Monitoring User Actions for Better Malware Specifications

Peter Chapman
University of Virginia
pmc8p@virginia.edu

Jeffrey Shirley
University of Virginia
jshirley@cs.virginia.edu

Objective: Harness User Intuition

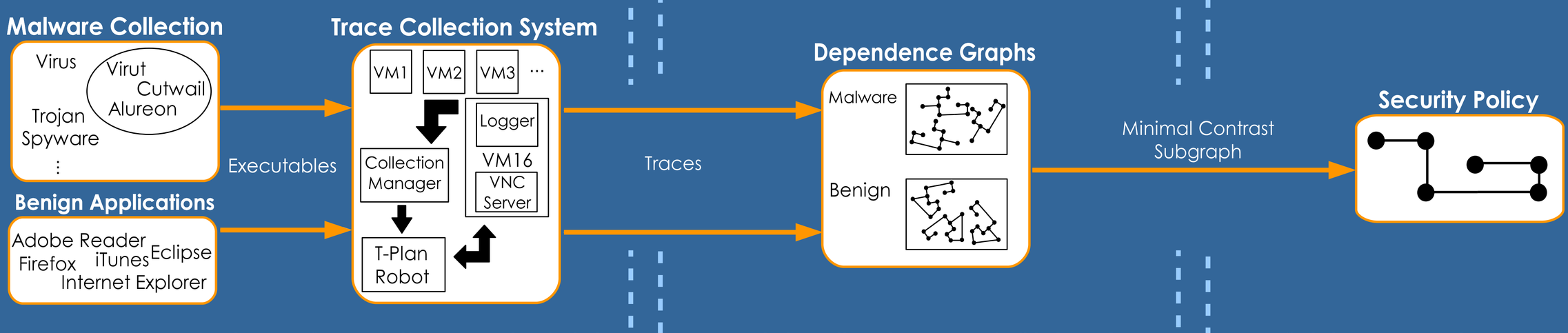
We propose incorporating user actions to improve the precision of malware specifications and introduce a system to create effective application security policies based on the relationships between user interaction, GUI events, and run-time operations of both benign and malicious applications.

Graphical malware such as Trojan:Win32/Fakeinit prevent us from simply allowing all user initiated actions.

Malware often modifies system folders and registry entries, but benign applications generally only perform such actions in conjunction with a graphical installer combined with user input

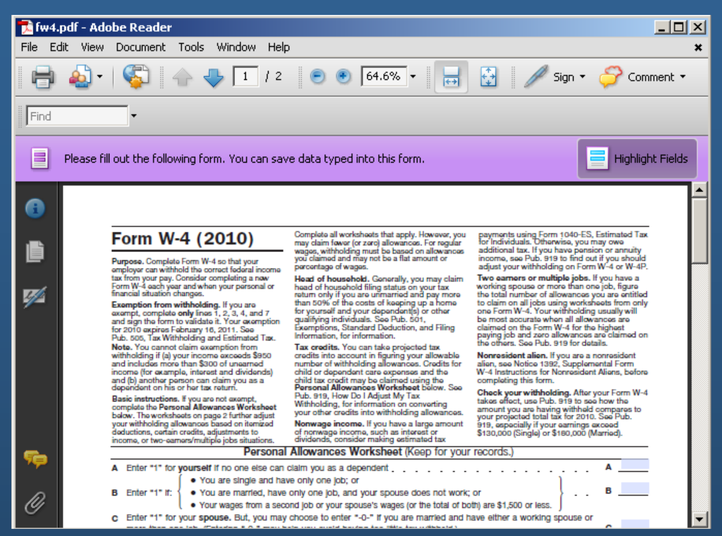
In other situations, a trusted application can be hijacked to perform undesired operations, but the relationship between the GUI and the underlying actions signals suspicious behavior.

For example, PDFs can exploit Adobe Reader using a JavaScript buffer-overflow (CVE-2009-0927) to gain access to the target system. Adobe Reader can then conduct malicious activities never associated with viewing PDFs, such as registering new system services, downloading and executing arbitrary files, and deleting unassociated registry entries.



We used a representative dataset of 3000 malicious samples including a variety of viruses, spyware, adware, worms, and exploits (e.g., malformed PDF and HTML files). To collect execution traces, we built a custom framework that runs samples in a virtual machine (VM), logging all events of interest to the host.

We scripted automated testing of interactive programs using the Java-based open source T-Plan Robot software and simulated typical use cases of popular and representative applications to gather benign data.



A screenshot from the automated Adobe Reader benign test.

The tracing software was implemented using Microsoft Detours, a library for intercepting Win32 API calls. Detours allows our tool to log user-level function calls (file, network, GUI, and user interaction) on each process running in the VM. Tracking the direct interaction of samples with the operating system circumvents complications brought by code obfuscation techniques.

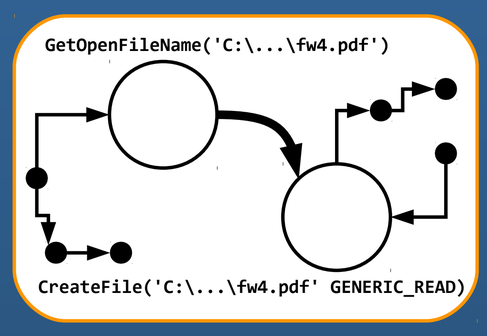
```
AcroRd32.exe WM_KEYUP caption='fw4.pdf'

AcroRd32.exe GetOpenFileName('C:\...\fw4.pdf')

AcroRd32.exe CreateFile('C:\...\fw4.pdf' 'GENERIC_READ')
```

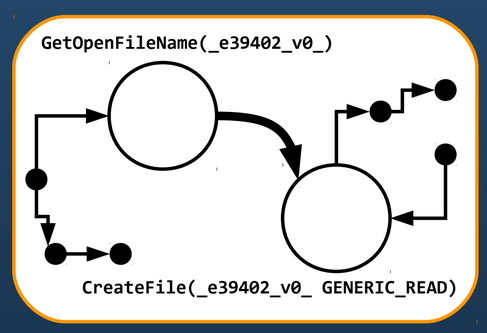
A sample from the Adobe Reader trace illustrating the connection between the file open dialog and the creation of a file handle.

We used the collected traces to build dependence graphs for the malicious and benign applications. Each node in the graph is a system call recorded in the execution trace, and edges are constructed between nodes that are data dependent (i.e., the output of a system call depends on a previous call) or occur within a specified time period.



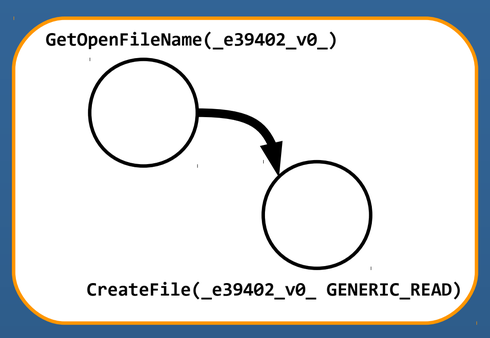
An example subgraph collected from Adobe Reader containing the relation between the file chooser dialog and read permission.

After creating the dependency graphs we generalize them in order to create security policies that are not limited to the specific applications tested. When a behavior flow is checked against the generated security policies, the abstract variables are bound to concrete ones.



An abstracted version of the above subgraph.

We find the contrast subgraphs for information flows present in the benign applications but not in the malware.



A generated security policy. When the user selects a file through the file chooser dialog, the program is able to read that file.

A sequence of events that correspond to an information flow in the security policy can be identified as benign. Sequences not found in the graph are to be treated with suspicion. For example, we hope to identify flows encapsulating the user interaction and GUI events related to creating a system file-chooser dialog followed by the creation of a file handle. Such a sequence is evident of user intentions.

Preliminary Results

Application	False Positives	False Negatives
MS Paint	1.52%	0%
GIMP	2.86%	0%
Internet Explorer	37.30%	0%
Thunderbird	25%	0%

Currently we are focusing on file accesses, so programs using the network have a disproportionately higher false positive rate.

This work is supported by the National Science Foundation through awards CCF-051123 and CNS-0627527.