PRECIP: Towards Practical and Retrofittable Confidential Information Protection

XiaoFeng Wang (IUB), Zhuowei Li (IUB), Ninghui Li (Purdue) and Jong Youl Choi (IUB)
How to protect your information from spyware?

However…

Prevent it!

Detect it!

However…
The last defense line

- Contain unauthorized surveillance
Spyware containment

- Existing access control mechanisms are insufficient
  - Spyware can watch *authorized party’s* access to a secret

- Alternative: information flow security
  - Track sensitive data
  - Prevent them from flowing into unauthorized parties
Information flow security

- The Bell-LaPadula model
However, this is insufficient for a modern OS

- User input object
  - keyboard, mouse…
  - When does it become sensitive?

- Other shared object
  - screen, clipboard …
  - sensitive? public?

- Multitasked subject
  - Work concurrently on public and sensitive data
  - Which output is sensitive?
Requirements for a usable IF model

- Work on a modern OS

- Efficient enough for online operation
  - Instruction-level tracking can be too slow

- Retrofittable to legacy systems
  - Avoid modifying the source code of app, of OS
A first step towards practical and retrofittable confidential information protection

- Track an application’s input/output dependence
- Model input object and shared object
- Designed for online operations
- Retrofittable to legacy applications and OS
The model

- **Subjects and objects**
  - Local objects (files, buffers, keyboard, screen, …)
  - Remote objects (website…)
  - User input objects (UIO): objects for transferring inputs (keyboard)

- **Channels**
  - Connect subject to subject, subject to object, object to subject
  - A path is composed of multiple channels

- **Messages**
  - Information on a channel in the form of “messages”
  - Examples: keyboard events, mouse events, data through a “read” call
The model (cont’d)

- **Dependency relation**
  - Output messages depend on some input messages
  - An input to the PRECIP model

- **Sensitivity levels**
  - high: “sensitive”, low: “public”

- **Trusted and untrusted subjects**
  - Untrusted: unknown dependency relations
  - Trusted: all dependency relations are known
Security objective

- Information is sensitive if
  - it depends (directly or transitively) upon a message from an sensitive object, or sensitive inputs from an UIO

- Information leakage happens if
  - Sensitive info gets into an untrusted subject or a remote public object

- Objective: Sensitive information shouldn’t be leaked
Policies achieving the objective

- **Tracing rules**
  - Sensitive msg: either from a sensitive obj or dependent upon a sensitive msg
  - Obj ⇒ sensitive if it receives a sensitive msg
  - UIO ⇒ sensitive iff a path connects it to a sensitive obj
  - Obj ⇒ public if it is cleaned

- **Control rules**
  - Block sensitive msg to public remote obj and untrusted sub
  - Sensitive info to a local obj ⇒ block the msg or mark the obj sensitive
Application of PRECIP to Windows XP

PRECIP system

Integrity Protector

Browser Add-on

Tracer

depend. rules

Classifier

subject: processes

objects: files, memory, screen, keyboard, windows etc

query

Controller

managmt of untrusted hooks
dissemina. control

control rules

user land

kernel land

label/query
Adversary model

- Spyware is not inside the kernel when PRECIP is installed
  - However, our integrity protector can prevent spyware to be installed through system calls

- PRECIP is not designed for preventing exploit of software vulnerabilities
  - We use existing tools to do the job
Classification and labeling

- Trust levels
  - Classify applications according to dependency rules
  - Mark an executable using its NTFS file stream

- Sensitivity levels
  - Automatic classification: using a file’s DAC
Dependency rules for editing/viewing App

100 Mil
There is a place with it!!! Don't tell other people!
Here is the map

Menu of the Best Chinese Restaurant in the town
Blar blar blar……

Sensitive
Public
Sensitive
Public
Sensitive
Public
Sensitive
Public
Dependency rules for web browsers
Management of hooks

Original Message-handling
- Untrusted Hook
- Trusted Hook
- App

Message-handling in PRECIP
- Proxy Hook
- Access Control
- Trusted Hook
- Sensitive msg
- Public msg
- App
Integrity protection

- Prevent unauthorized access of subject’s and object’s labels, contents and PRECIP settings
  - Regulate calls related to file system, auto-start extensibility points and process

- Only allow signed kernel drivers to be loaded
  - A policy also used in Windows Vista
Evaluation

- Dependency rules
  - Test dependency rules on Microsoft Office, Adobe Acrobat and Notepad
  - Quite effective in most cases

- Effectiveness

- Performance
## Effectiveness

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Control Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home KeyLogger [8]</td>
<td>Key Logger</td>
<td>bypass the hook host.</td>
</tr>
<tr>
<td>RunHook [19]</td>
<td>Key Logger</td>
<td>bypass the hook host.</td>
</tr>
<tr>
<td>Synthesized-1 [27]</td>
<td>Key Logger</td>
<td>block two system calls: NtUserGetKeyboardState and NtUserGetKeyState.</td>
</tr>
<tr>
<td>Synthesized-2 [34]</td>
<td>Key Logger</td>
<td>block one system call: NtUserGetAsyncKeyState.</td>
</tr>
<tr>
<td>GhostlyEye [7]</td>
<td>Screen Grabber</td>
<td>block one system call: NtGDISTretchBlt</td>
</tr>
<tr>
<td>Any Capture [4]</td>
<td>Screen Grabber</td>
<td>block two system calls: NtGDISTretchBlt and NtGDIBitBlt</td>
</tr>
<tr>
<td>Sub7 [15]</td>
<td>File Stealer</td>
<td>untrusted process does not allow to open sensitive files.</td>
</tr>
<tr>
<td>Cerberus [5]</td>
<td>Lightweight ftpd</td>
<td>untrusted process does not allow to open sensitive files.</td>
</tr>
</tbody>
</table>
Performance

- Performance of hook management
  - Baseline (no proxy): 691.015 microseconds
  - PRECIP: 784.809 microseconds
  - Overhead: 13.57%

- Performance of the kernel driver
  - Evaluated using WorldBench 5.0

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Baseline</th>
<th>with PRECIP</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office XP SP2</td>
<td>784 s</td>
<td>838 s</td>
<td>6.89%</td>
</tr>
<tr>
<td>Photoshop 7.0.1</td>
<td>647 s</td>
<td>675 s</td>
<td>4.33%</td>
</tr>
<tr>
<td>Mozilla 1.4</td>
<td>1122 s</td>
<td>1265 s</td>
<td>12.75%</td>
</tr>
</tbody>
</table>

*Table 4. Overhead of the Kernel Driver.*
Limitations

- Dependency rules are empirical
  - Research: automatic analysis of an application to generate rules

- Integrity model as a complementary

- Model is incomplete
  - Multiple sensitivity levels
  - Compartmentalization
Related research

- Language-based information flow security
  - For design of a new program

- Instruction-level tracking
  - Hard to use online without hardware support

- New systems such as Abestos, IX, Flume,...
  - Need to modify OS

- Sandboxing techniques
  - Too coarse-grained
Conclusions

- Propose a new confidentiality model for practical and retrofittable IF protection

- Application of the model to Windows XP

- Future research
  - Improve the model
  - Improve the techniques for enforcing the model