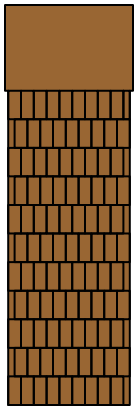
PRECIP: Towards Practical and Retrofittable Confidential Information Protection

XiaoFeng Wang (IUB), Zhuowei Li (IUB), Ninghui Li (Purdue) and Jong Youl Choi (IUB)

How to protect your information from spyware?

However…

Prevent it !

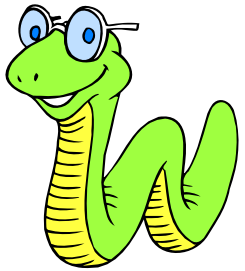


Detect it !

However…

The last defense line

- Contain unauthorized surveillance

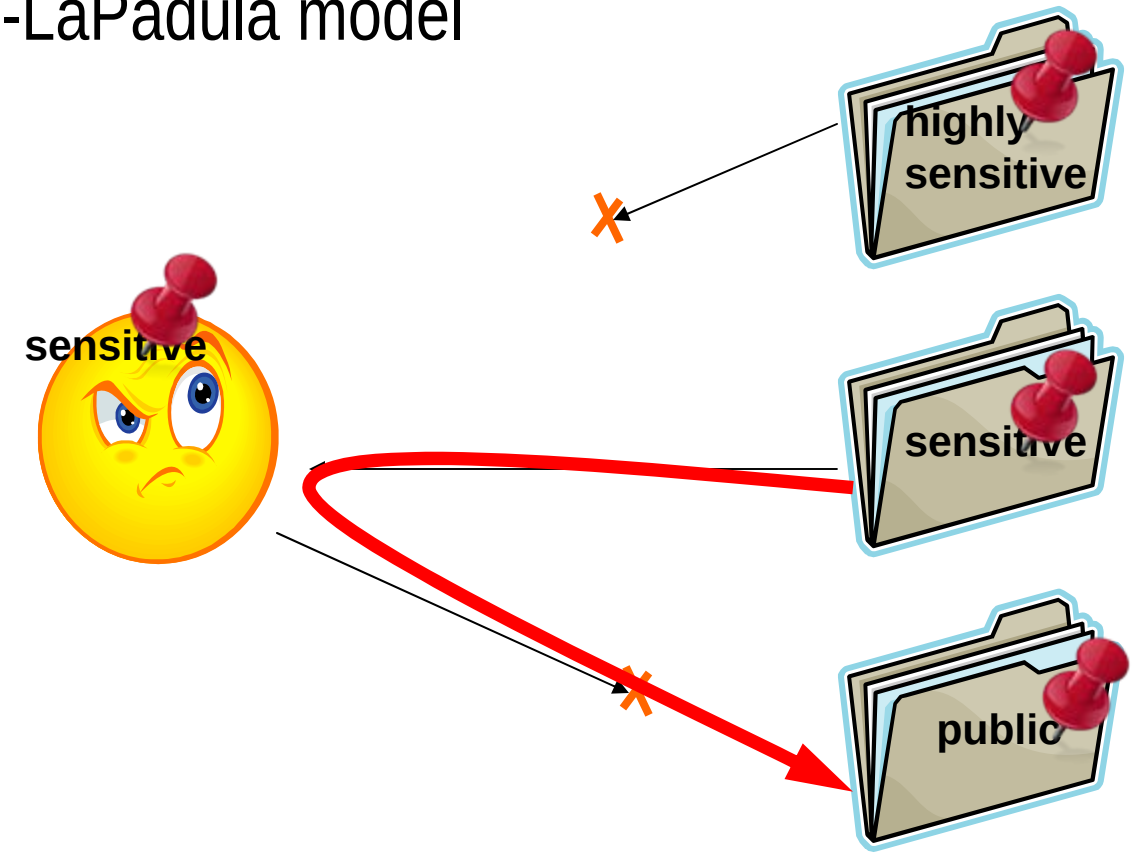


Spyware containment

- 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

PRECIP

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

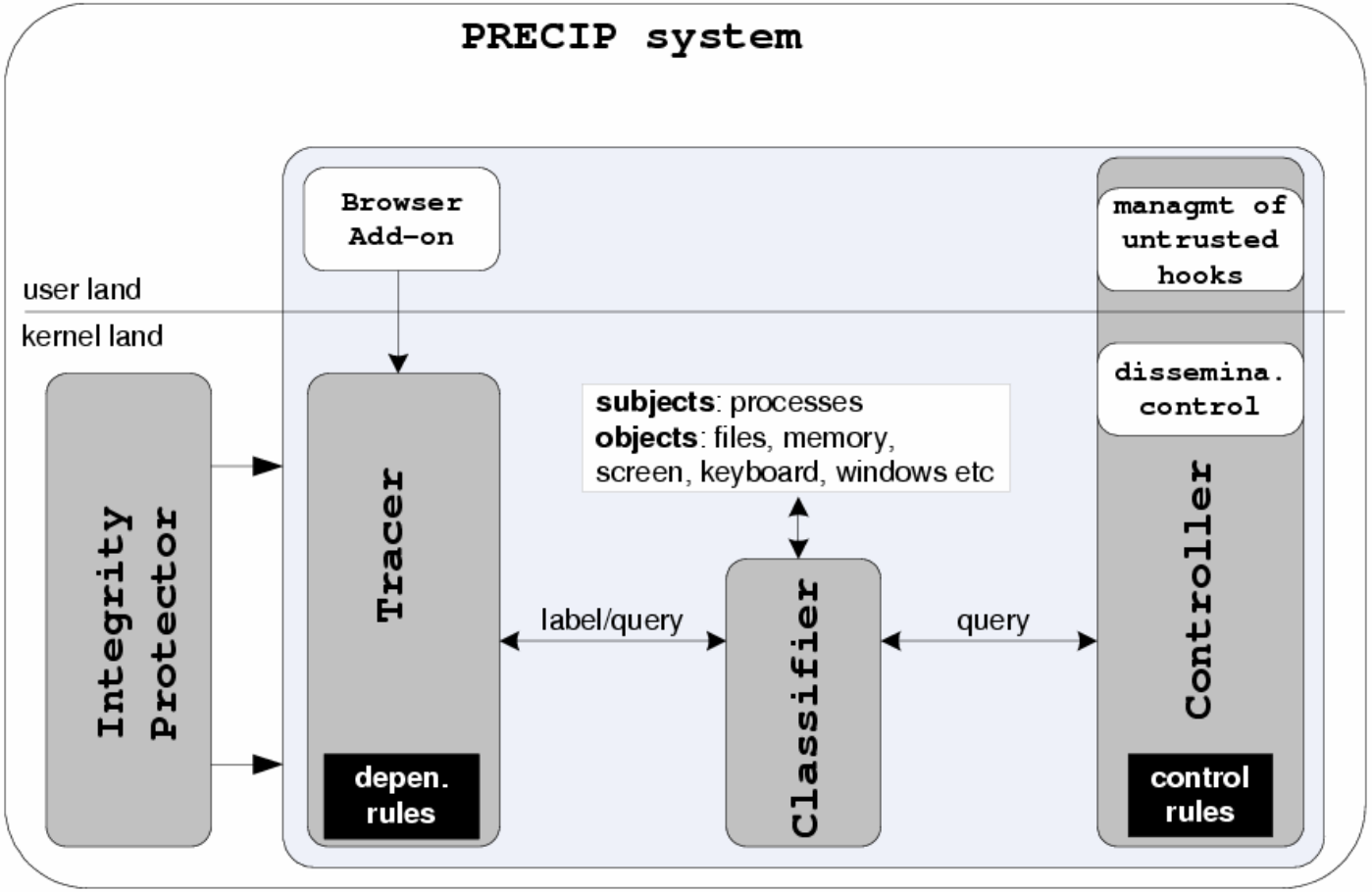
Security objective

- 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 $\Rightarrow$ sensitive if it receives a sensitive msg
  - UIO $\Rightarrow$ sensitive iff a path connects it to a sensitive obj
  - Obj $\Rightarrow$ public if it is cleaned
- Control rules
  - Block sensitive msg to public remote obj and untrusted sub
  - Sensitive info to a local obj $\Rightarrow$ block the msg or mark the obj sensitive

Application of PRECIP to Windows XP



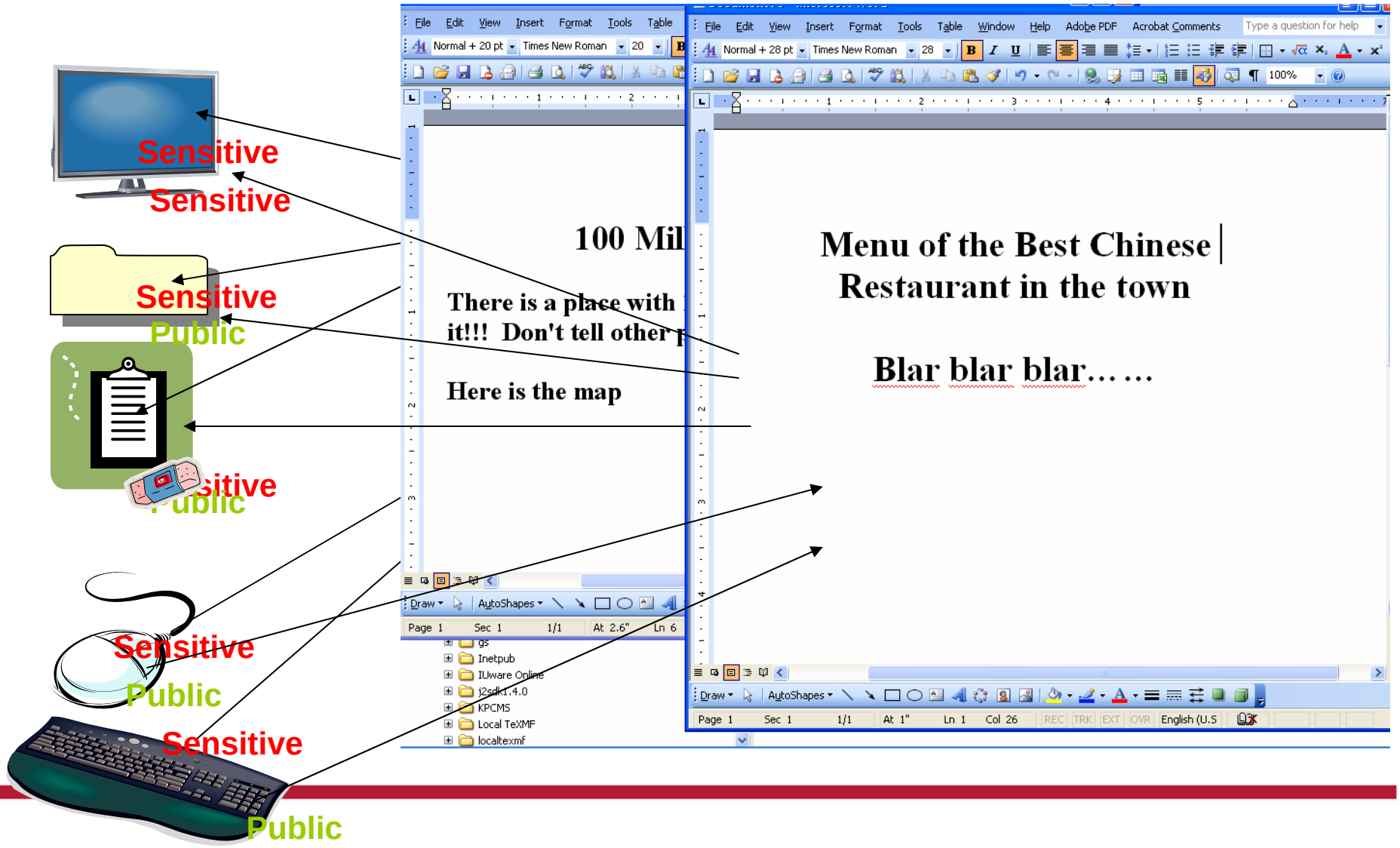
Adversary model

- Spyware is not inside the kernel when PRECIP is installed
  - However, our integrity protector can preventspyware to be installed through system calls
- PRECIP is not designed for preventing exploit of software vulnerabilities
  - We use existing tools to do the job

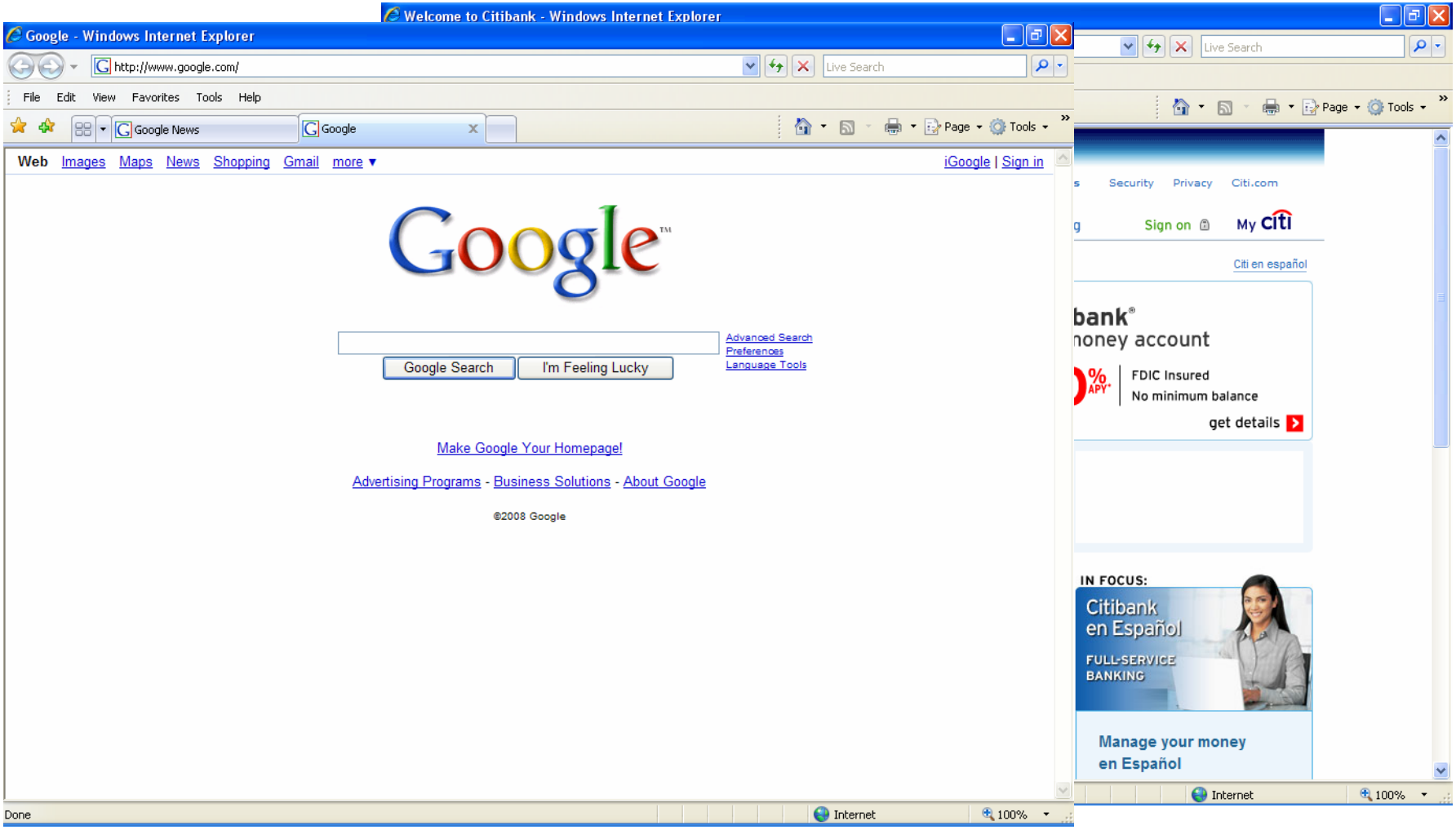
Classification and labeling

- 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

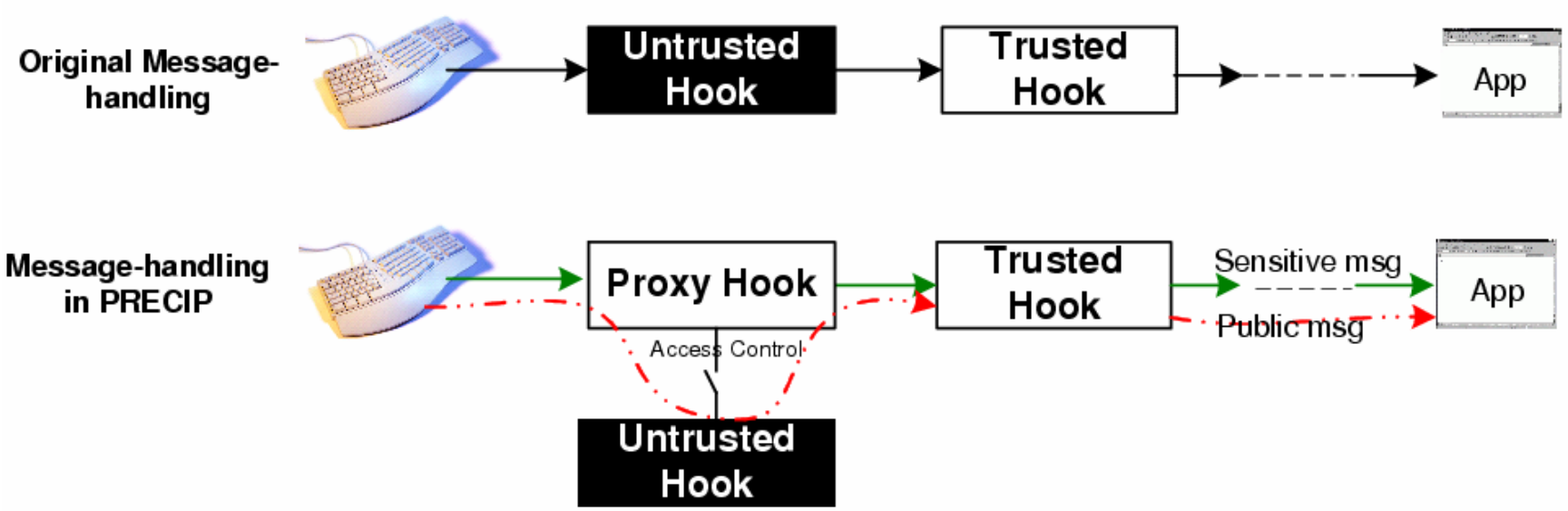
Dependency rules for editing/viewing App



Dependency rules for web browsers



Management of hooks



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

	Name	Type	Control Actions
1	KidLogger [50]	Key Logger	bypass the hook host.
2	Home KeyLogger [8]	Key Logger	bypass the hook host.
3	RunHook [19]	Key Logger	bypass the hook host.
4	Synthesized-1[27]	Key Logger	block two system calls: `NtUserGetKeyboardState` and `NtUserGetKeyState`.
5	Synthesized-2[34]	Key Logger	block one system call: `NtUserGetAsyncKeyState`.
6	GhostlyEye[7]	Screen Grabber	block one system call: `NtGDIStretchBlt`
7	Any Capture[4]	Screen Grabber	block two system calls: `NtGDIStretchBlt` and `NtGDIBitBlt`
8	Hidden Recorder[12]	Screen Grabber	block one system call: `NtGDIBitBlt`.
9	Sub7[15]	File Stealer	untrusted process does not allow to open sensitive files.
10	Cerberus[5]	Lightweight ftpd	untrusted process does not allow to open sensitive files.

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

Benchmark	Baseline	with PRECIP	Overhead
Office XP SP2	784 s	838 s	6.89%
Photoshop 7.0.1	647 s	675 s	4.33%
Mozilla 1.4	1122 s	1265 s	12.75%

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