**Your Linux Passwords Are in Danger: MimiDove Meets the Challenge**

|  |  |
| --- | --- |
| **Slide 1 Hello**  😊 Hi everyone! 😊  Thanks for inviting me. I am happy to be here.  Today I will be talking about attacks on users’ passwords stored in Linux memory and how to block them.  I think this topic is of crucial importance for all experts dealing with application security. |  |
| **Slide 2 Who we are**  This work was done by one of my students. Svetlana Golub worked on this topic as her bachelor thesis under my supervision. Unfortunately, she is currently unavailable. |  |
| **Slide 3 “Clear text passwords in process memory”**  Passwords authentication is common for modern software. One of the common ways of authentication is to calculate a hash from user-supplied password and compare it with a stored hash. For example, hashes for users’ passwords are stored in the /etc/passwd file.  The final step is to deallocate data used to process this sensitive information.  A perfect example of authentic software is Linux GNOME desktop, which uses gnome-keyring-daemon for storing security credentials such as usernames and passwords.  Security researcher Seong-Joong Kim from South Korea discovered that gnome-keyring 3.18.3 does not  overwrite users’ credentials located in memory. These credentials are not cleared automatically and reside in plaintext in process memory. This happens because there are no routine calls to overwrite sensitive data probably due to unnecessary compiler optimization. As a result, any user with an appropriate privilege can extract them by memory acquisition.  Let’s see how it can happen. |  |
| **Slide 4 “MimiPenguin”**  Security researcher Hunter Gregal from the USA developed a tool called MimiPenguin to extract users’ passwords from the memory of gnome-keyring-daemon. This tool can successfully extract ASCII passwords. |  |
| **Slide 5 “Mimipy”**  Another researcher Nicolas Verdier from the USA has improved MimiPenguin tool and implemented a new tool called Mimipy. This tool can do both locate and remove passwords from memory. The tool can successfully work with ASCII passwords. |  |
| **Slide 6 “Unicode in passwords”**  We analyzed these tools using different accounts in one machine. We tried to use various Unicode passwords.  Here are the corresponding memory dumps of gnome-keyring-daemon software.  First, we proved that gnome-keyring-daemon stores the users’ passwords in memory even if there are Unicode symbols. Here we have two passwords with Cyrillic alphabet as well with as some special symbols, like euro and yuan symbols.  Then we checked if these two tools can reveal and overwrite Unicode passwords. |  |
| **Slide 7 “Downsides of MimiPenguin and MimiDove”**  We can see that both these tools are not able to find these passwords.  Also, both tools work very slowly, which is a serious drawback.  To tackle this challenge, we have developed a tool called {p} MimiDove. |  |
| **Slide 8 “”**  MimiDove expands the MimiPenguin and Mimipy features and includes the following competitive advantages: it can locate passwords containing both ASCII characters and Unicode characters. |  |
| **Slide 9 “MimiDove”**  The next step is to make MimiDove faster to locate passwords as fast as possible.  Finally, we have solved this issue and significantly improve the speed of MimiDove. |  |
| **Slide 10 “MimiDove is Faster”**  We have revealed that users’ passwords are located only in stack memory region, so we can skip other memory regions and focus only on anonymous regions with enabled read and write access. This helps to significantly improve MimiDove performance.  The key steps of MimiDove are here.  First, we read users passwords hashes from /etc/shadow file.  Second, we access ‘gnome-keyring-daemon’ process and enumerate its memory segments.  Then, for we extract the sequences of bytes and calculate their hashes.  If the calculated hash value is match with one of hash from /etc/shadow file, it means that the password has been found.  The final step is to overwrite the detected passwords and protect the OS. |  |
| **Slide 11 “MimiDove”**  Here we can see the memory content before and after overwriting.  Now attackers cannot extract users’ passwords from the memory. |  |
| **Slide 12 “Agenda: PeePeeL illegal enabling”**  The comparison table is here.  We can see that MimiDove has the following advantages: it can locate and overwrite passwords regardless of their alphabet ASCII and UNICODE symbols.  MimiDove has been successfully tested using the following passwords and the following operating systems. |  |
| **Slide 13 “Agenda: PeePeeL illegal enabling”**  Here is the time comparison figure, which shows how faster Mimidove is.  The source code of Mimidove is uploaded here. |  |
| **Slide 14 “Agenda: PeePeeL illegal enabling”**  Thank you. |  |