MALWARE IMPLEMENTATION, DETECTION AND PREVENTION

Group 6: Ashish Pandit, Dhivya Govindarajan, Sushant Murdeshwar

B. Thomas Golisano College of Computing & Information, Rochester Institute or Technology

Introduction

1. With the cyberattacks on the rise comes an increased awareness among developers, security specialists and other professionals. To create a better defense, it is important to think “like a hacker” and a defender.
2. The objective of this project is to design and implement a malware to infect a target machine (black hat hacker) and perform detection and prevention techniques on any malware kind (white hat defender).
3. The malware chosen is the keylogger. Keylogger is a technique of recording the users’ keystrokes and storing the information in a log file. It could be software-based or hardware-based. Many user credentials have been stolen through the keyloggers.
4. A careful and detailed analysis of other team’s infected machines are reported using detection mechanisms and tools.

Related Work

1. The work proposed in the Yang et al [1] research paper explains the API-based keylogger using the Windows hooking mechanism.
2. The research paper by Ladakis et al [2] discusses a GPU based keylogger which uses graphic card to records the keystrokes and store the information in its memory space.
3. Adam Howard et al [3] research work proposed a standardized detecting mechanism of the unknown keyloggers in the system taking advantage of the system hooks.

Methodology

Design Considerations

1. The categories of software-based keylogger are several. The API-based keylogger is chosen for this project.
2. The windows hook handles the messaging mechanism. The hook procedure monitors and intercepts the keystrokes events before it reaches the application.
3. The Win32 API is used for this implementation. It offers several hook options for keyboard events, mouse click events, etc.

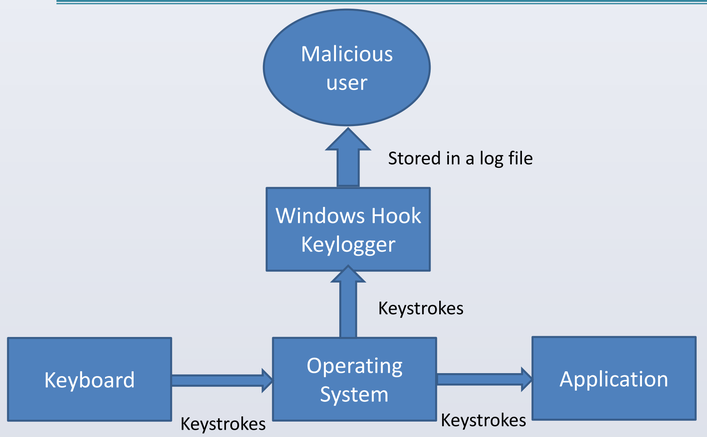

Figure 1. Architecture of the Keylogger malware

Architecture

1. Whenever a key is pressed and released on the keyboard device, the keystrokes are processed by the operating system internally.
2. Before the keystrokes are sent to the user’s application, the virtual key codes are logged in a text file using the windows hooking mechanism.
3. Then the keystrokes are sent to the application.

Implementation

1. Implementation Language : C++
2. Tools used : Qt Creator IDE, Microsoft Windows SDK and RLES environment.
3. Hooking API functions used:
   - SetWindowsHookEx()
   - WH_Keyboard_LL()
   - CallNextHookEx()
   - GetAsyncKeyState()
   - UnhookWindowsHookEx()
4. The tool Hstart is used to hide the console of the executable keylogger file.

Lessons Learned

1. It is important to make sure that the log file stays hidden completely because it showed up in the modified files list, the malware was identifiable during the investigation.
2. It is important to ensure that the malware created does not consume much resources which makes it easily identifiable.
3. Experience with new tools like Process Hacker, TCPView, etc. that are used during the investigation phase.

Current Status & Future Work

Current Status

1. Implemented an auto-executable key-logger using the Windows hooking mechanism.
2. The key codes of the keystrokes are stored in a log file.
3. Text file is stored in an unknown location, hidden from the user.

Future Work

Design an email extension to the existing model to send the log file to the attacker.

References

1. G. Yang, H. Gao, and W. Kang. Theory, implementation and protection of keylogger analysis the method of keylogger using a live example. In International Conference on Measurement and Control Engineering 2nd (ICMCE 2011). ASME Press, 2011.
2. E. Ladakis, L. Koromilas, G. Vasiliadis, M. Polychronakis, and S. Ioannidis. You can type, but you can’t hide: A stealthy gpu-based keylogger. In Proceedings of the 6th European Workshop on System Security (EuroSec), 2013.
3. A. Howard and Y. Hu. An approach for detecting malicious keyloggers. In Proceedings of the 2012 Information Security Curriculum Development Conference, pages 53-56. ACM, 2012.

Acknowledgments

This work was guided by Professor Rajendra K. Raj.

For more information Contact

Dhivya Govindarajan : dhivya@mail.rit.edu
Ashish Pandit : asp3948@rit.edu
Sushant Murdeshwar : spm5218@rit.edu

www.PosterPresentations.com