Clean Application Compartmentalization with SOAAP

Khilan Gudka*, Robert N.M. Watson*, Jonathan Anderson†, David Chisnall*, Brooks Davis§, Ben Laurie¶, Ilias Marinos*, Peter G. Neumann§, Alex Richardson*

*University of Cambridge, †Memorial University, §SRI International, ¶Google UK Ltd

ACM CCS 2015
14 October 2015
Vulnerabilities galore...

Mitigate both known and unknown vulnerabilities

Heartbleed

Shellshock

[Insert next big vulnerability here]
Principle of least privilege

Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job.

Saltzer 1974 - CACM 17(7)
Saltzer and Schroeder 1975 - Proc. IEEE 63(9)
Needham 1972 - AFIPS 41(1)
Application Compartmentalization

Conventional “fetch”

Vulnerable code

Kernel

Compartmentalized “fetch”

Ambient process

Sandboxed process

Vulnerable code

Kernel
When a conventional application is compromised, its ambient rights are leaked to the attacker, e.g., full network and file system access.
Application Compartmentalization

When a compartmentalized application is compromised, only rights held by the exploited component leak to the attacker.

Most vulnerabilities will no longer yield significant rights, and attackers must exploit many vulnerabilities to meet their goals.
1. fetch
   main loop
   ftp
   http
tls
   network sandbox

2. fetch
   main loop
   ftp
   http
tls
   FTP sandbox
   http
   HTTPS sandbox

3. fetch
   main loop
   ftp
   http
tls
   FTP sandbox
   http
   HTTP sandbox
   tls
   TLS sandbox

4. fetch
   main loop
   ftp
   http
tls
   URL-specific sandbox
   http
   URL-specific sandbox
   ftp
   URL-specific sandbox
   tls
   URL-specific sandbox

Object-oriented compartmentalization
3. fetch

**main loop**

- ftp
  - FTP sandbox
- http
  - HTTP sandbox
- tls
  - TLS sandbox
Before, a single exploit anywhere in the program would have leaked ambient rights.
A compartmentalized version with FTP, HTTP and TLS in separate sandboxes.
“exploit FTP to access TLS state”

TLS sandbox hard to access from FTP sandbox. Requires 3 exploits!
“exploit TLS to gain elevated privilege”

Gaining elevated privilege requires 2 exploits!
Compartmentalization is hard!

• “local” program turned into a distributed one

• Preserving functional correctness

• Mapping security model to sandboxing substrate

• Different compartmentalization tradeoffs

• Hard to change and maintain
Onus is on the developer

“It’s up to you to understand these elements [of App Sandbox] and then to use your understanding to create a plan for adopting App Sandbox.”

- Apple App Sandbox Design Guide
Security-Oriented Analysis of Application Programs

Repeated refinement

Inputs
- Source code
- Source annotations
- Sandbox description

Toolchain
- Clang
- LLVM
  - Past vulnerability
  - Call graph
  - Information flow
  - App rights
  - Performance

Outputs
- Warnings & Suggestions

App author
Security developer
OS vendor

App author
Source code
Source annotations
Sandbox description

Toolchain
Clang
LLVM
Past vulnerability
Call graph
Information flow
App rights
Performance

Outputs
Warnings & Suggestions
Example: Confidentiality/IFC

```c
char* server_key __soaap_classify("secret");
extern void compute_session_key(char*, char*);

void main() {
    ...
    while (...) {
        accept_connection();
    }
}

__soaap_sandbox_persistent("session")
void accept_connection() {
    char session_key[256] __soaap_private;
    compute_session_key(session_key, server_key);
    ...
}
```
Example: Confidentiality/IFC

```c
char* server_key __soaap_classify("secret");
extern void compute_session_key(char*, char*);

void main() {
    ...
    while (...) {
        accept_connection();
    }
}

__soaap_sandbox_persistent("session")
void accept_connection() {
    char session_key[256] __soaap_private;
    compute_session_key(session_key, server_key);
    ...
}
```
Example: Past vulnerabilities/supply-chain trojans

```c
__soaap_provenance("some vendor")
__soaap_sandbox_ephemeral("parser")
void parse(__soaap_fd_permit(read) int ifd, DOMTree* out) {
  if (...) {
    __soaap_vuln_pt("CVE-2005-ABC");
    ...
  }
}
__soaap_vuln_fn("CVE-2005-DEF")
void not_sandboxed() {
  ...
}
```
Example: Past vulnerabilities/supply-chain trojans

```c
__soaap_provenance("some vendor")
__soaap_sandbox_ephemeral("parser")
void parse(__soaap_fd_permit(read) int ifd, DOMTree* out) {
  if (...) {
    __soaap_vuln_pt("CVE-2005-ABC");
    ...
  }
}
__soaap_vuln_fn("CVE-2005-DEF")
void not_sandboxed() {
  ...
}
```
Case studies

• Fetch – design-space exploration
• Okular – large-scale new compartmentalization
• OpenSSH – long-term maintenance
• Chromium – sandboxing effectiveness
OpenSSH

sshd

Monitor

Monitor

Incoming connection

fork unprivileged process

Preauth

Postauth

export state

fork user child

Timeline
OpenSSH

Percentage of annotations to LOC

% of annotations to LOC

OpenSSH Version, No. of annotations and LOC

| Version | Annotations | LOC |
|---------|-------------|-----|
| 3.2.2   | 192         | 43K |
| 3.5     | 211         | 45K |
| 3.7     | 247         | 49K |
| 5.3     | 287         | 64K |
| 6.0     | 298         | 74K |
| 6.5     | 308         | 83K |
| 6.8     | 300         | 97K |
OpenSSH

Percentage of privileged functions

| OpenSSH Version | No. of privileged funcs | LOC |
|-----------------|-------------------------|-----|
| 3.2.2           | 281                     | 43K |
| 3.5             | 283                     | 45K |
| 3.7             | 302                     | 49K |
| 5.3             | 549                     | 64K |
| 6.0             | 629                     | 74K |
| 6.5             | 829                     | 83K |
| 6.8             | 893                     | 97K |

% of privileged functions
OpenSSH

System call warnings

SOAAP identifies disallowed system calls in Capsicum

rhosts removed

SOAAP detects rhosts broken

Capsicum version released. Still broken system calls

Number of warnings

OpenSSH Version and LOC

3.2.2
43K

3.5
45K

3.7
49K

5.3
64K

6.0
74K

6.5
83K

6.8
97K
Chromium

Browser process

User Interface | Network | Storage

Renderer process

HTML Renderer
V8 Engine
DOM Bindings

Renderer process

HTML Renderer
V8 Engine
DOM Bindings
New Chromium vulnerability?

Data flow for network data

- CompleteRead
- OnReadCompleted
- Read
- SniffForOfficeDocs [Cr issue #239580]
- SniffMimeType
- DetermineMimeType
- ReplayReadCompleted
- ProcessResponse
- ChromeMain
- main
- GetPluginInfo
- HasSupportingHandler
- SelectNextHandler
**SOAAP tool performance**

SOAAP running times as proportion of compile+link time

| Program and size (LOC) | SOAAP running time (s) | LLVM running time (s) |
|------------------------|-------------------------|----------------------|
| Chromium (13.1M)       | 3048s                   | 7300s                |
| Okular (4.2M)          | 15.37s                  | 2970s                |
| OpenSSH (97K)          | 4.4s                    | 8.7s                 |
| fetch (5K)             | 1.17s                   | 2.4s                 |

Proportion

- **LLVM**
- **SOAAP**
Conclusion

• Application compartmentalization is important but hard to get right

• Conceptual framework gives structure

• SOAAP enables reasoning about tradeoffs

• SOAAP can scale to multi-million LoC programs

http://bit.ly/soaap
Example SOAAP output (1)

```
$ make soap

*** Sandboxed method "accept_connection" read data
*** value of class: [secret] but only has clearances for: []
+++ Line 28 of file session.c

*** Sandboxed method "accept_connection" executing in
*** sandboxes: [session] may leak private data through
*** the extern function "compute_session_key"
+++ Line 28 of file session.c

*** Persistent sandbox "session" contains private data that
*** may leak when the sandbox is reused. Consider using an
*** ephemeral sandbox instead or scrub the memory region
*** before control returns
```
Example SOAAP output (2)

$ make soap

*** Method "not_sandboxed" had past vulnerability
*** "CVE-2005-DEF" but is not sandboxed. Another
*** vulnerability here could leak ambient authority
*** to an attacker including full network and file
*** system access

*** Sandboxed method "parse" has a past-vulnerability
*** annotation for "CVE-2005-ABC". Another vulnerability
*** here would only leak the following:
+++ Read access to file descriptor "ifd"
## Chromium security disparity

|        | OS     | Sandbox      | LoC   | FS | IPC | NET | S≠S’ | Priv |
|--------|--------|--------------|-------|----|-----|-----|------|------|
| **DAC** | Windows| DAC ACLs     | 22,350| ⚠️ | ⚠️  | ✗   | ✗   | ✓    |
|        | Linux  | chroot()     | 600   | ✓  | ✗   | ✗   | ✓    | ✗    |
| **MAC** | Mac OS X| Sandbox      | 560   | ✓  | ⚠️  | ✓   | ✓    | ✓    |
|        | Linux  | SELinux      | 200   | ✓  | ⚠️  | ✓   | ✗    | ✗    |
| **Cap** | Linux  | seccomp      | 11,300| ⚠️ | ✓   | ✓   | ✓    | ✓    |
|        | FreeBSD| Capsicum     | 100   | ✓  | ✓   | ✓   | ✓    | ✓    |

Watson et al., “Capsicum: practical capabilities for UNIX”