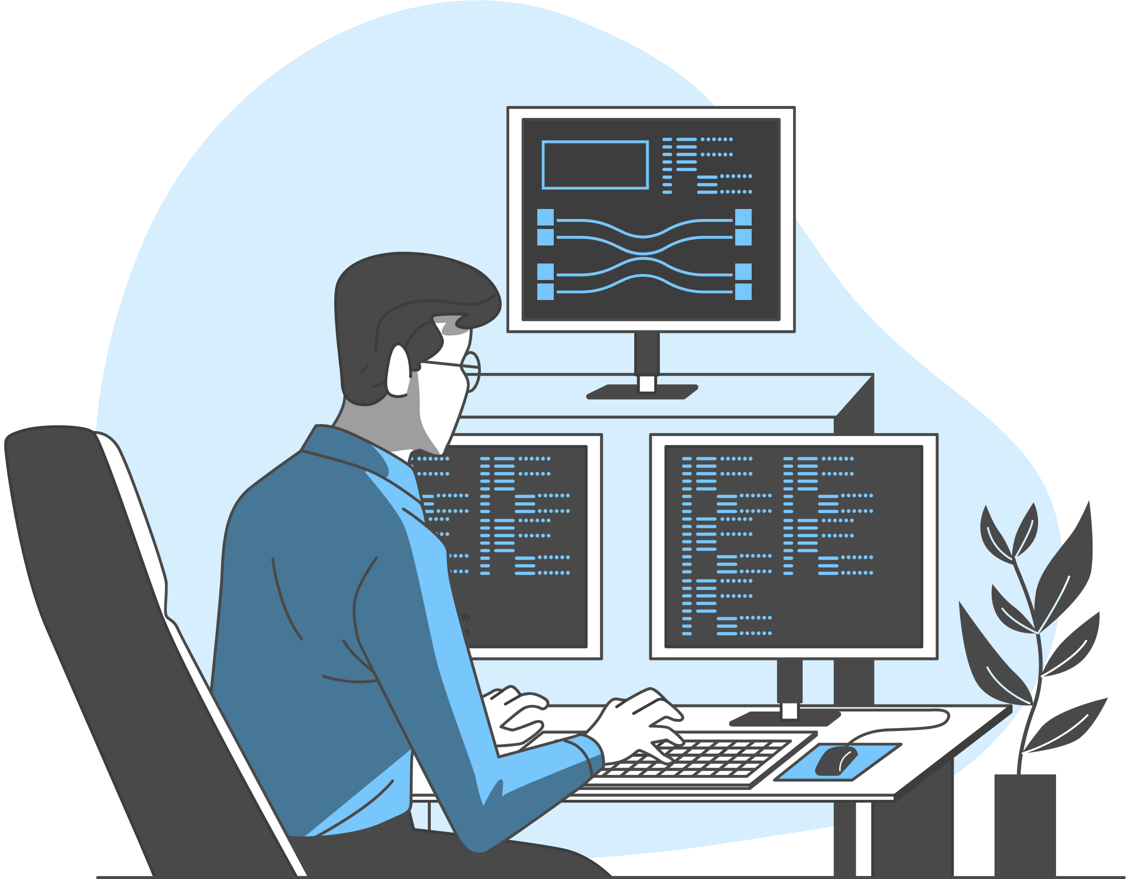
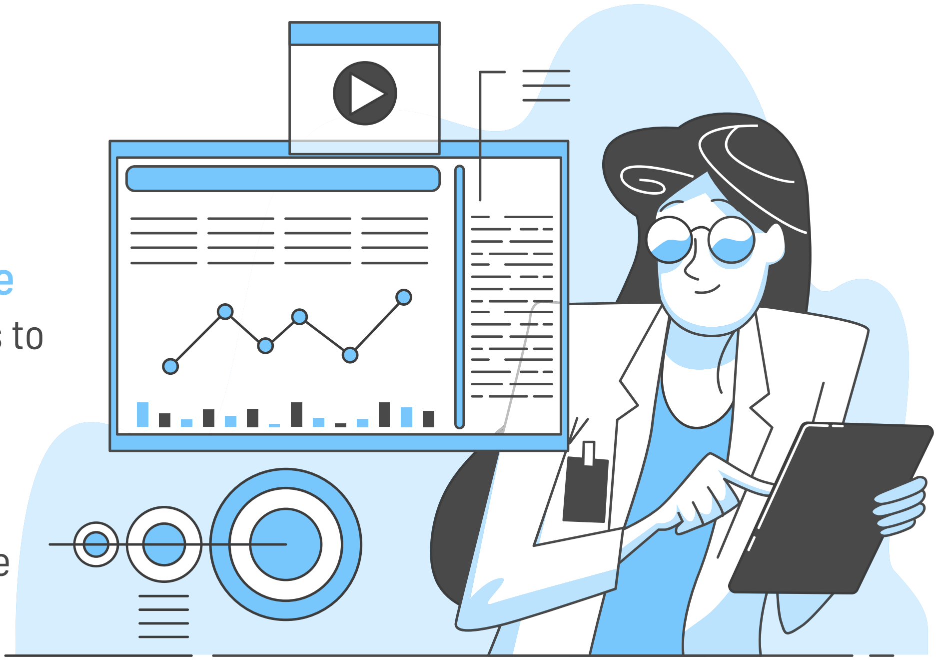


Panorama

Capturing System-wide Information Flow for Malware Detection and Analysis

Emily Miller, Tyler Prehl, Nick Hines

Table of Contents



01

Introduction

What is Panorama?

What it protects against

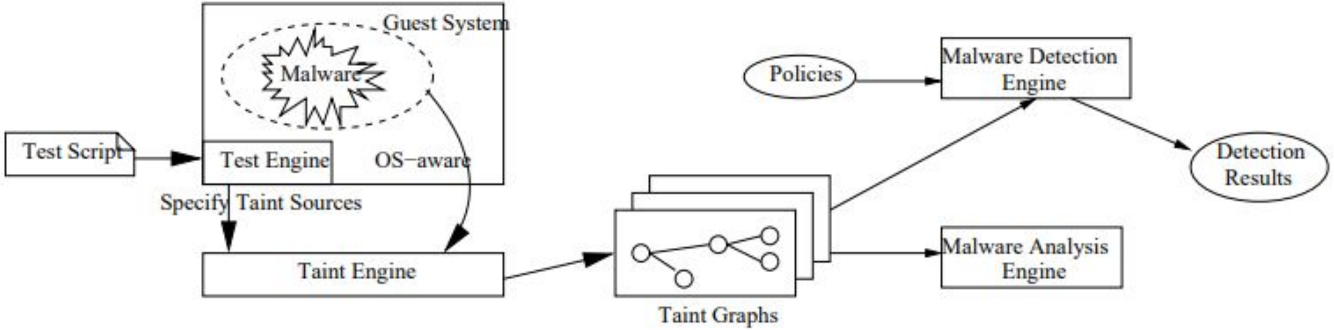
- Privacy and security leaks
- Programs from reputable vendors aren’t even safe
- A novel approach to combat keyloggers, spyware, rootkits, network sniffers, and stealth backdoors

What is Panorama and how it helps

- Malware detection prototype
- Successful with small drawback
- Efficiency over optimization

- The type of malware it detects
- What Panorama is and how it works
- Why it is considered a success

Test, Monitor, Analyze



...

02
Design and Implementation

How does it work?

Hardware-Level Dynamic Taint Tracking

- Shadow Memory
- Taint Sources
- Taint Propagation
    - Constant Function
    - Table Lookup
    - Control Flow Evasion

OS-Aware Taint Tracking

Resolving Process and Module Information

- Need: To know origins of instructions
- How: Maintain mapping of addresses between address in memory of modules via guest OS
- Solution: Module Notifier

Resolving Filesystem and Network Information

- Need: To know about data exchange between memory and hardware
- How: Properly map tainted data and its usage
- Solution: “The Sleuth Kit”

Identifying the Code under Analysis and Its Actions

- Need: To know when the malware accesses tainted data *indirectly*
- Case 1: Malware dynamically generates new code
- Case 2: Malware calls trusted code to perform tainted operations on its behalf

Automated Testing & Taint Graph Generation

Automated Testing

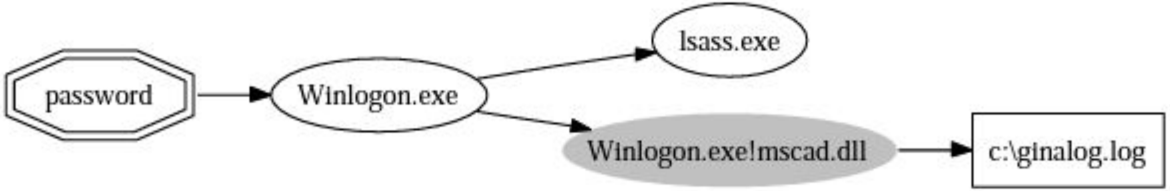
- Exactly what it sounds like
  - AutoHotkey scripts
- 8 Input Types:
  - Text
  - Password
  - HTTP
  - HTTPS
  - ICMP
  - FTP
  - Document
  - Directory

Taint Graph Generation

- A keystroke -> process A -> file -> process B -> ...
- g = (V,E)
  - g - taint graph
  - V - set of vertices
  - E - set of edges connecting the vertices

```
type ::= taint_source | os_object
taint_source ::= text | password | HTTP | HTTPS| FTP
      | ICMP | document | directory
os_object ::= process | module | network | file
```

A taint graph of GINA Spy



03

Taint Graphs and Malware

How can we use taint graphs to detect malware?

Anomalous Information Access Behavior

- Definition - a simple access performed by the samples under analysis
  - Why would this be malicious?
- Keyloggers and password thieves
  - Accessing text or password data
- Network sniffers and stealth backdoors
  - Accessing ICMP/TCP/UDP network inputs

Leakage and Excessive Access

Anomalous Info Leakage Behavior

- Definition - leaking information to third parties
    - Spyware/Adware vs Browser Helper Objects
- Considered "malicious activity" when private information is accessed and sent/saved elsewhere

Excessive Info Access Behavior

- Definition - excessively accessing the same information
    - Ex - rootkits and directory information
- Benign vs Malicious samples

Policies

$$\forall g \in G, (\exists v \in g.V, v.type = \texttt{module}) \wedge$$
$$g.root.type \in \{\texttt{text}, \texttt{password}, \texttt{FTP}, \texttt{UDP}, \texttt{ICMP}\}$$
$$\rightarrow Violate(v, \text{"}\texttt{No Access}\text{"})$$

$$\exists g \in G, (\exists v \in g.V, v.type = \texttt{module}) \wedge$$
$$(g.root.type \in \{\texttt{URL}, \texttt{HTTP}, \texttt{HTTPS}, \texttt{document}\}) \wedge$$
$$(\exists u \in descendants(v), u.type \in \{\texttt{file}, \texttt{network}\})$$
$$\rightarrow Violate(v, \text{"}\texttt{No Leakage!}\text{"});$$

$$(\forall g \in G,\ \ g.root.type = \texttt{directory} \rightarrow$$
$$\exists v \in g.V, v.type = \texttt{module})$$
$$\rightarrow Violate(v, \text{"}\texttt{No Excessive Access}\text{"})$$

Taint-Graph-Based Malware Analysis

Step 1

Check graph for a node corresponding to sample

Step 2

Obtain information that the sample has accessed

Step 3

Examine sample's successor nodes based on policies

04

Evaluation

Is Panorama a good resource to detect malware?

Experimental Evaluation

01 Taint-Graph-Based Malware Detection Approach

Used a large amount of real-world malware and benign samples

02 Google Desktop Case Study

Explored the amount of detailed information that could be extracted from the taint graph of an unknown sample

03 Tests Evaluating the Performance Overhead of Prototype

In experiments, Panorama was ran on a Linux machine

Malware Detection

- Malware collection
  - 42 real-world malware samples
    - 5 keyloggers, 2 password thieves, 2 network sniffers, 3 stealth backdoors, 22 spyware BHOs, and 8 rootkits
  - 56 benign samples
- Panorama correctly identified all malware samples but falsely declared three benign samples to be malicious
- Two of the false positives were personal firewall programs and the third was a browser accelerator
- The taint graphs of the false positives showed that the behaviors were similar to malware
- The reason for the false positives is that the taint-graph-based detection can only identify information access and process behavior of a given sample but not its intent

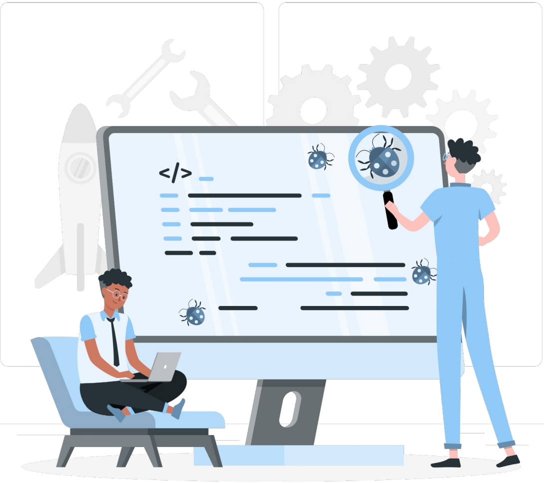
Category	Total	FNs	FPs
Keyloggers	5	0	-
Password thieves	2	0	-
Network sniffers	2	0	-
Stealth backdoors	3	0	-
Spyware/adware	22	0	-
Rootkits	8	0	-
Browser plugins	16	-	1
Multi-media	9	-	0
Security	10	-	2
System utilities	9	-	0
Office productivity	4	-	0
Games	4	-	0
Others	4	-	0
Sum	98	0	3

Malware Analysis

- Google Desktop Case Study
- Privacy policy states that it indexes and stores data files, mail, chat logs, and web history of a user
- If the special configuration setting “Search Across Computers” is enabled then Google Desktop will securely transmit copies of the index files to Google servers
    - This is malware type behavior since some index files may contain sensitive information
- Steps to conduct analysis
    1. Download installation file - 18 executables and shared libraries were installed
    2. Ran test cases using default settings and observed some components accessed tainted inputs (HTTPS, HTTP)
    3. Changed settings and enabled “Search Across Computers” and ran the test cases again
- Taint graph showed that Google Desktop does send sensitive information if a special feature is activated

Performance Overhead

- Measured Panorama’s performance overhead using utilities in Cygwin and while running tainted files and network inputs
- Found that Panorama is unoptimized and has a slowdown of 20 times on average
- Panorama needs to be optimized in order to be effective in real time malware analysis



Potential Evasion Techniques by Malware Writers

Breaking the propagation of taint information

Not behaving maliciously when tested

Subverting Panorama

Breaking the Propagation of Taint Information

The Attack

- Malware creators attempt to design their code in a way that makes the taint engine fail so the taint engine cannot properly keep track of tainted information
- By exploiting indirect dependencies a malicious program could conceal that sensitive information was leaked

The Countermeasure

- Enhance the implementation to keep track of taint propagation through control flow in the future
- If any sensitive data is accessed without authorization it is enough to classify as malware

Not Behaving Maliciously When Tested

The Attack

- Malware can evade detection by not exhibiting malicious behavior at the time the test cases are conducted
- Time bombs can activate on specific dates and some keyloggers only record keystrokes for certain applications or windows.
- Malware can also detect if it is running in the QEMU environment and can remain dormant if it is.

The Countermeasure

- Current Panorama prototype does not detect this type of malware
- Complementary work uses QEMU to build malware analysis system to uncover hidden behavior of malware through exploring execution paths

Subverting Panorama

The Attack

- There is a possibility to interfere with Panorama and the host system through exploiting buffer overflows and integer bugs

The Countermeasure

- This can be solved by fixing these bugs
- Panorama provides strong isolation and it is unlikely for malware to interfere

Security

Related Approaches Compared to Panorama

Malware Detection Approaches

Signature based malware detection is limited by its inability to detect previous unseen malware by cross-view based technique can only identify a list of hidden entries. Panorama recognizes the rootkit directly.

Dynamic Taint Analysis

Many systems detect exploits through tracking data from untrusted sources. Whole-system dynamic taint analysis analyzes how sensitive data are handled in OS. Panorama was developed with OS-aware analysis.

Information Flow Analysis

Previous work performed forensic analysis based on information flow. These systems only monitor the system call interface and are not as comprehensive and are not as precise. It is also difficult to track data while it is processed by a program.

Final Thoughts

- Current techniques for malware detection are ineffective.
- Whole-system fine-grained taint analysis captures the characteristics of a wide-range of malware
    - Keyloggers
    - Password Sniffers
    - Packet Sniffers
    - Stealth Backdoors
    - BHO-based spyware
    - Rootkits
- To evaluate effectiveness of this approach Panorama was designed and created
    - Evaluated 42 malware samples and 56 benign samples
    - Panorama had no false negatives and few false positives
- Panorama has the ability to offer valuable assistance to malware analysts to allow them to quickly identify malicious behavior

References

Yin, H., Song, D., Egele, M., Kruegle, C., & Kirda, E. (2007). Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, 1-12.

Thanks!

Do you have any questions?