Towards Secure User-space Provenance Capture

Nikilesh Balakrishnan, Thomas Bytheway, Lucian Carata, Ripduman Sohan, and Andy Hopper
University of Cambridge
OPUS

Observed Provenance in User Space

Data Scientists

Low Intrusion
User Space Provenance

Low Privilege Requirement

Easier Install Path

Semantically Closer
User Space Provenance Techniques

LD_PRELOAD

Binary Rewriting

ptrace

FUSE
How can we rely on user-space provenance?
Assumptions

Malicious User

Malicious Application

Trusted Kernel/Hardware
5 Attack Classes

**Circumvention**
- Direct Library Call
- Direct Syscall
- Denial of Service

**Falsification**
- Man in the Middle
- Time of Check Time of Use
5 Attack Classes

Circumvention
- Direct Library Call
- Direct Syscall
- Denial of Service

Falsification
- Man in the Middle
- Time of Check Time of Use
```python
def open(file):
    log("open", file)
    return real_open(file)
```

```c
/* */
syscall
/* */
```
def open(file):
    log("open", file)
    return real_open(file)

def open(file):
    /* */
    syscall
    /* */

open("foo.txt")
func_ptr real_open = 0x45211f
*real_open("foo.txt")
def open(file):
    log("open", file)
    return real_open(file)
Man In The Middle

Application

open("results.txt")

Provenance Library

def open(file):
    log("open", file)
    return real_open(file)

Malicious User Library

def open(file):
    return real_open("fake_results.txt")

C Library

def open(file):
    /* */
    syscall /* */
Denial of Service

User Application Resources

Provenance System Resources

File Descriptors
Memory
Our Approach

Intel Secure Guard Extensions (SGX)

Existing Sandboxing Techniques

Dynamic Binary Rewriting
Conclusions

System-level provenance is not always preferable to user-space provenance

User-space provenance suffers from some threats

It can still be made secure
Thank you

Any Questions?

For more Info:

http://www.cl.cam.ac.uk/research/dtg/fresco/