last time (1)

CHALLENGE logistics

the system call interface is big:

- hard to enumerate needed system calls
- easy to miss features (e.g. runc bug) that need to be restricted

isolating programs that used shared services (e.g. windowing service)

- proxy? another system-call like interface?

last time (2)

mandatory access control, example: SELinux

- "type" labels for objects (files, etc.)
- explicit list of allowed operations
- enforcement in OS

separate views of system resources for sandboxes

- chroot: program views subset of filesystem
- mount namespace: independent view of available disks
  - "bind mounts" to expose directory 'outside' as virtual disk
- pid, network, etc. namespaces — container $\approx$ lightweight VM sharing OS

runc bug

2019 bug in Docker, other container implementations (CVE-2019-5736)

- blog post for vulnerability finders:
  `https://blog.dragonsector.pl/2019/02/cve-2019-5736-escape-from-docker-and.html`

bug setup:

- user starts malicious container X
- user tells docker to start a new command in malicious container X
- malicious container X hijacks the "new command" starting program
- hijacked program used to access stuff outside container

part of problem: Docker and others weren't using user namespaces at the time

- compatability problems

runc bug

2019 bug in Docker, other container implementations (CVE-2019-5736)

- blog post for vulnerability finders:
  `https://blog.dragonsector.pl/2019/02/cve-2019-5736-escape-from-docker-and.html`

bug setup:

- user starts malicious container X
- user tells docker to start a new command in malicious container X
- malicious container X hijacks the "new command" starting program
- hijacked program used to access stuff outside container

part of problem: Docker and others weren't using user namespaces at the time

- compatability problems

setup: /proc/PID

Linux provides /proc directory to access info about programs

used for implementing process list utils, debugging

- needed to make a functional container

subdirectory for each process in current container

- process ID PID has /proc/PID subdirectory
- /proc/self is alias for current process's subdirectory

included is /proc/PID/exe file — alias for executable file

running a command in existing container

to run command X in existing container:

step 1: switch current process to that container

step 2: execute command X

running a command in existing container

to run command X in existing container:

step 1: switch current process to that container

code in container can access /proc here?

including overwriting /proc/self/exe!

- which is a program run as root!

step 2: execute command X

partial fix

can disable access to /proc/PID/exe (and related things)

system call: `prctl(PR_SET_DUMPABLE, 0)`

but...the run-in-container tool did this for a while

partial fix

can disable access to /proc/PID/exe (and related things)

system call: `prctl(PR_SET_DUMPABLE, 0)`

but...the run-in-container tool did this for a while

problem: this gets reset on executing a new program

and attacker could make the new program be /proc/PID/exe

- one mechanism: symbolic links (file aliases)

but change dynamic linking setup to run attacker code

...which accesses /proc/self/exe

full fix

make single-use copy of start-in-container tool each time command run

- in-memory file

...so modifying it doesn't change anything

- (but it's also protected from modification)

other solutions:

- make executable non-writable (e.g. SELinux, don't run container as root)

SELinux escape

When executing a program via the SELinux sandbox, the nonpriv session can escape to the parent session by using the TIOCSTI ioctl to push characters into the terminal's input buffer, allowing an attacker to escape the sandbox.

```
$ cat test.c
#include <unistd.h>
#include <sys/ioctl.h>

int main()
{
     char *cmd = "id\n";
     while(*cmd)
      ioctl(0, TIOCSTI, cmd++);
     execlp("/bin/id", "id", NULL);
}

$ gcc test.c -o test
$ /bin/sandbox ./test
id
uid=1000 gid=1000 groups=1000
context=unconfined_u:unconfined_r:sandbox_t:s0:c47,c176
$ id    <------ did not type this
uid=1000(saken) gid=1000(saken) groups=1000(saken)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Android sandbox

Android — Linux based OS for phones/tablets

`https://source.android.com/security/app-sandbox`

current version: SELinux + seccomp (system call filter)

OS X sandboxing

OS X (tries to) implement system call filtering

main challenge: what about files?

- user can open a file anywhere — we expect that to work

OS X sandboxing

OS X (tries to) implement system call filtering

main challenge: what about files?

- user can open a file anywhere — we expect that to work

OS X solution: OS service displays file-open dialog

- OS knows user really choose a file

application can ask to remember file was chosen previously

not chosen/remembered — can't access

- requires changes to how applications open files

another sandboxing OS: Qubes

Qubes: heavily sandboxed OS

runs seperate VMs instead of filtering syscalls

UI that clearly shows what VM each window is from

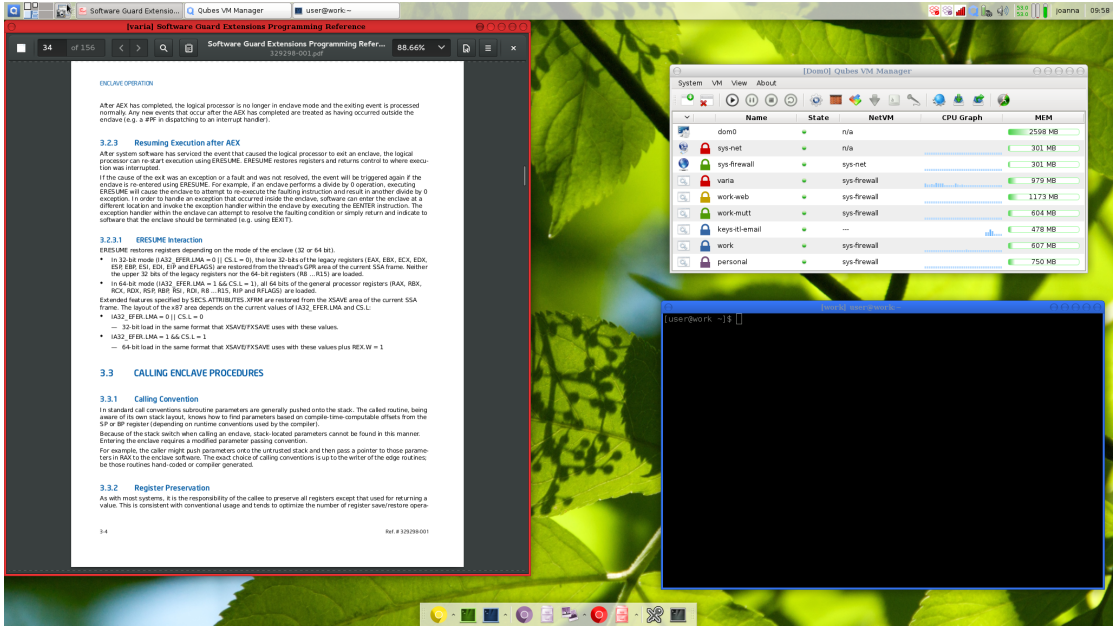
advantage: easier to gaurentee isolation

many, many more bugs in system call filtering than VMs

disadvantage: harder to share between VMs

disadvantage: much more runtime overhead

Qubes screenshot



which sandboxing?

which whole-application sandboxing technique seems better for

security, performance, usability, handling unchanged applications

(full answer: could mix techniques + probably depends on details of app)

A. chroot + system call filtering

B. chroot + mount and user namespaces

C. virtual machine dedicated to application

D. SELinux-like mandatory access control

sandboxing without OS support

so far: relying on OS features for sandboxing

good reasons:

- primarily want to filter system calls
- hardware-assisted, strong protection

but problems with relying on OS:

- sending information in/out of sandbox relatively slow
- requires heavily OS-specific code

sandboxing without OS ideas

'dynamic' language virtual machine, like Java VM, .Net CLR

- hard to use with code intended to compile to native machine code

virtual machine targetted for C/C++-like code, like WebAssembly

assembly-to-assembly conversion

- example: Wahbe, Lucco, Anderson, and Graham, "Efficient Software-Based Fault Isolation" (1993)
- example: Ford and Cox, "Vx32: Lightweight User-level Sandboxing on the x86" (2008)

WebAssembly

WebAssembly: language virtual machine specification intended...

- similar idea to Java VM

to be compiled to from C/C++

- support by Clang/LLVM

to be easy to just-in-time compile to native machine code

to be run in web browsers (fast web apps)

WebAssembly memory management

WebAssembly 'modules' have a single "linear memory"

starts at index 0, goes to some maximum

load/store instructions take index into current memory

observation 1: close to memory model "normal" C/C++ code expects

observation 2: only goal is to prevent sandbox (WebAssembly) code from interfering with outside code

...so no need to check array bound or similar

observation 3: no need to worry about garbage collection

WebAssembly validation

WebAssembly virtual machine code designed to be *validated* before running

allows for efficient interpreters or conversion to assembly

- validation ensures that you can safely skip certain type checks, etc.

language specification very explicit about what needs to be checked at runtime

example WebAssembly validation

check that instructions have right number of operands available

- WebAssembly instructions use stack (compile 2 + 2 into 2 2 +)

check operands that can be checked (constants)

check the calls go to only functions listed in table

- should make it easier to do just-in-time compilation to machine code?

check the branches go to only locations listed in table, and only within one function

- should make it easier to do just-in-time compilation to machine code?

example WebAssembly instruction specification

return

1. Let F be the current frame.
2. Let n be the arity of F.
3. Assert: due to validation, there are at least n values on the top of the st
4. Pop the results val^n from the stack.
5. Assert: due to validation, the stack contains at least one frame.
6. While the top of the stack is not a frame, do:
    a. Pop the top element from the stack.
7. Assert: the top of the stack is the frame F.
8. Pop the frame from the stack.
9. Push val^n to the stack.
10. Jump to the instruction after the original call that pushed the frame.

WebAssembly as sandboxing

can compile existing C/C++ library using WebAssembly...

then call using language virtual machine

RLBox

saw interfaces for using sandboxes from user perspective?

what about for privilege separation?

- recall: like Chrome separate renderer process idea
- need to navigate OS sandboxing API + create interface for sandboxed part?

some reusable tools have appeared for this (but no clear winner)

one example: RLBox (published in Usenix Security 2020)

- Shravan Narayan and Craig Disselkoen, UC San Diego; Tal Garfinkel, Stanford University; Nathan Froyd and Eric Rahm, Mozilla; Sorin Lerner, UC San Diego; Hovav Shacham, UT Austin; Deian Stefan, UC San Diego

RLBox usage

part of example from author's presentation:

goal: invoke JPEG parser in sandbox

```
autosandbox = rlbox::create_sandbox<wasm>();
tainted<jpeg_decompress_struct*> p_jpeg_img = sandbox.malloc_in_sandbox<jpeg_decompress_struc
tainted<jpeg_source_mgr*> p_jpeg_input_source_mgr = sandbox.malloc_in_sandbox<jpeg_source_mgr
sandbox.invoke(jpeg_create_decompress, p_jpeg_img);
p_jpeg_img->src = p_jpeg_input_source_mgr;
p_jpeg_img->src->fill_input_buffer = ...;
sandbox.invoke(jpeg_read_header,p_jpeg_img/*...*/);
```

tool handles running 'jpeg_create_decompress', 'jpeg_read_header' in sandbox

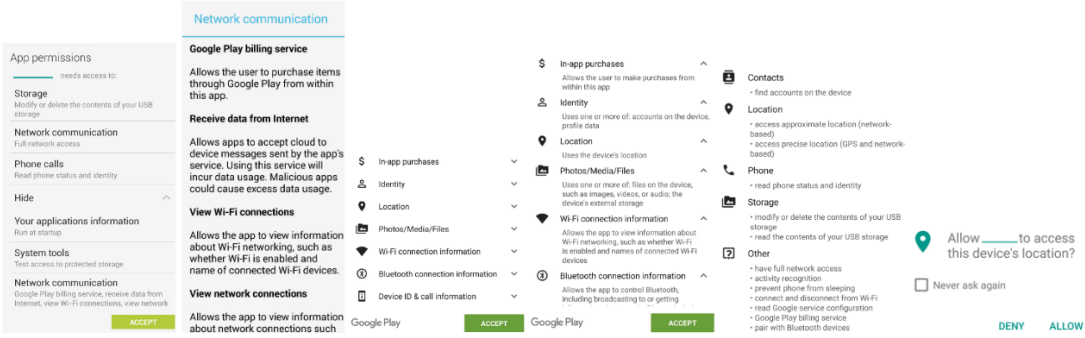
values shared with sandbox marked as "tainted"

C++ (template) class

this example: using WebAssembly-based sandbox

used in firefox

some Android prompts



(a) Explicit (b) Explicit(II) (c) Grouped (d) Grouped(II) (e) Grouped(III) (f) Request

Figure 1: The permissions displays under consideration. From left to right: explicit permissions model (Explicit) prior to Play Store 4.8.20, expanded explicit permissions (Explicit(II)) for "Network Communication", grouped permissions (Grouped) after Play Store 4.8.20, expanded grouped permissions (Grouped(II)) for all displayed categories, detailed group permissions (Grouped(III)) for the app on the Play Store, and a permission request (Request) for Location in Android M.

UI problems with application permissions

do applications request sensible permissions?

do users pay attention to permission requests?

do users understand what permissions mean?

are permissions fine-grained enough?

are permissions coarse-grained enough?

UI problems with application permissions

do applications request sensible permissions?

do users pay attention to permission requests?

do users understand what permissions mean?

are permissions fine-grained enough?

are permissions coarse-grained enough?

right permissions?

Felt, Chin, Hanna, Song and Wagner, "Android Permissions Demystified" (CCS 2011)

used static analysis to compare requested permissions to what applications did

- at the time: permissions requested at installation

sample of 900 applications

estimate approx 200 over-privileged

- (estimate because using false positive rate from manual checking)

why extra permissions?

selected from Felt et al's analysis:

developers confused similar permissions

- `ACCESS_NETWORK_STATE` versus `ACCESS_WIFI_STATE`

developers thought permissions were needed for delegated tasks

- `CALL_PHONE` not needed to invoke phone app
- `INSTALL_APPLICATION` not needed to open app store install dialog

developers thought permissions needed for all methods of class

- `WRITE_SETTINGS` when using (no-permission) read-settings operations

copy-and-paste

UI problems with application permissions

do applications request sensible permissions?

do users pay attention to permission requests?

do users understand what permissions mean?

are permissions fine-grained enough?

are permissions coarse-grained enough?

a user study (2012)

Felt, Ha, Egelman, Haney, Chin, Wagner, "Android Permissions: User Attention, Comprehension, and Behavior"

performed lab study; task: find + install coupon app

at the time: Android prompted for permissions on installation

a user study (2012)

Felt, Ha, Egelman, Haney, Chin, Wagner, "Android Permissions: User Attention, Comprehension, and Behavior"

performed lab study; task: find + install coupon app

at the time: Android prompted for permissions on installation

17% looked at app permissions detail

42% aware of permissions

42% unaware of permissions

versus: 88% read reviews

a user survey (2012)

same paper did survey about what permissions meant

three multiple choice questions

selected from bank of 11

302 respondents; 3 fully correct

average 21%

example survey question

'Read phone state and identity' allows which of these?

Read your phone number

See who you have called

Track you across applications

Load adverisements

survey questions (1)

INTERNET Category: Network communication Label: Full Internet access	109	✔ Send information to the application's server	45	41.3%
		✔ Load advertisements	30	27.5%
		✗ None of these	16	14.7%
		✗ Read your text messages	13	11.9%
		✗ Read your list of phone contacts	11	10.1%
		I don't know	36	33.0%
READ_PHONE_STATE Category: Phone calls Label: Read phone state and identity	85	✔ Read your phone number	41	47.7%
		✗ See who you have called	37	43.0%
		✔ Track you across applications	20	23.3%
		✗ Load advertisements	11	12.8%
		✗ None of these	10	11.6%
		I don't know	15	17.4%
CALL_PHONE Category: Services that cost you money Label: Directly call phone numbers	83	✔ Place phone calls	30	35.3%
		✗ Charge purchases to your credit card	27	31.8%
		✗ None of these	16	18.8%
		✗ See who you have made calls to	14	16.5%
		✗ Send text messages	11	12.9%
		I don't know	16	18.8%
WRITE_EXTERNAL_STORAGE Category: Storage Label: Modify/delete SD card contents	92	✔ Read other applications' files on the SD card	41	44.6%
		✔ Change other applications' files on the SD card	39	42.4%
		✗ None of these	16	17.4%
		✗ See who you have made phone calls to	15	16.3%
		✗ Send text messages	11	12.0%
		I don't know	15	16.3%

survey questions (2)

WRITE_EXTERNAL_STORAGE Category: Storage Label: Modify/delete SD card contents	92	✔ Read other applications' files on the SD card	41	44.6%
		✔ Change other applications' files on the SD card	39	42.4%
		✗ None of these	16	17.4%
		✗ See who you have made phone calls to	15	16.3%
		✗ Send text messages	11	12.0%
		I don't know	15	16.3%
WAKE_LOCK Category: System tools Label: Prevent phone from sleeping	81	✔ Keep your phone's screen on all the time	49	60.5%
		✔ Drain your phone's battery	37	45.7%
		✗ None of these	7	8.6%
		✗ Send text messages	4	4.9%
		✗ Delete your list of contacts	4	4.9%
		I don't know	13	16.0%
CHANGE_NETWORK_STATE Category: System tools Label: Change network connectivity	66	✔ Turn your WiFi on or off	36	52.9%
		✗ Send information to the application's server	13	19.1%
		✗ Read your calendar	7	10.3%
		✗ None of these	7	10.3%
		✗ See who you have made calls to	5	7.4%
		I don't know	17	25.0%

survey questions (3)

READ_SMS$_2$ Category: Your messages Label: Read SMS or MMS	54	✔ Read text messages you've sent	30	54.5%
		✔ Read text messages you've received	25	45.5%
		✗ Send text messages	10	18.2%
		✗ Read your phone's unique ID	6	10.9%
		✗ None of these	4	7.3%
		I don't know	11	20.0%
READ_SMS$_1$ Category: Your messages Label: Read SMS or MMS	77	✔ Read text messages you've received	44	56.4%
		✗ Read e-mail messages you've received	30	38.5%
		✗ Read your call history	13	16.7%
		✗ None of these	8	10.3%
		✗ Access your voicemail	8	10.3%
		I don't know	13	16.7%
READ_CALENDAR Category: Your personal information Label: Read calendar events	101	✔ Read your calendar	56	53.3%
		✗ None of these	18	17.1%
		✗ Add new events to your calendar	12	11.4%
		✗ Send text messages	12	11.4%
		✗ Place phone calls	9	8.6%
		I don't know	19	18.1%

survey questions (4)

READ_CONTACTS Category: Your personal information Label: Read contact data	86	✔ Read your list of contacts ✔ Read your call history ✗ None of these ✗ Delete your list of contacts ✗ Place phone calls *I don't know*	52 19 14 9 5 14	60.5% 22.1% 16.3% 10.5% 5.8% 16.3%
CAMERA Category: Hardware controls Label: Take pictures	72	✔ Take pictures when you press the button ✔ Take pictures at any time ✔ See pictures taken by other applications ✔ Delete pictures taken by other apps ✗ None of these *I don't know*	27 27 16 13 13 17	37.0% 37.0% 21.9% 17.8% 17.8% 23.3%

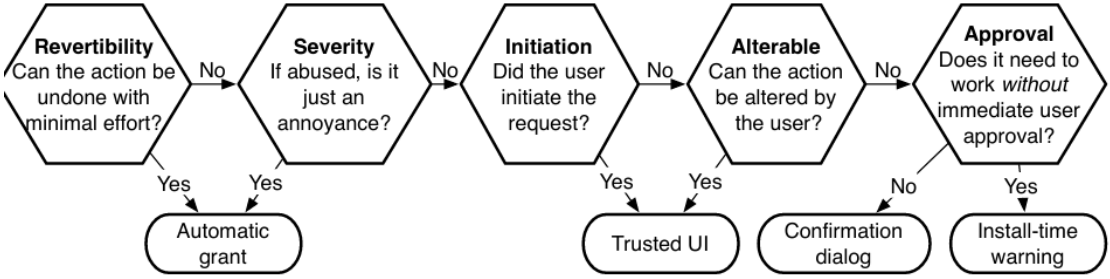


Figure 1: A guide to selecting between the different permission-granting mechanisms.

from Felt et al, "How To Ask For Permission" (HotSec'12)

principles

Felt et al list "principles":

"Conserve user attention, utilizaing it for only permissions that have severe consquences"

> too many security warnings means users won't pay attention

"When possible, avoid interrupting the user's primary task with explicit security decisions"

> users will dismiss warnings because they get in the way of work

Cloak and Dagger

Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop

Yanick Fratantonio
UC Santa Barbara
yanick@cs.ucsb.edu

Chenxiong Qian, Simon P. Chung, Wenke Lee
Georgia Tech
qchenxiong3@gatech.edu
pchung34@mail.gatech.edu
wenke.lee@gmail.com

cloak and dagger permissions

the two permissions:

SYSTEM_ALERT_WINDOW:
draw windows on top of screen
(at time: enabled by default)
BIND_ACCESSIBILITY_SERVICE:
"Observe your actions"
"Retrieve window content"

can hide window content while user interacts with it

...and stealthy get user to do more things

also, a clickjacking attack

at the time, could draw overlay window over permissions dialog

...convince user to press where "OK" button is

countermeasure: permissions dialog would detect this, ignore clicks

problem: wouldn't detect if overlay didn't cover enough of button

privacy and permissions

50 Ways to Leak Your Data:
An Exploration of Apps' Circumvention of the Android Permissions System

Joel Reardon
University of Calgary
AppCensus, Inc.

Álvaro Feal
IMDEA Networks Institute
Universidad Carlos III de Madrid

Primal Wijesekera
U.C. Berkeley / ICSI

Amit Elazari Bar On
U.C. Berkeley

Narseo Vallina-Rodriguez
IMDEA Networks Institute / ICSI
AppCensus, Inc.

Serge Egelman
U.C. Berkeley / ICSI
AppCensus, Inc.

2019 paper

many mobile application permissions related to privacy

getting phone ID, email address, location, ...

but applications (especially ad libraries) find workarounds

permissions being insufficient

permissions check limited API calls for getting private info,...

...but there were alternative, unfiltered system calls for

getting MAC address (effectively phone ID)

- Linux `ioctl` system call on socket

WiFi base station address

- ARP cache (recently seen machines on network, to know where to send packets)

location

- geolocation tag on recent photos

covert channels

advertising libraries would store phone ID/account info in a file

...when they had permissions to retrieve it

and would read phone ID/account info from a file

...when they did not

backup slides