appeared at the 4120792 byte offset within process memory. When the password was typed correctly on the first try, the preceding bytes were the same across all tests. (Incorrect password attempts changed the preceding bytes, although not the password location.) Other related information, such as the path to the encrypted volume, was listed shortly after the cleartext password in memory.

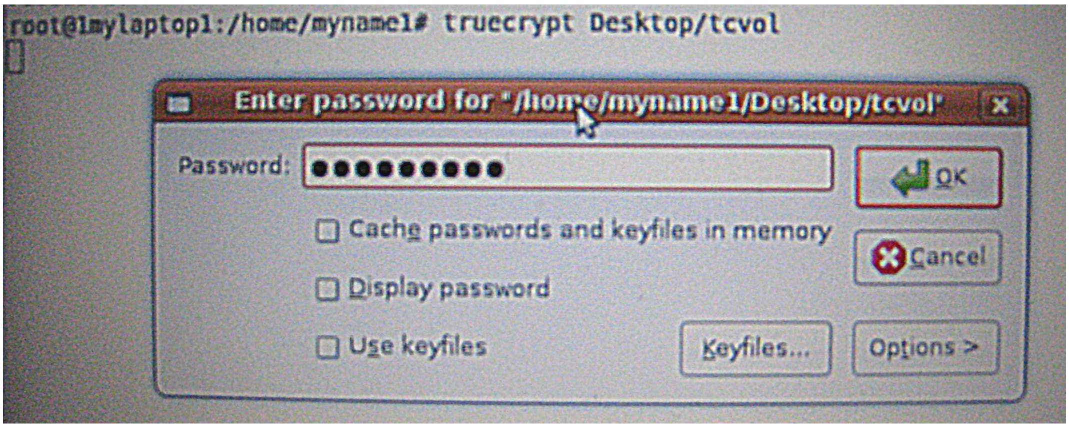

Figure 3: Screenshot of Truecrypt volume being mounted. Note that the "Cache passwords and keyfiles in memory" box is NOT checked, yet the cleartext password was consistently found in process memory.

```
003e:e040  20 5f 43 08 01 00 00 00 20 74 43 08 01 00 00 00   _C..... tC.....
003e:e050  41 6c 6c 6f 77 65 64 00 20 00 00 00 59 00 00 00  Allowed. ...Y...
003e:e060  28 5c 35 08 00 00 00 00 b4 c9 41 08 b4 c9 41 08  (\5.....□□A.□□A.
003e:e070  00 00 00 00 00 00 00 00 d0 76 43 08 18 62 43 08  ........□vC..bC.
003e:e080  00 00 00 00 b8 60 43 08 f0 5f 43 08 c0 5f 43 08  ....□`C.□_C.□_C.
003e:e090  20 61 43 08 01 00 00 00 00 00 00 00 00 00 00 00   aC.............
003e:e0a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003e:e0b0  01 00 00 00 21 00 00 00 b0 d1 35 08 c8 84 36 08  ....!...□□5.□.6.
003e:e0c0  d8 60 43 08 40 00 00 00 09 00 00 00 00 75 6c 6c  □`C.@........ull
003e:e0d0  00 6c 00 00 49 00 00 00 21 6d 79 74 63 76 6f 6c  .l..I...!mytcvol
003e:e0e0  21 24 98 b7 00 00 00 00 b8 67 44 08 00 00 00 00  !$.□....□gD.....
003e:e0f0  00 00 00 00 45 00 00 00 00 00 00 00 00 00 00 00  ....E...........
003e:e100  00 00 00 00 00 00 00 00 00 00 00 00 73 5e 4c 48  ............s^LH
003e:e110  aa 16 03 00 fa 00 00 00 00 00 00 00 31 00 00 00  □...□.......1...
003e:e120  08 d7 2d 08 01 00 00 00 00 00 00 00 01 00 00 00  .□-.............
003e:e130  00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00  ................
003e:e140  00 00 00 00 00 00 00 00 6d 00 00 00 39 00 00 00  ........m...9...
003e:e150  88 9c 35 08 00 00 00 00 00 00 00 00 00 00 00 00  ..5.............
003e:e160  00 00 00 00 00 00 00 00 00 00 00 00 c8 84 36 08  ............□.6.
003e:e170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003e:e180  00 00 00 00 11 00 00 00 30 75 43 08 d0 7a 43 08  ........0uC.□zC.
003e:e190  00 00 00 00 81 00 00 00 1b 00 00 00 1b 00 00 00  ................
003e:e1a0  02 00 00 00 2f 00 00 00 68 00 00 00 6f 00 00 00  ..../...h...o...
003e:e1b0  6d 00 00 00 65 00 00 00 2f 00 00 00 6d 00 00 00  m...e.../...m...
003e:e1c0  79 00 00 00 6e 00 00 00 61 00 00 00 6d 00 00 00  y...n...a...m...
003e:e1d0  65 00 00 00 31 00 00 00 2f 00 00 00 44 00 00 00  e...1.../...D...
003e:e1e0  65 00 00 00 73 00 00 00 6b 00 00 00 74 00 00 00  e...s...k...t...
003e:e1f0  6f 00 00 00 70 00 00 00 2f 00 00 00 74 00 00 00  o...p.../...t...
003e:e200  63 00 00 00 76 00 00 00 6f 00 00 00 6c 00 00 00  c...v...o...l...
```

Figure 4: Truecrypt process memory viewed in a hex editor, with the Truecrypt password highlighted. The relative location and preceding bytes were the same for all captures when the password was typed correctly.

3.5 SSH Password

The cleartext SSH password was stored as Ascii text within a large block of nulls approximately 870M into the memory image. It was usually immediately followed by snippets of other commands that had been typed into the same terminal. The SSH password was often trivial to recover using strings because it usually appeared shortly after the SSH command, last login information and a home directory listing.

```
3667:5f70  00 00 00 00 00 00 00 00 78 54 67 f6 00 f0 f9 b7  ........xTg□.□□□
3667:5f80  00 00 fa b7 20 5f 67 f6 25 00 00 00 71 00 00 00  ..□□ _g□%...q...
3667:5f90  29 52 67 f6 38 5f 67 f6 18 80 67 f6 9c 5f 67 f6  )Rg□8_g□..g□._g□
3667:5fa0  9c 5f 67 f6 00 00 00 00 00 00 00 00 00 00 00 00  ._g□............
3667:5fb0  00 00 00 00 00 00 00 00 88 4e 3b c0 00 00 00 00  .........N;□....
3667:5fc0  80 4d 67 f6 00 00 00 00 00 00 00 00 00 00 00 00  .Mg□............
3667:5fd0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:5fe0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:5ff0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:6000  21 6d 79 73 73 68 70 77 64 21 61 62 0a 30 2e 30  !mysshpwd!ab.0.0
3667:6010  2e 31 2e 33 33 0d 2e 67 70 67 0d 00 00 00 00 00  .1.33..gpg......
3667:6020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:6030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:6040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:6050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:6060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:6070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:6080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3667:6090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 5: Cleartext SSH password in memory.

a)
```
911999496 myname1@1mylaptop1:~$ ssh myname2@10.0.1.33
912000005 Last login: Sat Jun  7 17:03:40 2008 from 1mylaptop1.local
912011264 !mysshpwd!ab
```

b)
```
912598019 Last login: Sun Jun  8 18:39:59 2008 from 1mylaptop1.local
912609266 /home/myname1
912629746 /home/myname1
912707576 myname1
912744448 !mysshpwd!ab
```

c)
```
912494000 Last login: Sun Jun  8 19:20:15 2008 from 1mylaptop1.local
912519154 /home/myname1
912592888 myname1
912601074 /home/myname1
912601088 !mysshpwd!ab
```

d)
```
912011782 myname1@1mylaptop1:~$ ssh myname2@10.0.1.33
912012571 Last login: Thu Jun  5 18:46:45 2008 from 1mylaptop1.local
912019442 /home/myname1
912048128 !mysshpwd!ab
```

e)
```
912089437 Last login: Thu Jun  5 17:52:22 2008 from 1mylaptop1.local
912306162 /home/myname1
912343039 _!mysshpwd!ab
```

Figure 6: Screenshot of the Ascii strings which appeared before the SSH password in memory, from five individual tests. The last login data often shortly preceded the cleartext SSH password in memory.

3.6 GPG Data

The GPG passphrase was not found within memory. However, GPG was used within a terminal to decrypt the contents of a file on the desktop, and the contents of the decrypted file were found with other shell data in the gnome-terminal process memory. The GPG long name, email address and comment were also included.

```
6764096 You need a passphrase to unlock the secret key for
6764147 user: "MyGPGName (mygpgcomment) <mygpg@email.com>"
6764198 2048-bit ELG-E key, ID 4C2D69DA, created 2008-03-24 (main key ID 3237840E)
6764274 gpg: gpg-agent is not available in this session
6764322 gpg: encrypted with 2048-bit ELG-E key, ID 4C2D69DA, created 2008-03-24
6764394       "MyGPGName (mygpgcomment) <mygpg@email.com>"
6764445 !memtest1textfile!
6764464 myname1@1mylaptop1:~$ ssh myname2@10.0.1.33
```

Figure 7: Ascii strings from GPG process memory. Note that the decrypted file contents ("!memtest1textfile!" were cached and are displayed here.

3.7 Root Password

The cleartext root password was consistently recovered in Ascii format from a dd image of memory (it was not found in the memory of any specific process gathered using pcat). During testing, the root password was typed into two separate terminals using the "su root" command. It was found one or two times in memory per test, approximately 870M into the memory image.

The root password was stored in a similar location and format to the SSH cleartext password. Both passwords were normally found within a block of nulls, immediately followed by pieces of other text that had been previously or subsequently typed on the keyboard. The cleartext root password was often located shortly after the "su root" command, in an area which also contained other commands and passwords.

```
365f:3f40 00 00 00 00 44 3f 5f f6 44 3f 5f f6 00 00 00 00 ....D?_□D?_□....
365f:3f50 00 00 00 00 00 01 10 00 00 02 20 00 f0 c8 40 f6 .......... .□□@□
365f:3f60 88 4e 3b c0 03 00 00 00 40 e5 a6 f6 00 00 00 00 .N;□....@□□□....
365f:3f70 00 00 00 00 00 00 00 00 20 3f 5f f6 00 60 35 b7 ........ ?_□.`5□
365f:3f80 00 b0 3c b7 00 80 5f f6 25 00 00 00 75 00 00 00 .□<□.._□%...u...
365f:3f90 21 81 5f f6 70 80 5f f6 e0 3e 5f f6 9c 3f 5f f6 !._□p._□□>_□.?_□
365f:3fa0 9c 3f 5f f6 00 00 00 00 00 00 00 00 00 00 00 00 .?_□............
365f:3fb0 00 00 00 00 00 00 00 00 88 4e 3b c0 00 00 00 00 .........N;□....
365f:3fc0 c0 69 9d f6 00 00 00 00 00 00 00 00 00 00 00 00 □i.□............
365f:3fd0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:3fe0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:3ff0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:4000 21 52 6f 6f 74 50 77 64 21 0a 74 72 75 65 63 72 !RootPwd!.truecr
365f:4010 79 70 74 20 44 65 09 74 63 09 76 09 0d 00 00 00 ypt De.tc.v.....
365f:4020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:4030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:4040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:4050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:4060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:4070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:4080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
365f:4090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Figure 8: Cleartext root password in memory.

Note: the encrypted root password is also stored in the process memory of the "su" program.

```
912285722 myname1@1mylaptop1:~$ su root
912285766 ]0;root@1mylaptop1: /home/myname1
912285800 root@1mylaptop1:/home/myname1# truecrypt Desktop/tcvol
912285858 ]0;root@1mylaptop1: /home/myname1
912285892 root@1mylaptop1:/home/myname1#
912289792 !RootPwd!
```

a)

```
912211994 myname1@1mylaptop1:~$ su root
912212074 ]0;myname1@1mylaptop1: ~
912212099 myname1@1mylaptop1:~$ su root
912212143 ]0;root@1mylaptop1: /home/myname1
912212177 root@1mylaptop1:/home/myname1# ps -deaf > linux21.ps; for i
in `ps -deaf | grep -
912216064 !RootPwd!
912441330 /home/myname1
912457714 /home/myname1
912478208 !mysshpwd!ab
912580608 !RootPwd!
```

b)

```
912216090 myname1@1mylaptop1:~$ su root
912216134 ]0;root@1mylaptop1: /home/myname1
912216168 root@1mylaptop1:/home/myname1# truecrypt Desktop/tc
912216227 ]0;root@1mylaptop1: /home/myname1
912216261 root@1mylaptop1:/home/myname1#
912433138 /home/myname1
912445426 /home/myname1
912445441 ]0;myname1@1mylaptop1: ~
912445466 myname1@1mylaptop1:~$ gpg -d mygpgt
912445602 user: "MyGPGName (mygpgcomment) <mygpg@email.com>"
912445892       "MyGPGName (mygpgcomment) <mygpg@email.com>"
912445944 !memtest1textfile!
912445965 ]0;myname1@1mylaptop1: ~
912445990 myname1@1mylaptop1:~$ ssh myname2@10.0.1.33
912446499 Last login: Sun Jun  8 19:08:03 2008 from 1mylaptop1.local
912457714 /home/myname1
912465906 /home/myname1
912474104 myname1
912478208 !RootPwd!
```

c)

Figure 9: Screenshot of the Ascii strings surrounding the root password in memory, from three individual tests. The command, “su root,” often shortly preceded the root password in memory.

4 Constraints

The test system was deliberately configured without a swap partition to prevent pollution of memory across multiple reboots. The author also removed the battery and power supply from the system in between test runs to lower the risk of power being supplied to memory.

Due to the transient nature of memory, some data was certainly overwritten before and during collection. The data was written to a file system on an external USB drive, and not over the network, so data in the file system cache was modified during data collection. Each command typed into the data collection terminal also modified the shell history stored in memory.

In general, the time that data remains in memory and swap space is highly dependant upon system activity and configuration. Cleartext passwords may remain in systems with swap space for long periods of time. In systems with swap space, the author has recovered passwords from multiple prior reboots.

In the version of Linux tested, there was an 896M limit on /dev/mem access. As a result, approximately 1.1G of physical memory (ZONE_HIGHMEM) was not imaged or examined. The author did examine the ZONE_NORMAL memory which was captured, in addition to process memory. The SSH and root passwords were consistently retrieved from dd images of ZONE_NORMAL memory, although they were not found within a specific process accessible via pcat. The GPG password was not found within process memory or ZONE_NORMAL memory, but it it is possible that it was stored in ZONE_HIGHMEM and was simply not captured.

The author took careful precautions to avoid memory pollution and ensure that each iteration of the test represented a clean slate. Some data in memory may have been overwritten during testing, and some of it was not captured during the collection process. Therefore, it is possible to draw conclusions based on what was found, but not based on what was missing.

5 Password Recovery

There are a number of strategies for recovering passwords from memory. For example:

- Byte Signature: Researchers can look for bytes that are consistently near known password in memory, and use this as a signature to later extract unknown passwords.
- Strings: Researchers can search memory for strings which are often near the password in memory.
- Location: Specific passwords may consistently be found near particular locations in memory (especially relatively within process memory). These areas may be identified and extracted.

For development of a proof-of-concept example, the author focused primarily on the byte signature method for password recovery. It is often possible to identify byte patterns which consistently surround known passwords. The author was successfully able to find a signature for the Truecrypt password, and then used it to recover unknown Truecrypt passwords on another system.

To accomplish this, the author used a hex editor and the Memsniff tools to examine the bytes surrounding the known Truecrypt password in process memory. The author examined process capture from both the test system and another system, with different volume passwords. These immediately revealed striking similarities, as shown in Figure 10.

The bytes immediately preceding the Truecrypt password were:

0? 00 00 00 00 75 6c 6c 00 6c 00 00 49 00 00 00

The “?” represents a nibble that was not consistent. Further experiments revealed that the ? nibble represented the length, in characters, of the Truecrypt password. There was no observed consistency in the bytes immediately following the Truecrypt password.

The author was able to consistently extract the Truecrypt password from memory using the following command:

a)

```
003e:e050  41 6c 6c 6f 77 65 64 00 20 00 00 00 59 00 00 00  Allowed. ...Y...
003e:e060  28 5c 35 08 00 00 00 00 b4 c9 41 08 b4 c9 41 08  (\5.....□□A.□□A.
003e:e070  00 00 00 00 00 00 00 00 d0 76 43 08 18 62 43 08  ........□vC..bC.
003e:e080  00 00 00 00 b8 60 43 08 f0 5f 43 08 c0 5f 43 08  ....□`C.□_C.□_C.
003e:e090  20 61 43 08 01 00 00 00 00 00 00 00 00 00 00 00   aC.............
003e:e0a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003e:e0b0  01 00 00 00 21 00 00 00 b0 d1 35 08 c8 84 36 08  ....!...□□5.□.6.
003e:e0c0  d8 60 43 08 40 00 00 00 09 00 00 00 00 75 6c 6c  □`C.@.......ull
003e:e0d0  00 6c 00 00 49 00 00 00 21 6d 79 74 63 76 6f 6c  .l..I...!mytcvol
003e:e0e0  21 24 98 b7 00 00 00 00 b8 67 44 08 00 00 00 00  !$.□....□gD.....
003e:e0f0  00 00 00 00 45 00 00 00 00 00 00 00 00 00 00 00  ....E...........
003e:e100  00 00 00 00 00 00 00 00 00 00 00 00 73 5e 4c 48  ............s^LH
003e:e110  aa 16 03 00 fa 00 00 00 00 00 00 00 31 00 00 00  □...□.......1...
003e:e120  08 d7 2d 08 01 00 00 00 00 00 00 00 01 00 00 00  .□-.............
003e:e130  00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00  ................
003e:e140  00 00 00 00 00 00 00 00 6d 00 00 00 39 00 00 00  ........m...9...
003e:e150  88 9c 35 08 00 00 00 00 00 00 00 00 00 00 00 00  ..5.............
003e:e160  00 00 00 00 00 00 00 00 00 00 00 00 c8 84 36 08  ............□.6.
003e:e170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003e:e180  00 00 00 00 11 00 00 00 30 75 43 08 d0 7a 43 08  ........0uC.□zC.
003e:e190  00 00 00 00 81 00 00 00 1b 00 00 00 1b 00 00 00  ................
003e:e1a0  02 00 00 00 2f 00 00 00 68 00 00 00 6f 00 00 00  ..../...h...o...
003e:e1b0  6d 00 00 00 65 00 00 00 2f 00 00 00 6d 00 00 00  m...e.../...m...
003e:e1c0  79 00 00 00 6e 00 00 00 61 00 00 00 6d 00 00 00  y...n...a...m...
003e:e1d0  65 00 00 00 31 00 00 00 2f 00 00 00 44 00 00 00  e...1.../...D...
003e:e1e0  65 00 00 00 73 00 00 00 6b 00 00 00 74 00 00 00  e...s...k...t...
003e:e1f0  6f 00 00 00 70 00 00 00 2f 00 00 00 74 00 00 00  o...p.../...t...
003e:e200  63 00 00 00 76 00 00 00 6f 00 00 00 6c 00 00 00  c...v...o...l...
```

b)

```
003e:e460  10 00 00 00 59 00 00 00 28 5c 35 08 00 00 00 00  ....Y...(\5.....
003e:e470  b4 c9 41 08 b4 c9 41 08 00 00 00 00 00 00 00 00  □□A.□□A.........
003e:e480  b8 63 43 08 40 66 43 08 00 00 00 00 c0 64 43 08  □cC.@fC.....□dC.
003e:e490  f8 63 43 08 c8 63 43 08 28 65 43 08 01 00 00 00  □cC.□cC.(eC.....
003e:e4a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003e:e4b0  00 00 00 00 00 00 00 00 0a 00 00 00 21 00 00 00  ............!...
003e:e4c0  b0 d1 35 08 c8 84 36 08 e0 64 43 08 40 00 00 00  □□5.□.6.□dC.@...
003e:e4d0  0b 00 00 00 00 75 6c 6c 00 6c 00 00 49 00 00 00  .....ull.l..I...
003e:e4e0  21 77 6f 77 74 63 63 6f 6f 6c 21 00 00 00 00 00  !wowtccool!.....
003e:e4f0  00 00 00 00 0c 00 00 00 00 00 00 00 00 00 00 00  ................
003e:e500  00 00 00 00 00 00 f0 3f 00 00 00 00 00 00 00 00  ......□?........
003e:e510  00 00 00 00 00 00 00 00 00 00 00 00 00 00 f0 3f  ..............□?
003e:e520  00 00 00 00 31 00 00 00 08 d7 2d 08 01 00 00 00  ....1....□-.....
003e:e530  00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00  ................
003e:e540  00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00  ................
003e:e550  6e 00 00 00 39 00 00 00 88 9c 35 08 00 00 00 00  n...9.....5.....
003e:e560  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003e:e570  00 00 00 00 c8 84 36 08 00 00 00 00 00 00 00 00  ....□.6.........
003e:e580  00 00 00 00 00 00 00 00 00 00 00 00 11 00 00 00  ................
003e:e590  b0 68 43 08 b8 7e 43 08 00 00 00 00 11 00 00 00  □hC.□~C.........
003e:e5a0  50 7f 43 08 68 7b 43 08 10 00 00 00 91 00 00 00  P.C.h{C.........
003e:e5b0  1f 00 00 00 1f 00 00 00 02 00 00 00 2f 00 00 00  ............/...
003e:e5c0  68 00 00 00 6f 00 00 00 6d 00 00 00 65 00 00 00  h...o...m...e...
003e:e5d0  2f 00 00 00 61 00 00 00 6c 00 00 00 69 00 00 00  /...a...l...i...
003e:e5e0  65 00 00 00 6e 00 00 00 2f 00 00 00 44 00 00 00  e...n.../...D...
003e:e5f0  65 00 00 00 73 00 00 00 6b 00 00 00 74 00 00 00  e...s...k...t...
003e:e600  6f 00 00 00 70 00 00 00 2f 00 00 00 74 00 00 00  o...p.../...t...
003e:e610  63 00 00 00 63 00 00 00 6f 00 00 00 6f 00 00 00  c...c...o...o...
```

Figure 10: Two examples of Truecrypt process memory viewed in a hex editor, with the signature highlighted. The password immediately follows the signature, and begins and ends with an exclamation point. Note that the byte immediately before the signature appears to correspond with the password length. Example a) was collected from the Ubuntu 7.10 test system, whereas example b) was collected from an Ubuntu 8.04 distribution. Both systems were running Truecrypt 5.1a.

```
# memdump | memsniff.pl -b "00 00 00 00 75 6c 6c 00 6c 00 00 49 00 00 00"
```

This command prints the characters immediately following the given signature in memory. The author tested this out on a second Linux system, running a different version of Ubuntu, and was able to automatically extract unknown Truecrypt passwords from memory.

6 Analysis

The presence of cleartext passwords in memory increases the risk that loss of a single laptop, or the compromise of a single desktop, will lead to the compromise of other systems. This is especially noteworthy due to the recent development of "cold boot" memory dumping attacks, which in certain cases facilitate unprivileged access to system memory. This could allow insiders, such as low-level employees, to steal administrative credentials, or help mobile device theives gain access to encrypted files and related systems.

In this paper, common Linux programs were analyzed for cleartext password retention. In all but one case, the cleartext password was present in memory and remained there consistently for the duration of the test (up to forty-five minutes). Although for the purposes of testing the system did not include swap space, most Linux systems do contain swap space, and this can dramatically extend the lifetime of passwords. Data which is swapped out is written to disk, and may include cleartext passwords. In casual testing, the author has recovered data from swap space which was last used months previously.

It is difficult to ensure that passwords are not retained in system memory. A computer is a very complex environment, and application developers have only limited control. Passwords may be cached for many different reasons, not simply by the application itself. To address the issue of cleartext passwords in memory, developers would need to examine application code, the effect of compiler optimizations, shared libraries, and operating system code.

In some cases, retaining cleartext passwords may be a deliberate design choice. For example, it is possible that the developers of Thunderbird chose to retain the cleartext password in memory to facilitiate automated email retrieval. At a minimum, obfuscating the password rather than storing it in plain text would prevent an attacker from using a listing of memory strings to conduct a successful brute force attack.

In other cases, the applications tested were written by programmers who undoubtedly had security considerations in mind and endeavored to avoid retaining passwords in memory. SSH is one such example. The fact that the SSH password was never found in the SSH process memory itself indicates that the application programmers were careful to ensure that it was not stored there. Nevertheless, the SSH password still appeared as plain text outside of process memory. Similarly, the authors of Truecrypt undoubtedly considered the question of whether to store passwords and keyfiles in memory, because it is listed as an option for the user every time a volume is mounted. However, regardless of whether the box was checked, the password did appear in process memory.

To prevent passwords from being retained in memory, software developers would need to overwrite memory once the password is no longer needed. This would have to be enforced not just by the application memory itself, but in all the libraries and operating system code which also handle the password. Quickly overwriting passwords in memory would minimize the risk of capture via physical access, cold boot techniques, swap space forensics or simple, live, privileged memory captures.

The author's initial analysis of the location and data surrounding cleartext passwords in memory indicates that it will be possible to develop effective retrieval tools. At a minimum, strings from memory can be used to create a dictionary for cracking passwords, and it is likely that signatures can be developed to quickly find and extract passwords from memory.

A proof-of-concept signature-based memory sniffer called "Memsniff" (originally "DaisyDukes") was presented by Sherri Davidoff and Tom Liston at the 2008 Cansec West conference. The presenters were able to extract the Outlook Express 6 from a Windows memory image based on consistent bytes located before and after the password in memory.

In this paper, the author used the Memsniff tools to consistently retrieve cleartext Truecrypt passwords from Linux memory. The author has also created a Sourceforge project called "memsniff" for development of a bootable memory sniffer, and invites interested parties to join the project:

http://sourceforge.net/projects/memsniff/

Snapshots of interesting process memory that were used in this project are publicly available for download at:

http://philosecurity.org/research/cleartext-passwords-linux/

7 Conclusion

Cleartext passwords in memory increase the risk that a single compromised desktop or laptop will be used as a gateway for compromising related accounts, machines and applications. This is particularly true given Felten et al's recent report on cold boot data remnance in memory, which implies that in certain cases, anyone with physical access to a system can recover contents of memory.

In this report, the author provides detailed location and context information for cleatext passwords in Linux memory. This includes Linux login and root passwords, as well as Thunderbird, Pidgin, Truecrypt and SSH passwords. The author also presents a proof-of-concept demonstration for consistently recovering the cleartext Truecrypt password from a Linux memory dump. The "Memsniff" tools used for the demonstration are available on Sourceforge.

It is the author's hope that this paper is useful to forensic analysts and the Linux development community.

8 Thanks

The author would like to thank the following individuals for their technical and editing assistance: Jacob Appelbaum, Blake Brasher, Aaron Culich, Matt Knox, Tom Liston, Matt Malchano, Eric Michaud, Seth David Schoen, Weitse Venema.

References

[1] Jacob Appelbaum, *Advanced Memory Forensics: Releasing the Cold Boot Utilities*, The Last HOPE, New York, New York, July 2008

[2] Mariusz Burdach, *Finding Digital Evidence in Physical Memory*, Black Hat, Las Vegas, NV 2006

[3] Ed Felten et al, *Lest We Remember: Cold Boot Attacks on Encryption Keys*, http://citp.princeton.edu/pub/coldboot.pdf February 2006

[4] Jorge Urrea, *An Analysis of Linux RAM Forensics*, Naval Postgraduate School Thesis, Monterey, California March 2006

A Appendix A - Testing Procedure

- Unplug laptop power cord and ensure battery is removed.
- Plug in laptop
- Boot into the most recent Ubuntu kernel
- Login (Enter login password)
- Plug in external drive (automount)
- Open three terminals.
- In two terminals, type "su root," and enter the root password for each.
- One of the root terminals will be used exclusively for capturing data from the system. We will refer to this as the "data collection terminal."
- In the data collection terminal, change to the current data capture directory. Create it using "mkdir" if it does not already exist.
- Start Thunderbird by clicking on Applications→ Internet→ Thunderbird. Click "Get Mail." Enter password for default account.
- Start Pidgin by clicking on Applications→ Internet→ Pidgin. Enter password for default account.
- In the user terminal, type:
  $ gpg -d mytestfile1.txt.gpg
  Enter the GPG passphrase.
- In the root terminal, type:
  # truecrypt Desktop/tcvol
  Enter the Truecrypt volume password.
- In the user terminal, type:
  $ ssh myname2@10.0.1.33
  Enter the SSH password.
- Collect the process listing. In the data collection terminal, type:
  # ps -deaf > linux??.ps
  (Replace the ?? with the appropriate test identifier.)
- Collect a memory image or capture process memory using pcat, as appropriate.
- Shut down the system.

B Appendix B - OS and Application Information

- Operating System: Ubuntu 7.10 (10/2007) 2.6.22-14-generic #1 SMP Sun Oct 14 23:05:12 GMT 2007 i686 GNU/Linux
- Gnome: Version: 2.20.0, Distributor: Ubuntu, Build Date: 9/17/2007
- SSH: OpenSSH_4.6p1 Debian-5build1, OpenSSL 0.9.8e 23 Feb 2007
- Pidgin: Version 2.2.1
- Thunderbird: Version 2.0.0.6 (20071008)
- GPG: gpg (GnuPG) 1.4.6
- Truecrypt: 5.1a