Discovering passwords in the memory

Abhishek Kumar (abhishek.kumar@paladion.net)

November 2003

Escalation of privileges is a common method of attack where a low privileged user exploits a vulnerability to become an administrator or a higher privileged user. Privilege escalation may be achieved through cracking of administrative passwords, local buffer overflows and stealing of passwords. This paper discusses a common vulnerability that could be exploited by low privileged users to steal critical passwords and escalate their privileges. While this vulnerability has been known for several years, our research indicates that a large number of applications are still vulnerable to this flaw. As of this writing, we have informed the software vendors about the vulnerability, and are working with them to fix it.

Understanding the vulnerability

While servers and applications store passwords encrypted or in digest form in the hard disk, we have seen several instances when such encryption is not applied while storing passwords in memory. Frequently access to memory is not restricted based on privilege levels. Thus attackers with local access to the system can read the memory and extract passwords. Using a memory viewer[1] they can locate a specific process in memory and read its contents that can include passwords. These passwords could be an administrator password for a server, a user password for an application, or a database login password. Once a password is discovered attackers could escalate their privileges in the application. Thus any application that uses password for authentication could be vulnerable if it leaves the password unencrypted in memory.

[1] Memory viewers as the name implies are tools used to read the memory of the system. Many of these graphical tools let you select a process and walk through its memory space. WinHex is one such memory viewer that we used - it is available from: http://www.x-ways.net/index-e.html

Locating the password in memory

A memory viewer displays all the code and data associated with a process in memory. The data is generally huge and may include both encrypted as well as plain text data.

Passwords can be located in memory by following either of the two approaches:

1) By searching for the password at a fixed address in memory - All installations of an application might contain the password in the same fixed location in the memory. For example, all instances of a server might store the password variable in say, location 10BD862C. Once this location is known, then the password can be extracted from the memory if an attacker can read the memory. To discover the location of the password for an application, attackers could install the application in their own system first. Then they could search for the location of the password in memory, as they know the password that has been used. Once they have the location, they could discover the password of any instance of the application that they have local access to.

2) By locating a particular pattern within the memory – The data near the password might follow a pattern. For example, all instances of the server might have a pattern like "auth-password" next to which the password is stored. Attackers could discover this pattern after installing the application in their own system- they could search for their password in the memory viewer and note the pattern near the password. Once they have the pattern, they could discover the password of any instance of the application that they have local access to. Sometimes identifying the password becomes easy because of distinct and identifiable strings in the memory such as 'username' and 'password'. This is illustrated through a screen shot.

When do passwords stay in memory?

Let us take the example of a typical application server to illustrate why passwords appear in memory:

When the application server is started it reads the Java command line arguments and the environment variables and binds itself to a TCP port. The application takes the identity of an operating system user and security group. The application also consults a configuration file to get all its configuration information.

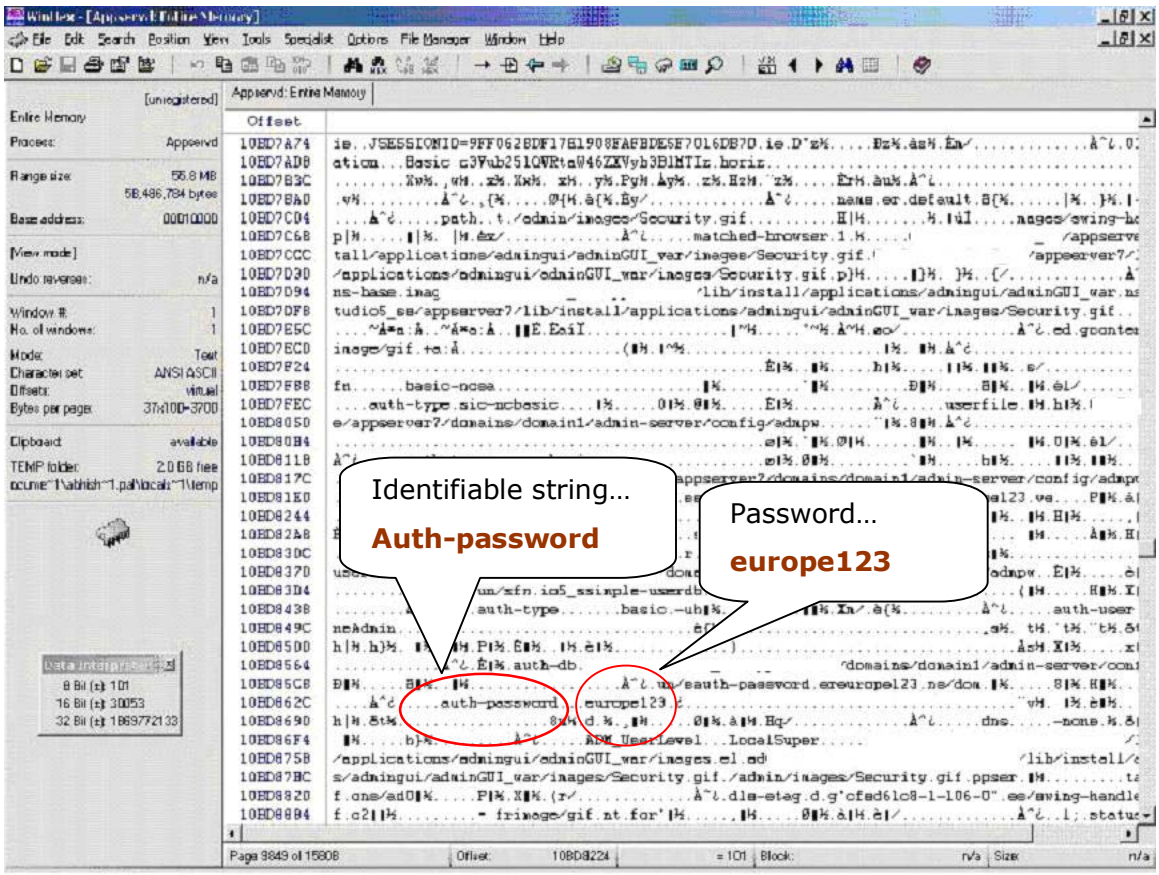
On startup the server loads the administrator password into its memory in plain text. Among other things, the configuration file could provide information on database connection sources which contain database login passwords. The server also loads these passwords into memory in plain text.

Alternatively the passwords may not be loaded when the server starts up but appear in the memory when the administrator or a user is authenticated. Thereafter the passwords remain in the memory even after the user has logged off.

Possible attack scenarios

Here are three scenarios where we found the application could be at risk:

- Consider a server, an application server or a web server. The server loads its administrator password into memory when it starts up. The attackers will use their account to log into the system and open the server process in a memory viewer. Within the memory viewer they could search a fixed address or a known pattern to identify and extract the administrator password. This vulnerability is exploitable even by a low privileged user because the server's data in the memory is not protected based on user privileges.



The above figure shows the administrator password for an application server stored in the memory near an identifiable string.

- In a different situation, the server may not load the administrator password at startup. But the password is loaded in the memory when the administrator is to be authenticated. Thereafter the password may stay in the memory as long as the server is running. Even though the administrator accessed the configuration console of the server from a remote machine, the server's memory contains the password in plain text and can be discovered.

- Here is a situation that may reveal database passwords to an attacker. Consider a server which is used to configure a database connection source. The connection source requires the database username and password to connect to the database. When the server starts up, it loads the database password also into the memory as part of the connection source. This password can also be identified using a memory viewer. In this scenario the application server reveals the password for a database and increases the overall risk.

Conditions favoring stealing of the password

This vulnerability is quite easy to exploit as the attacker is not required to write any exploit code. If attackers have read access to the memory of an application, they could use a memory viewer to extract passwords. Memory viewers are easily available and simple to use. In several applications we tested, access to the memory is not restricted based on user privilege. Hence **low privileged OS users could inflict damage by logging in to their OS account and finding high privileged passwords**. Once the password is discovered it could be used to read and modify sensitive information. It is difficult to track and detect these breaches because only the memory is being accessed to locate the password and no easily detectable trails are left.

Constraints on exploiting the vulnerability

This vulnerability requires local access to be exploited. Attackers need to log in to the system to find the passwords. They also need to install a memory viewer on the victim and that may not be possible always.

Securing the passwords in the memory

This vulnerability results from storing critical information in plain text in the memory. Information in the memory should not be neglected and passwords should be stored encrypted in memory. If the password is used as plain text then it should be immediately reset to an encrypted value in the memory.

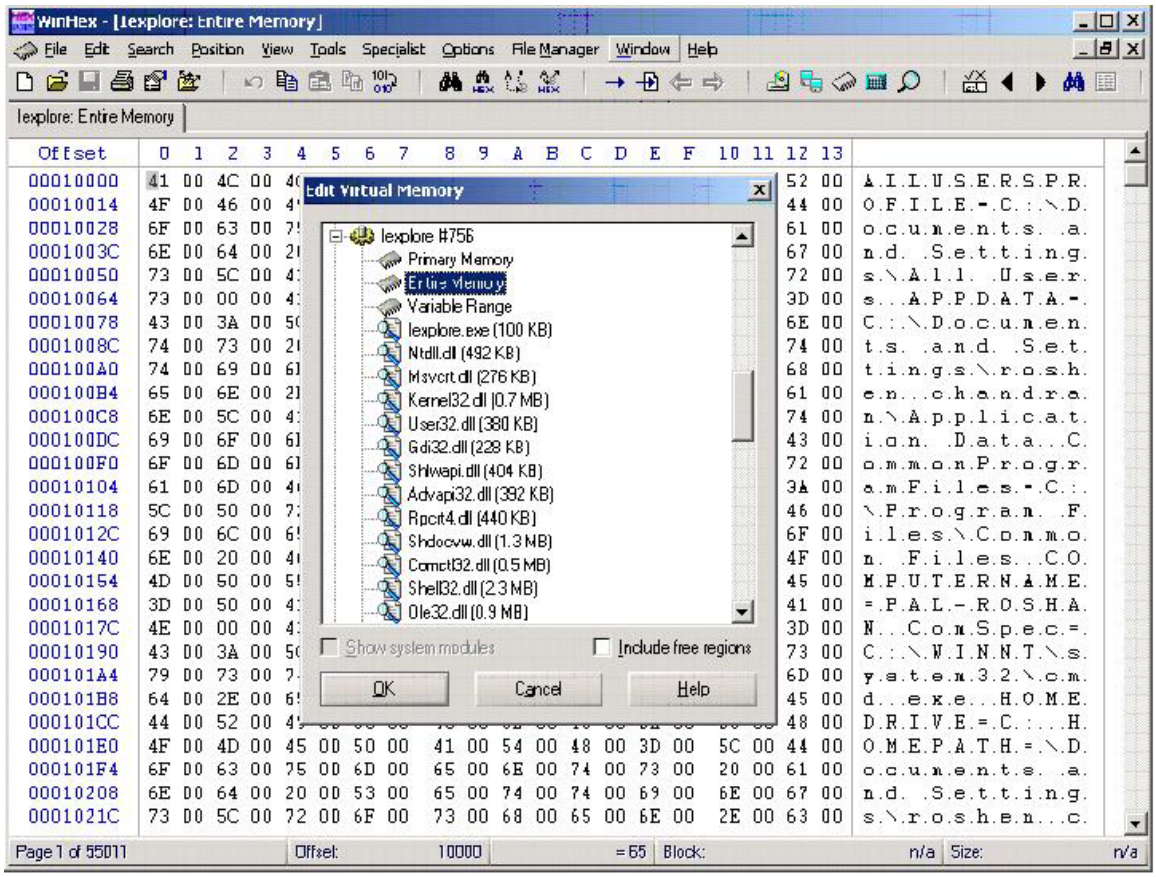
Need for action

Memory related issues are often ignored by developers and hence an application could be compromised by a malicious internal user. It is important that companies and developers consider this issue right from the initial stage of product development. We also suggest that memory testing be made an integral part of all security testing to prevent these vulnerabilities.

Appendix

WinHex

The figure below shows the interface of WinHex. WinHex is a memory viewer that can be purchased at http://www.x-ways.net/index-e.html.



Contact information

Paladion Networks

307, Devarata

Plot No. 83, Sector -17

Vashi, Navi Mumbai – 400 703

Ph: +91 22 55910513

FAX: +91 22 55912429

Website: http://www.paladion.net