

Hunting CVEs for fun and profit

Flanker



#whoami

- Flanker
 - Senior Security Researcher at KeenLab
 - Apple/Android/Chrome CVE hunter (“frequent creditor”)
 - Speaker at BlackHat USA/ASIA, DEFCON, RECON, CanSecWest, HITCON, QMSS
 - Pwn2Own 2016/ Mobile Pwn2Own 2016 winner
 - Recognized researcher of Android Security Reward Program

I've been working on...

- Kernel fuzzing/auditing
- Privilege Escalation in Userspace
- Sandbox escapes
- Browser fuzzing/exploitation
- Android/macOS/iOS

Agenda

- Browser fuzzing/exploitation
- Sandbox escapes
- Privilege escalation
- Kernel code execution

Lifetime of a complete exploit chain

- Remote vector is usually browser
- Escalate the sandbox via
 - Broker IPC calls
 - Userspace privileged components
 - Kernel
- Privilege Escalation

Chain in Mobile Pwn2Own 2016 Android Category

THE ENTRANCE – JAVASCRIPT ENGINES

V8 Javascript Engine

- Widely known and used
- Runtime optimization and JIT to machine code
 - Strongtalk
 - Crankshaft
 - Turbofan



Vulnerabilities in V8 showcase

- CVE-2016-1646
 - Property redefinition
- CVE-2016-5198
 - JIT optimization out-of-bound

Case study: CVE-2016-1646

- V8 Array.concat redefinition out-of-bounds in Pwn2Own 2016 by KeenLab

```
b = new Array(10);
b[0] = 0.1; <-- Note that b[1] is a hole!
b[2] = 2.1;
b[3] = 3.1;

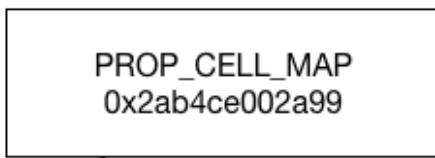
Object.defineProperty(b.__proto__, 1, { <-- define b.__proto__[1] to gain the control in the middle of the loop
  get: function () {
    b.length = 1; <-- shorten the array
    gc(); <-- shrink the memory
    return 1;
  },
  set: function(new_value){
    /* some business logic goes here */
    value = new_value
  }
});
});
```

Case study: CVE-2016-1646

```
case FAST_HOLEY_DOUBLE_ELEMENTS:  
case FAST_DOUBLE_ELEMENTS: {  
    //.....  
    for (int j = 0; j < fast_length; j++) {  
        HandleScope loop_scope(isolate);  
        if (!elements->is_the_hole(j)) {  
            double double_value = elements->get_scalar(j);  
            Handle<Object> element_value =  
                isolate->factory()->NewNumber(double_value);  
            visitor->visit(j, element_value);  
        } else {  
            Maybe<bool> maybe = JSReceiver::HasElement(array, j);  
            if (!maybe.IsJust()) return false;  
            if (maybe.FromJust()) {  
                Handle<Object> element_value;  
                ASSIGN_RETURN_ON_EXCEPTION_VALUE(  
                    isolate, element_value, Object::GetElement(isolate, array, j),  
                    false);  
                visitor->visit(j, element_value);  
            }  
        }  
    }  
}  
break;  
}
```

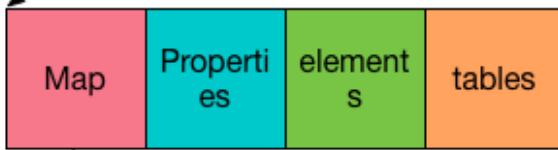
CVE-2016-5198 – oob in Deoptimization

- Eager Deoptimization
 - Usually seen in function argument checks
 - Bail out to interpreter mode immediately
- Lazy Deoptimization
 - Usually seen on global object access
 - Who changes the object is responsible for patching following users
 - What if itself is also JITed?



PropertyCell n: 0x79d334abfc1

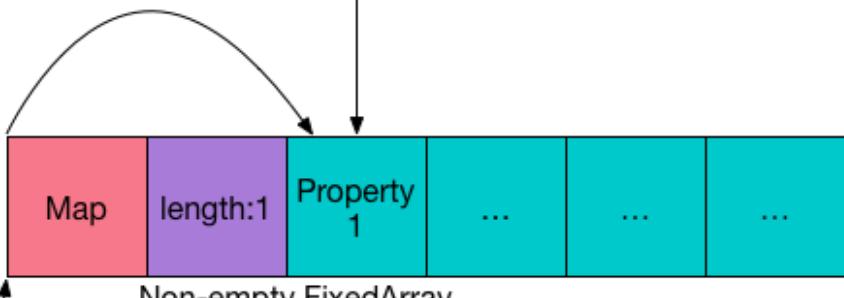
`mov rax,QWORD PTR [rax+0xf]`



JSSet: 0x130199d5c511

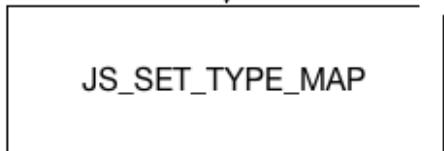
Javascript: n.xyz = 0x31337

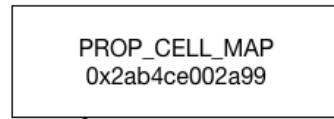
0x31337



Non-empty FixedArray

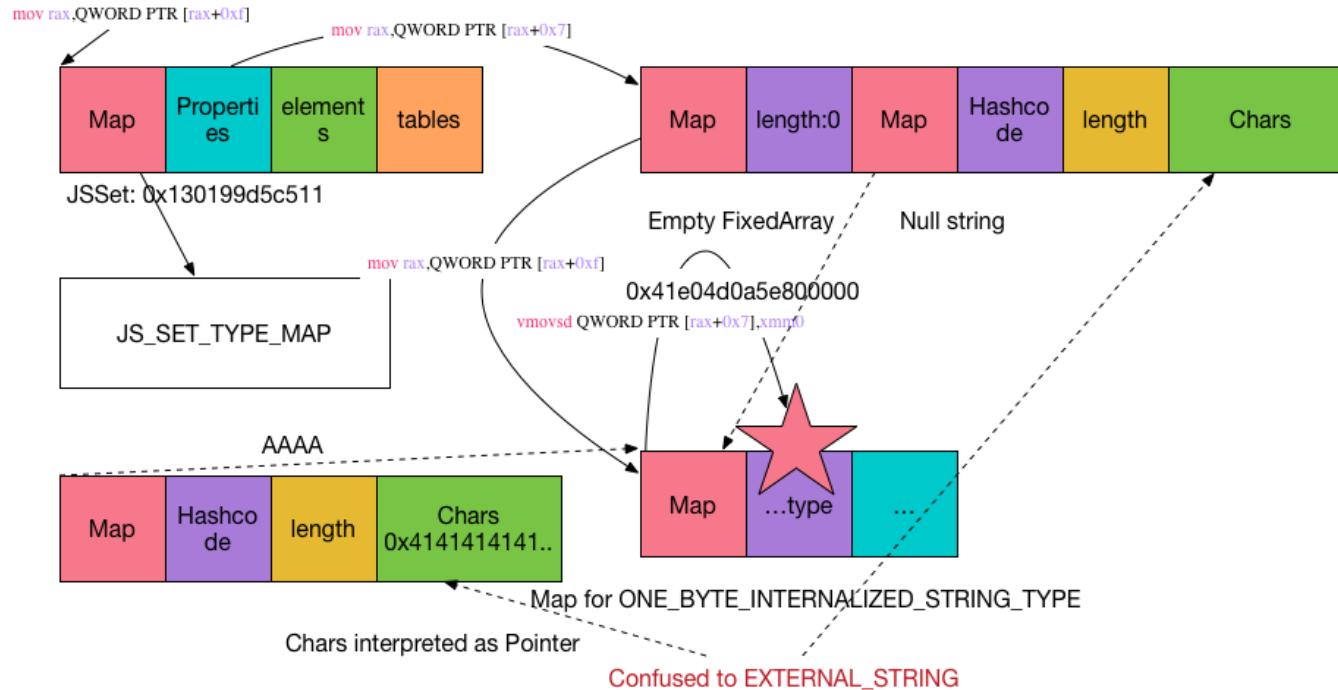
`mov rax,QWORD PTR [rax+0x7]`





PropertyCell n: 0x79d334abfc1

0x826852f4
gdb-peda\$ i r \$xmm0
xmm0
{
v4_float = {0x0, 0x1c, 0x0, 0x0},
v2_double = {0x826852f4, 0x0},
v16_int8 = {0x0, 0x0, 0x80, 0x5e, 0xa, 0x4d, 0},
v8_int16 = {0x0, 0x5e80, 0xd0a, 0x41e0, 0x0, 0x0, 0x0},
v4_int32 = {0x5e800000, 0x41e04d0a, 0x0, 0x0},
v2_int64 = {0x41e04d0a5e800000, 0x0},
uint128 = 0x000000000000000041e04d0a5e800000
}



Exploiting CVE-2016-5198

- OOB write chars field of null string to leak ArrayBuffer address
- Overwrite ArrayBuffer **backing_store** to leak Function code address
- Overwrite ArrayBuffer **backing_store** with Function code address
- Write shellcode to ArrayBuffer and exec!

How to fuzz Javascript engines?

- Jsfunfuzz can be a good start
- Collect samples
 - Split, mutate and join
- New features, new vulnerabilities
 - Callbacks, protos, ...
 - Intentionally generate them

Chain in Pwn2Own 2016 OSX category

BREAKING SANDBOX LIKE A BOSS

Sandbox

- In modern operating systems, a “Sandbox” is a mechanism to run code in a constrained environment.
- A Sandbox specifies which resources this code has access to
- Shift of approach/complementary approach:
 - Let's confine software, so even if it's compromised it has restricted access to the system.

Structure of the Safari Sandbox

- The UI Process is the parent and in charge of managing the other processes
- Web Process runs webkit/javascript engines

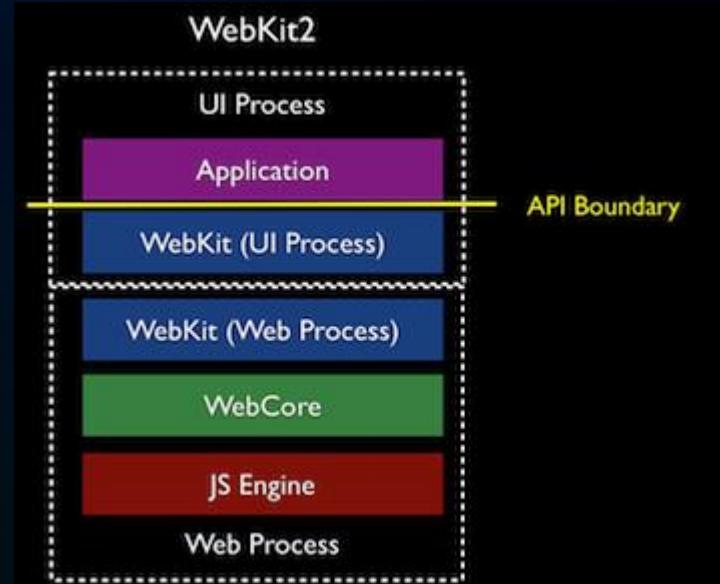
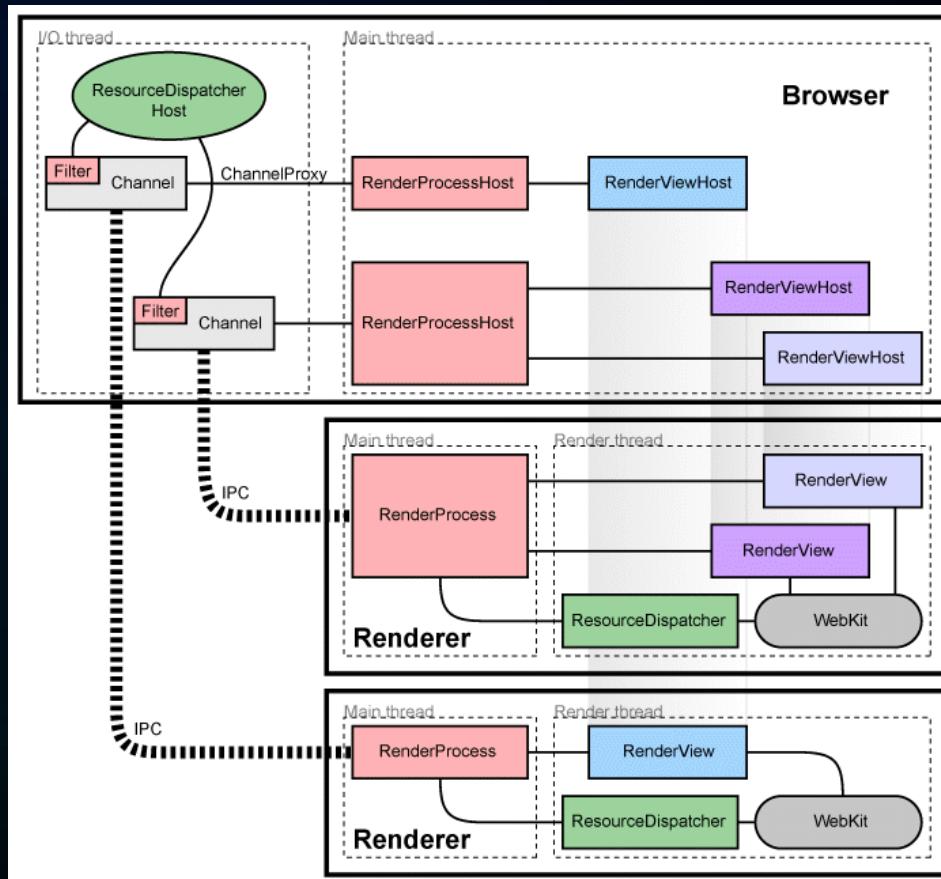


Image courtesy of:
<https://trac.webkit.org/attachment/wiki/WebKit2/webkit2-stack.png>

The anatomy of Chrome sandbox

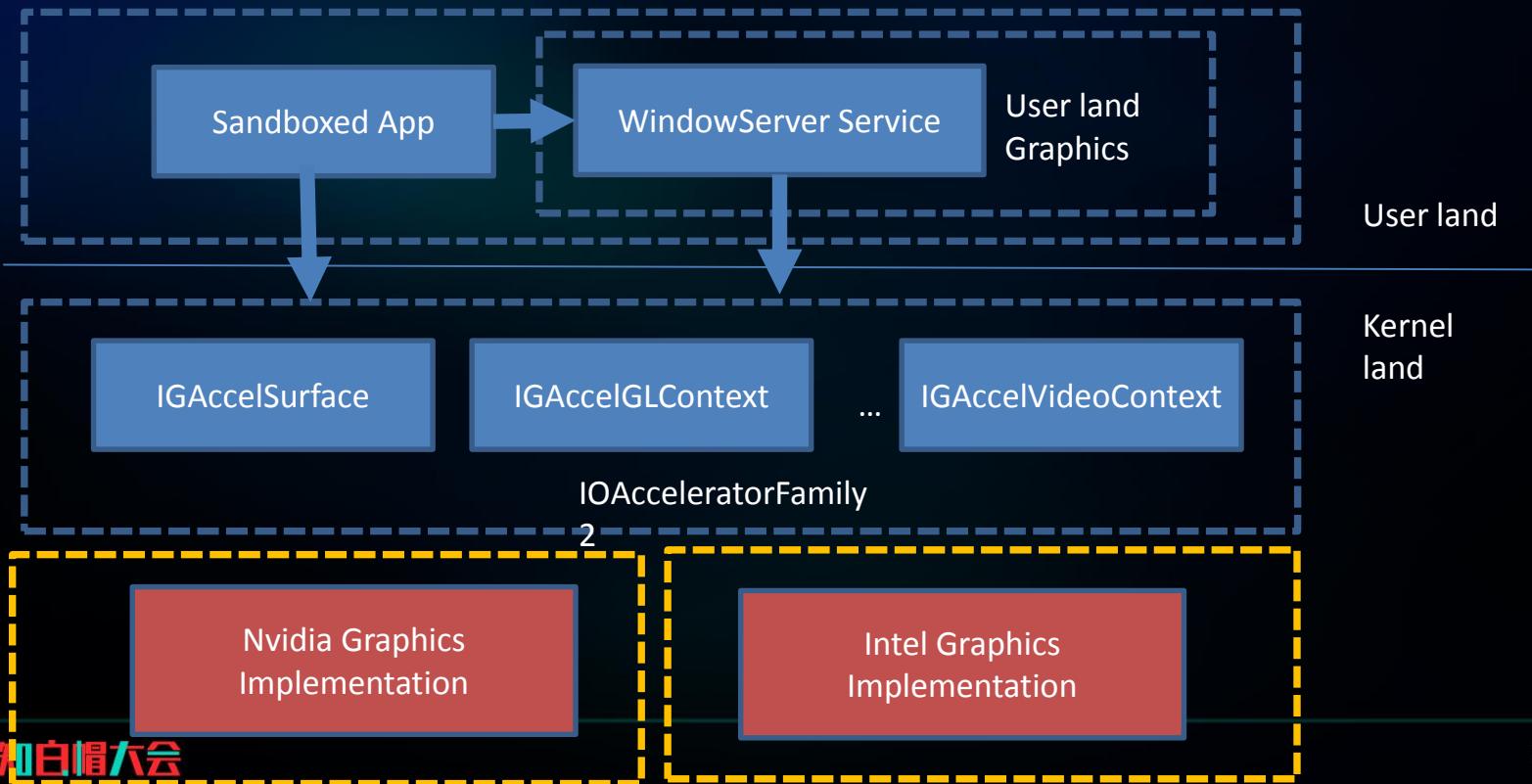
- All untrusted code runs in Target process
- Relay most operations to Broker
- Try best to
 - lock down the capabilities of renderer
- Even renderer is compromised
 - Access is still strictly prohibited
- GPU process have higher level access
 - Than normal sandbox process



How to escape the sandbox?

- To beat your enemies, know them first
- Sandbox profiles

Apple graphics architecture



- On macOS, stored in
/System/Library/Frameworks/WebKit.framework/Versions/A/Resources/com.apple.WebProcess.sb
- On iOS, binary file embed in kernel:
 - Sandbox_toolkit:
https://github.com/sektioneins/sandbox_toolkit
- What's in sandbox profile:
 - File operation
 - IPC
 - IOKit
 - Sharedmem
 - Etc.

```
(allow file-read  
    ;; Basic system paths  
    (subpath "/Library/Dictionarys")  
    (subpath "/Library/Fonts")  
    (subpath "/Library/Frameworks")  
    (subpath "/Library/Managed Preferences")  
    (subpath "/Library/Speech/Synthesizers")  
    (regex #"/private/etc/(hosts|group|passwd)$")
```

```
; Various services required by AppKit and other frameworks  
(allow mach-lookup  
    (global-name "com.apple.DiskArbitration.diskarbitrationd")  
    (global-name "com.apple.FileCoordination")  
    (global-name "com.apple.FontObjectsServer")  
    (global-name "com.apple.FontServer")  
    (global-name "com.apple.SystemConfiguration.configd")  
    (global-name "com.apple.SystemConfiguration.PPPController")  
    (global-name "com.apple.audio.VDCAssistant")  
    (global-name "com.apple.audio.audiohald")  
    (global-name "com.apple.audio.coreaudiod")
```

```
;; IOKit user clients  
(allow iokit-open  
    (iokit-user-client-class "AppleUpstreamUserClient")  
    (iokit-user-client-class "IOHIDParamUserClient")  
    (iokit-user-client-class "RootDomainUserClient")  
    (iokit-user-client-class "IOAudioControlUserClient")  
    (iokit-user-client-class "IOAudioEngineUserClient"))
```

Escaping the Safari sandbox

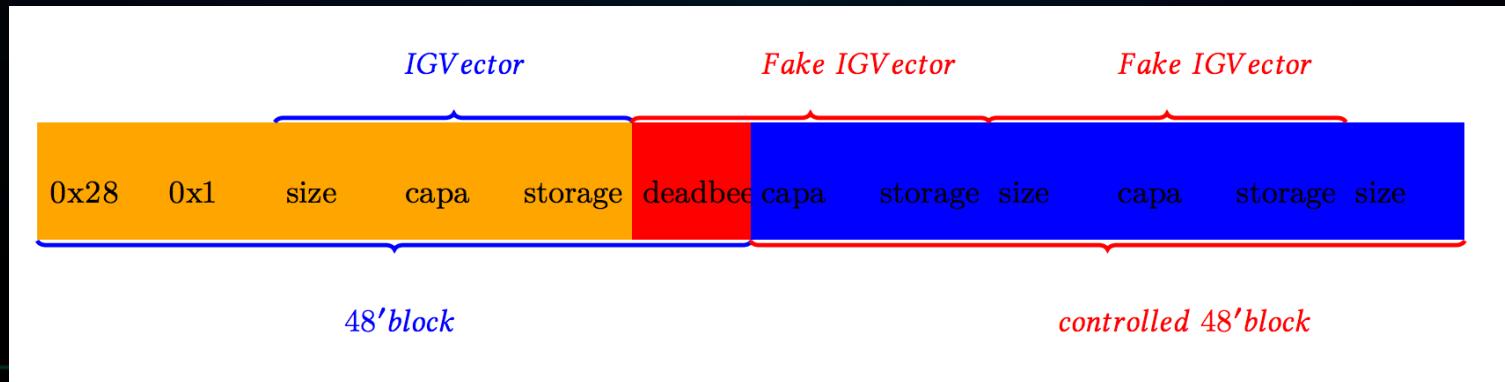
- Fuzzing Graphics IOKit calls
 - Actively generate
 - Passive injection
 - Coverage guidance – instrument the xnu kernel
- Fuzzing XPCs in privileged userspace daemons
 - Yes windowserver I'm talking about you

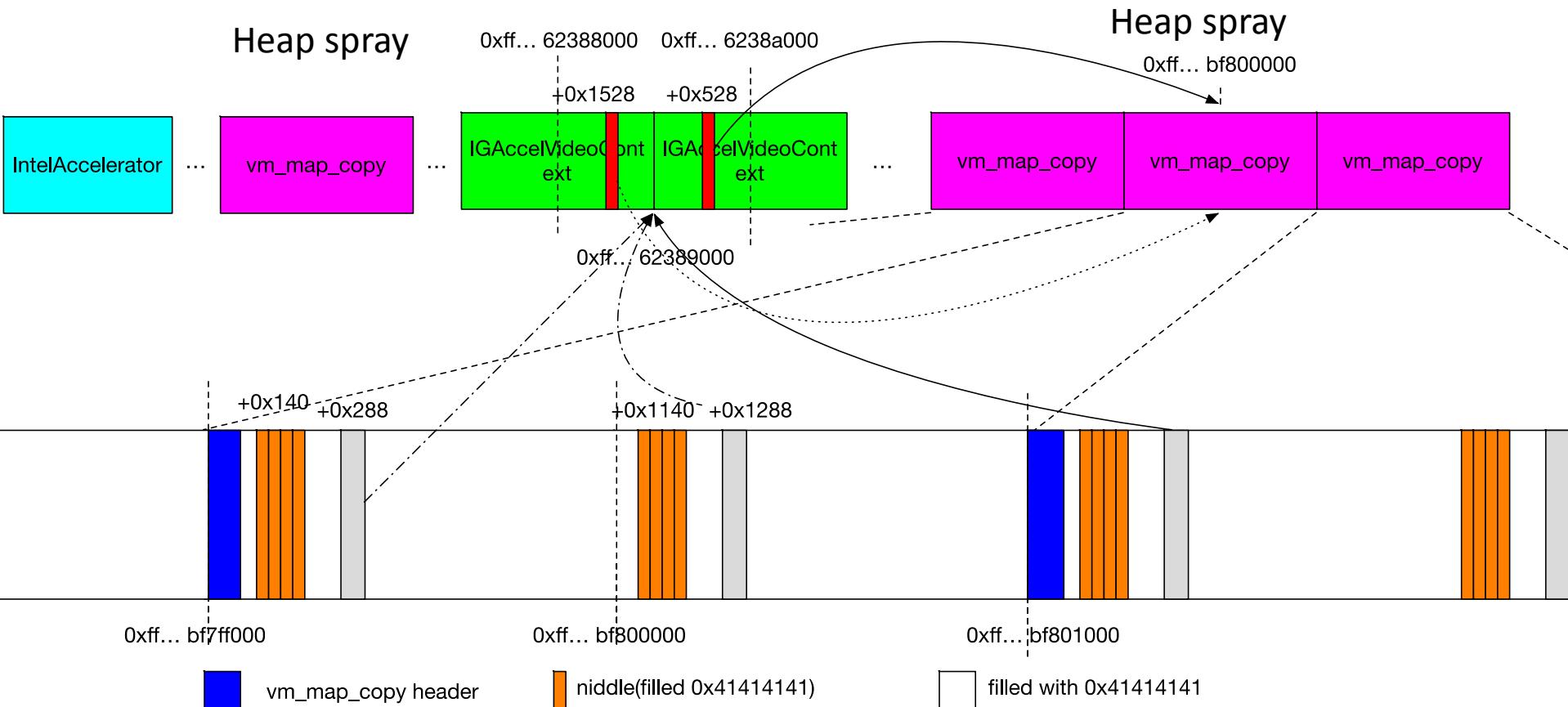
Python/Go wrappers for fuzzing

- Easy SMT solvers integration
- Feasible strategy evolution
- Import kitlib

CVE-2016-1815 – ‘Blitzard’ - our P2O bug

- This bug lies in IOAcceleratorFamily
- A vector write goes out-of-bound under certain carefully prepared situations (8 IOKit calls) in a newly allocated kalloc.48 block
- Finally goes into IGVector::add lead to OOB write
- Arbitrary-write-but-content-limited





Evolution of the Android Sandbox (old time)



Evolution of the Android Sandbox (current state)



Chromium Android Sandbox (1)

- On Android, Chromium leverages the isolatedProcess feature to implement its sandbox.

```
{% for i in range(num_sandboxed_services) %}  
<service android:name="org.chromium.content.app.SandboxedProcessService{{ i }}"  
        android:process=":sandboxed_process{{ i }}"  
        android:permission="{{ manifest_package }}.permission.CHILD_SERVICE"  
        android:isolatedProcess="true"  
        android:exported="{{ sandboxed_service_exported|default(false)}}"  
        {% if (sandboxed_service_exported|default(false)) == 'true' %}  
        tools:ignore="ExportedService"  
        {% endif %}  
        {{sandboxed_service_extra_flags|default('')}} />  
{% endfor %}
```

Chromium Android Sandbox (2)

```
type isolated_app, domain;
app_domain(isolated_app)

# Access already open app data files received over Binder or local socket IPC.
allow isolated_app app_data_file:file { read write setattr lock };

allow isolated_app activity_service:service_manager find;
allow isolated_app display_service:service_manager find;

# only allow unprivileged socket ioctl commands
allow isolated_app self:{ rawip_socket tcp_socket udp_socket } unpriv_sock_ioctl;

#####
##### Neverallow
#####

# Isolated apps should not directly open app data files themselves.
neverallow isolated_app app_data_file:file open;

# b/17487348
# Isolated apps can only access two services,
# activity_service and display_service
neverallow isolated_app {
    service_manager_type
    -activity_service
    -display_service
}:service_manager find;

# Isolated apps shouldn't be able to access the driver directly.
neverallow isolated_app gpu_device:chr_file { rw_file_perms execute },
```

- Very restrictive Sandbox profile
- No data file access at all
- Only 2 IPC services
- Minimum interaction with sockets
- No graphic drivers access 
- ServiceManager also restricts implicit service export

Per interface constraint

- Isolated_app inherits from app_domain (app.te)
- Only interfaces without enforceNotIsolatedCaller can be invoked

```
void enforceNotIsolatedCaller(String caller) {
    if (UserHandle.isIsolated(Binder.getCallingUid())) {
        throw new SecurityException("Isolated process not allowed to call " + caller);
    }
}

void enforceShellRestriction(String restriction, int userHandle) {
    if (Binder.getCallingUid() == Process.SHELL_UID) {
        if (userHandle < 0
            || mUserManager.hasUserRestriction(restriction, userHandle)) {
            throw new SecurityException("Shell does not have permission to access user "
                + userHandle);
        }
    }
}

@Override
public int getFrontActivityScreenCompatMode() {
    enforceNotIsolatedCaller("getFrontActivityScreenCompatMode");
    synchronized (this) {
        return mCompatModePackages.getFrontActivityScreenCompatModeLocked();
    }
}
```

Userspace escapes in Android

- Broker IPCs
- Binder calls to privileged daemons
 - System server

```
bool RenderWidgetHostViewAndroid::OnMessageReceived(  
const IPC::Message& message) {  
//...  
bool handled = true;  
IPC_BEGIN_MESSAGE_MAP(RenderWidgetHostViewAndroid, message)  
IPC_MESSAGE_HANDLER(ViewHostMsg_StartContentIntent, OnStartContentIntent)  
IPC_MESSAGE_HANDLER(ViewHostMsg_SmartClipDataExtracted,  
OnSmartClipDataExtracted)  
IPC_MESSAGE_HANDLER(ViewHostMsg_ShowUnhandledTapUIIfNeeded,  
OnShowUnhandledTapUIIfNeeded)  
IPC_MESSAGE_UNHANDLED(handled = false)  
IPC_END_MESSAGE_MAP()  
return handled;  
}
```

```
public void onStartContentIntent(Context context, String intentUrl, boolean isMainFrame) {  
    Intent intent; // Perform generic parsing of the URI to turn it into an Intent.  
    try {  
        intent = Intent.parseUri(intentUrl, Intent.URI_INTENT_SCHEME);  
        String scheme = intent.getScheme();  
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  
    } catch (Exception ex) {  
        Log.w(TAG, "Bad URI %s", intentUrl, ex);  
        return;  
    }  
    try {  
        context.startActivity(intent);  
    } catch (ActivityNotFoundException ex) {  
    }  
}
```

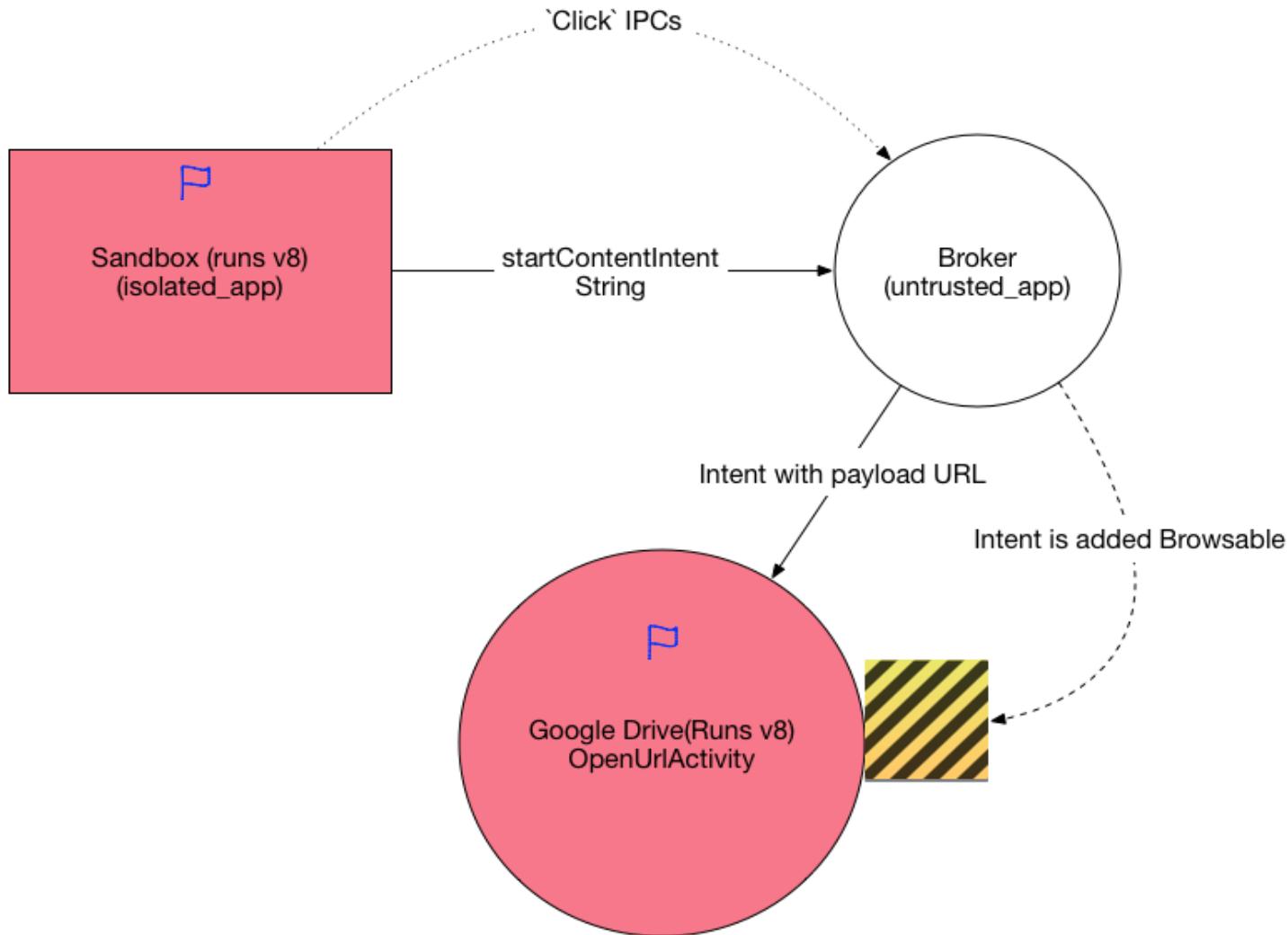
CVE-2016-5197

Arbitrary intent start in broker

IPC sandbox escapes

- See that holy Google Drive
- Have full access to Google account
- Trusted by Google Play
 - To “install” app
- **Blindly opens any intent-controlled URL**
- Pwn it to jump from isolated to untrusted
 - Plus App installation ability!

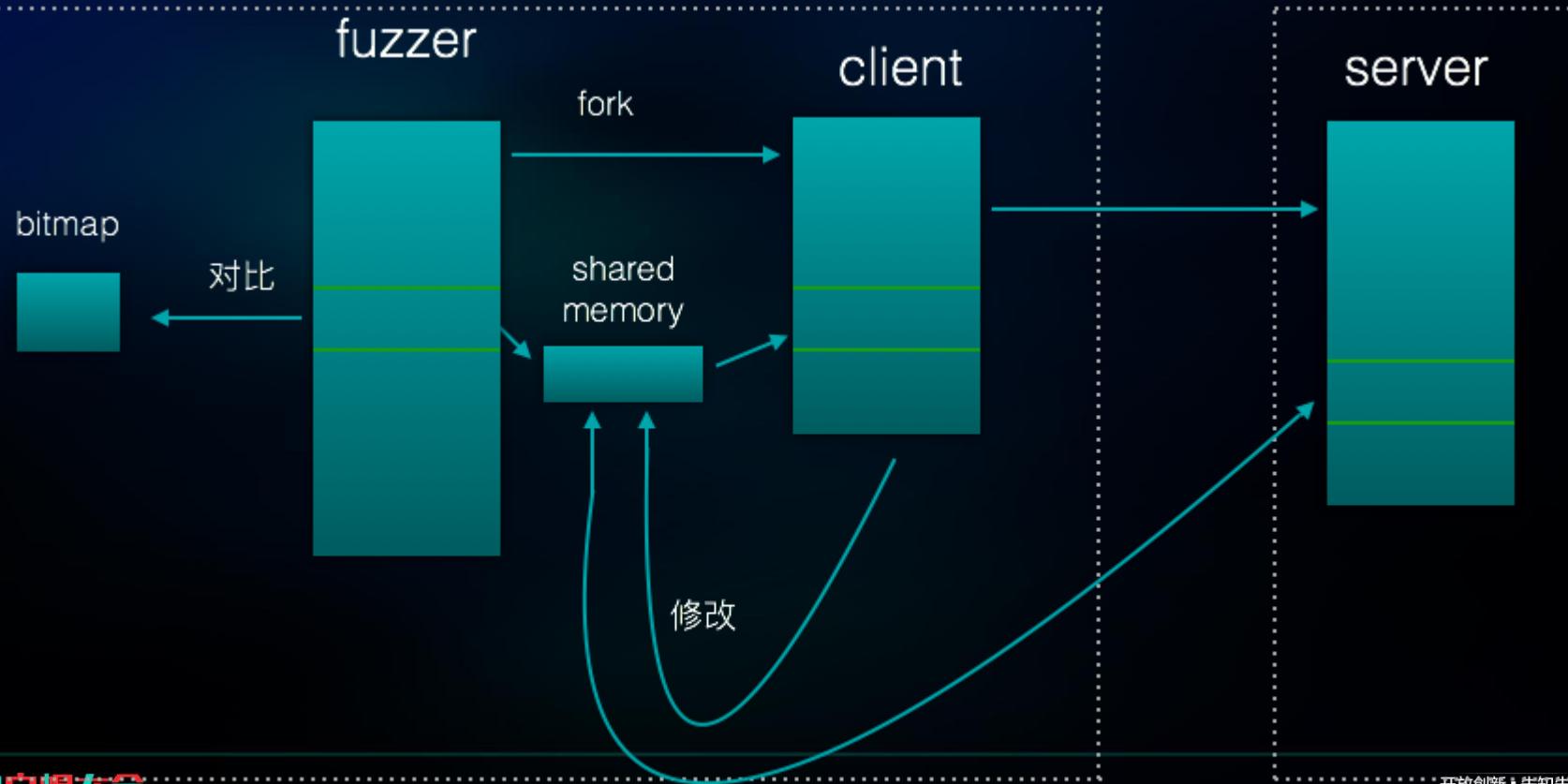




Privilege escalations

GO DEEPER

Fuzzing daemons with AFL+ASAN



Coverage guided kernel fuzzing

- GCC6 fully supports instrumentation in each basic block
- Coverage exported via /sys/kernel/debug/kcov
- Only samples increasing coverage survives
- KASAN instrumentation
- Go integration

Some bugs can be reached from sandbox

- CVE-2015-1805
- Race-condition in pipe_read
 - Waste for qiku though

Majority kernel bugs are not

- Example: **CVE-2015-6637** for Qiku phones rooting
 - Driver protected by SELinux policy
 - Userspace escalation come to rescue
 - **CVE-2016-3832**
- Credit also goes to James & nforest

CVE-2015-6637 in /dev/misc-sd

```
if(msdc_ctl->total_size <= 0) return -EINVAL;  
host_ctl = mtk_msdc_host[msdc_ctl->host_num]; <== Bug here  
BUG_ON(!host_ctl); BUG_ON(!host_ctl->mmc);  
  
if (host->ops->enable && !stop && host->claim_cnt == 1)  
host->ops->enable(host); <== Code execution
```

Who can access misc-sd?

- Em_srv is a system executable, holds em_svr context
`# ps -Z | grep "em_svr"
u:r:em_svr:s0 system 619 1 /system/bin/em_svr`
- Listens on @EngineerModeServer socket
- Execute command

```
# cat qiku_av.txt | grep "ALLOW " | grep ">misc_sd_device" | grep "ioctl"  
[AV] 4378: ALLOW factory-->misc_sd_device (chr_file) [ioctl read open]  
[AV] 7554: ALLOW em_svr-->misc_sd_device (chr_file) [ioctl read open]  
[AV] 10552: ALLOW unconfineddomain-->misc_sd_device (file) [append create write ... [AV] 10556: ALLOW  
recovery-->misc_sd_device (chr_file) [append create execute ... [AV] 10559: ALLOW unconfineddomain-->  
misc_sd_device (chr_file) [append create ... [AV] 10562: ALLOW vold-->misc_sd_device (chr_file)  
[ioctl read open]  
[AV] 12202: ALLOW mmc_ffu-->misc_sd_device (chr_file) [ioctl read open]
```

How can it access misc-sd?

- SELinux forbids `em_svr` from running /data executable directly
- But /system/bin/toolbox keeps ioctl gadget for our interest ☺

```
# cat qiku_av.txt | grep "ALLOW " | grep "em_svr-->" | grep "execute"  
[AV] 4418: ALLOW em_svr-->system_file (file) [execute_no_trans]  
[AV] 5393: ALLOW em_svr-->shell_exec (file) [execute read execute_no_trans open] [AV]  
6076: ALLOW em_svr-->thermal_manager_exec (file) [execute getattr read ...] [AV] 7135:  
ALLOW em_svr-->em_svr_exec (file) [execute getattr read entrypoint open]
```

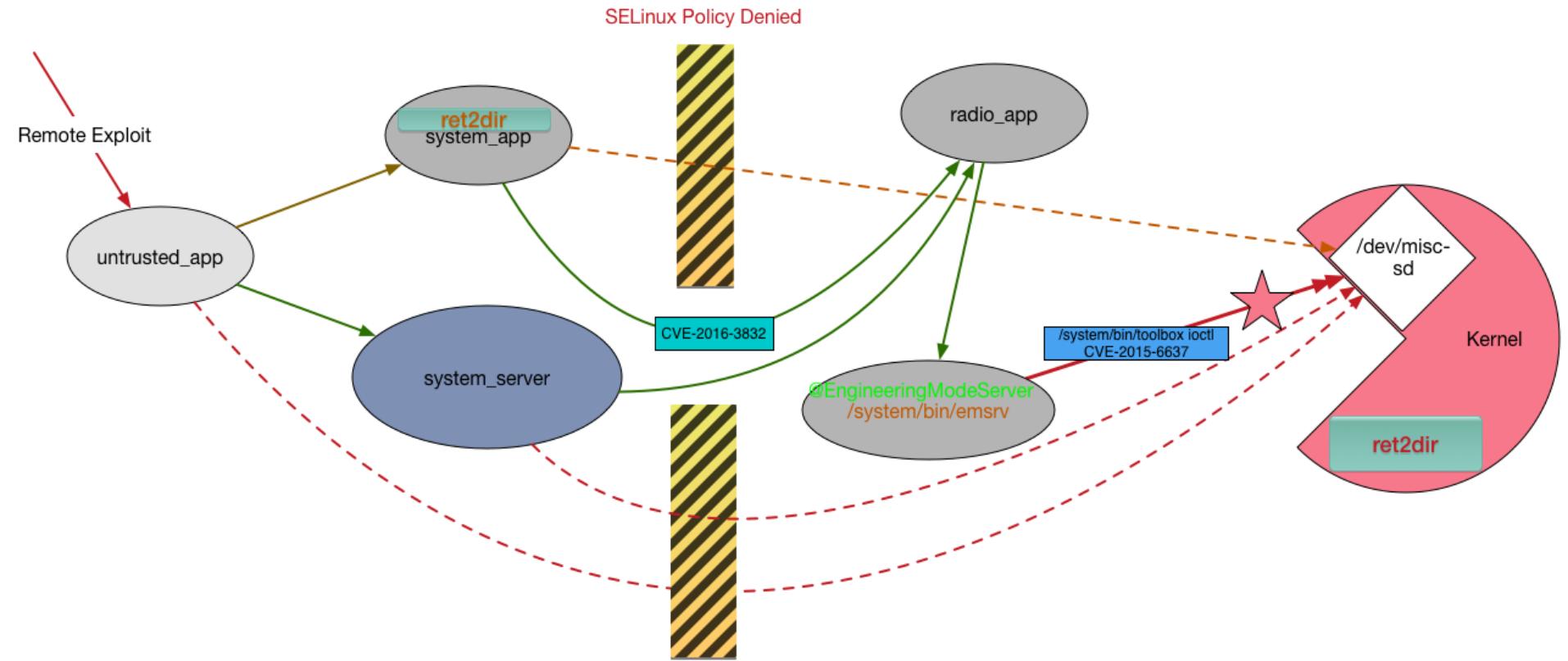
Who can access em_srv?

- Radio uid
- But how?

```
1.# cat qiku_av.txt | grep "ALLOW " | grep ">em_svr" | grep "connect"  
[AV] 2244: ALLOW radio-->em_svr (unix_stream_socket) [connectto]  
[AV] 8567: ALLOW em_svr-->em_svr (unix_stream_socket) [append bind  
connect ... [AV] 8571: ALLOW em_svr-->em_svr (unix_dgram_socket)  
[append bind connect ...
```

Do you have radio contact?

- Get `system_server` context
 - Transient to `radio`
- Or:
 - `bindBackupAgent` provides a way for us to get arbitrary context/uid from `system_app` context
 - How to get `system_app` uid?
 - Too easy on MTK phones..



Conclusion

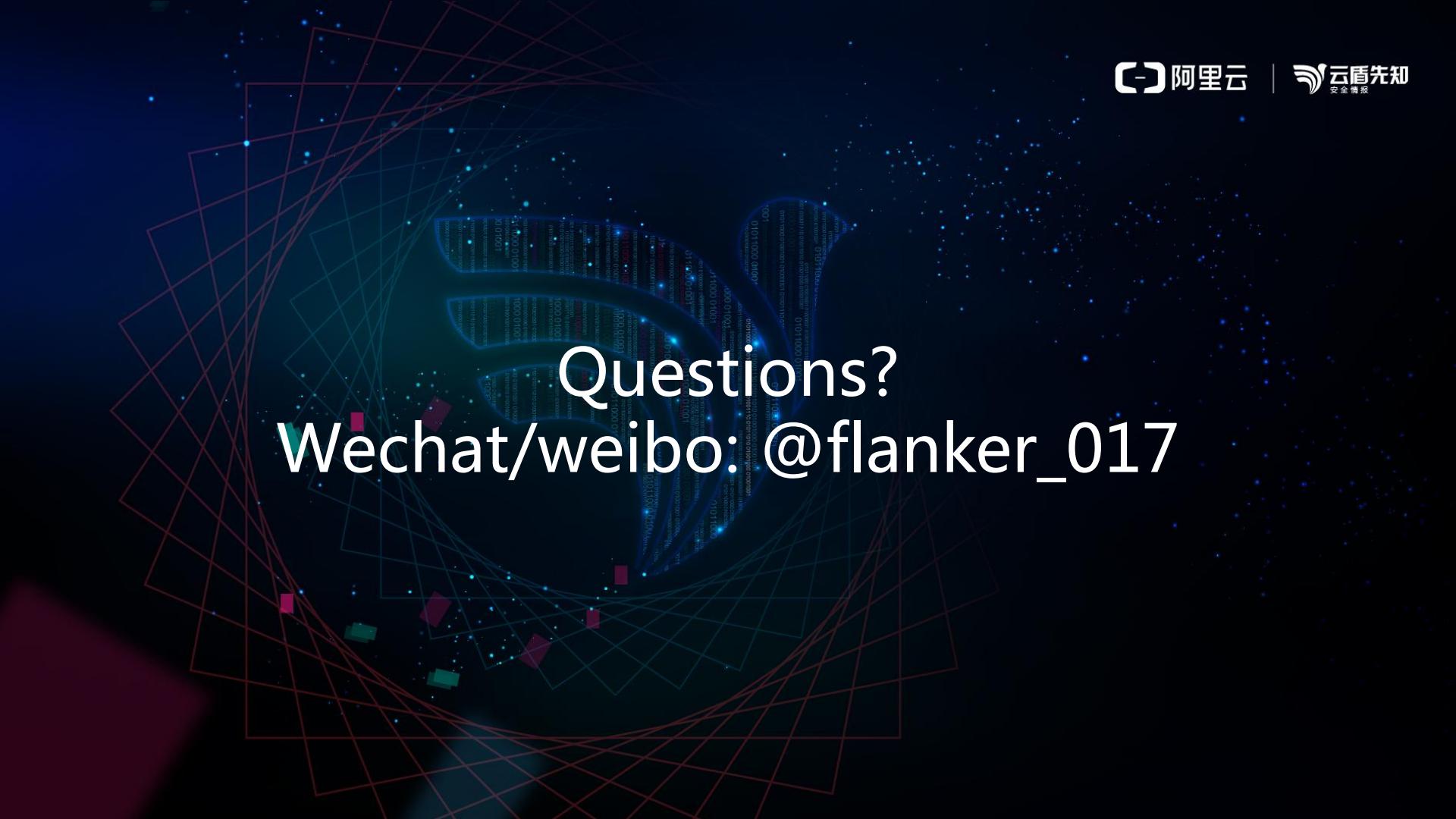
- Fuzzing is proved useful against complex system with coverage guidance
 - Domain knowledge lead to better result
- Mitigations make whole exploit chain longer and longer
 - Multiple vulnerabilities required than before, e.s.p on Google products

Fun!Profit?Profit!

- \$\$\$
- \$\$\$\$\$\$
- \$\$\$\$\$\$\$\$

Credits

- All colleagues at KeenLab



Questions?
Wechat/weibo:@flanker_017

THANKS!