**German University in Cairo**

**Faculty of Media Engineering and Technology**

**Dr. Amr El-Mougy**

**Eng. Rania Said**

**CSEN 1001: Computer and Network Security**

**Computer Science Department**

**Spring, 2015**

**Milestone 1 Report**

|  |  |
| --- | --- |
| **Team ID: 9** |  |

Please refer to the document of the project teams <http://goo.gl/AdPVZB>

**Motivation/ Purpose of the project**

Intended to hide a remote shell. The rootkit will hide the port used to connect remotely, hide the process of the shell, hide the files on the hard drive. Its goal is to provide the shell with root access to the system. It can be used to acquire more information about the target system such as usernames and passwords, remove sensitive files and data, monitor system activity.

**List of Functionalities**

1. Hide port.
2. Hide process.
3. Hide file.
4. Provide access to root.

**Cryptography in your project**

Not applicable

**Security in your Project**

Provide a detailed comparison between security methods, protocols and algorithms you researched. Identify which ones have you chosen and justify your current choices. (List advantages. disadvantages, and trade-offs

We hook to the system call table, to change the behaviour of existing system call functions.

**Implementation**

If you have started the implementation phase, please identify frameworks/libraries/languages that you are using and justify how they would serve the purpose of your project. If you have not, identify how you intend to implement your project and list briefly your current findings resultant from your search about implementation.

1. We are using the C language as the linux kernel modules are written in C.
2. To hide the rootkit from the system, we remove it from multiple kernel lists such as kobjects list, thereby removing it from the list sys/modules. Also to hide the module from lsmod and the proc filesystfem, we remove it the kernel modules list.
3. To prevent from removal by rmmod, we force the module to be in a busy state.
4. Rootkit provides functionality of receiving commands such as show, release to allow the module to be removed.
5. We are choosing between two approaches for finding the system table:
   1. Retrieving the system call address from the system call interrupt handler by searching the handler hex dump. We retrieve the address of interrupt description table, then we know the offset of the system call handler so we can retrieve the hex dump of the assembly instruction of the handler. By knowing the structure of the instruction, we can locate the address used to access the system call table.
   2. Brute-force: search over all kernel memory, this sacrifices speed for reliability since it is guaranteed to find the system call table.
   3. To identify the correct address in both methods: We compare it to the address of the known system call function sys\_close.
6. Having the system call table, we can modify pointers to system call functions to point to our modified implementations which do the following:

**Dividing Tasks**

Mention all contributions done up till now for each of the team members.

|  |  |
| --- | --- |
| ID | Contributions |
| 22-3798 | Retrieving the system call table using method 1. |
| 22-4408 | Retrieving the system call table using method 2. |
| 22-1452 | Hiding rootkit |
| 22-1483 | Hide process |
| 22-0627 | Hide file |
| 22-2128 | Hide port |