Mobile Exploitation - The past, present, and the future

Ki Chan Ahn @externalist

ZERONCON

DATAFLOW SECURITY

What this talk is about

- Evolution of Attackers & Defenders on browser exploit chains (Initially wanted to cover domains other than browsers as well, but slides exploded )
- The meta game of Android / iOS bughunting & exploitation
- How it used to be, how it's changing, and what to expect in the future
- Mostly browser exploitation centric, but some short comments on other domains

DATAFLOW SECURITY

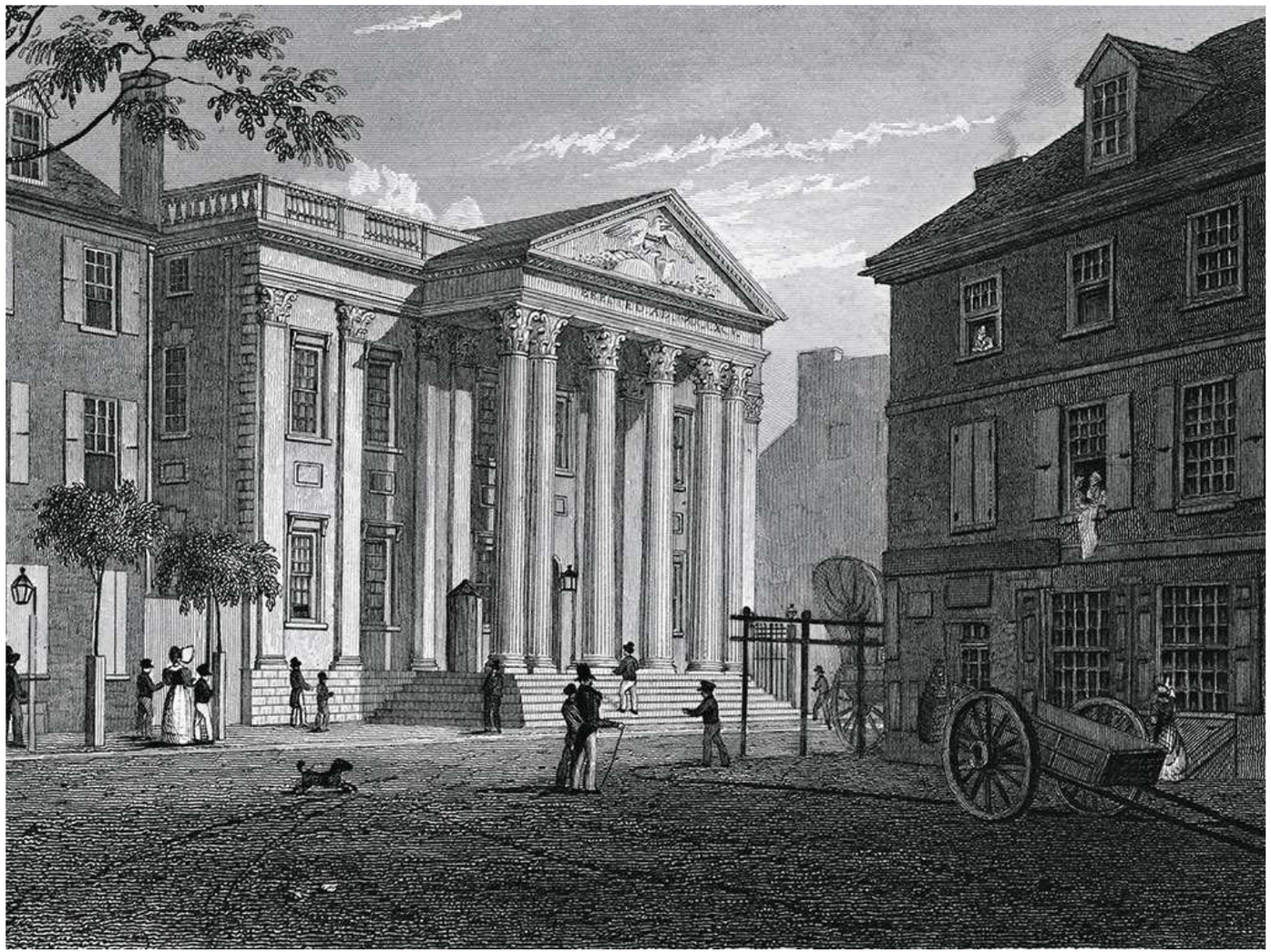
$ id

- Android Technical Lead @ Dataflow Security

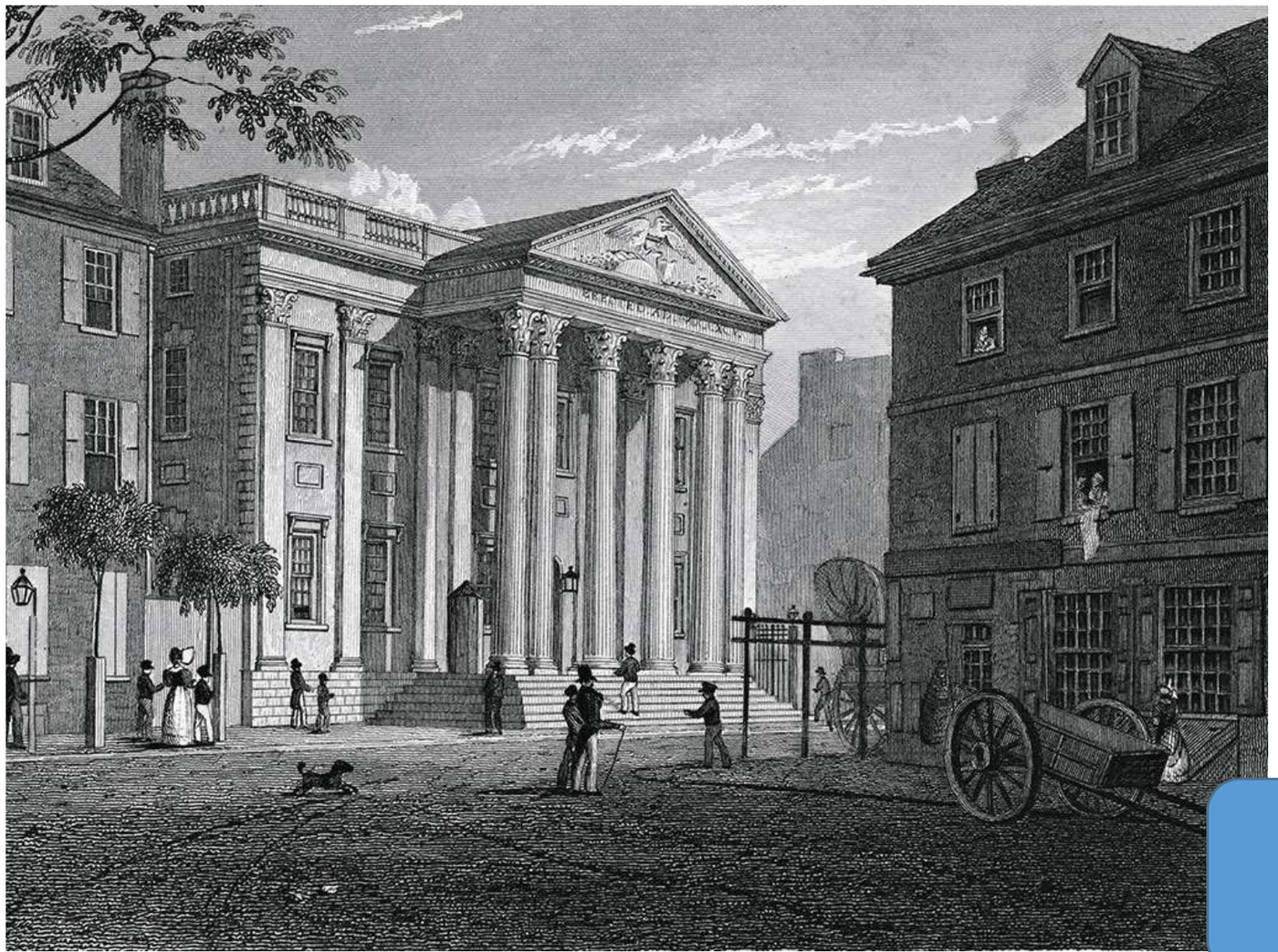
- Before Dataflow Security
  - Chrome & Safari RCE/SBX, Android & iOS PE, VM escape, pentesting…

- Dataflow Security
  - Chrome/Safari RCE/SBX, Android/iOS userland…

DATAFLOW SECURITY



The distant past



The distant past

Fast forward…

The recent past

Bughunting & Exploitation

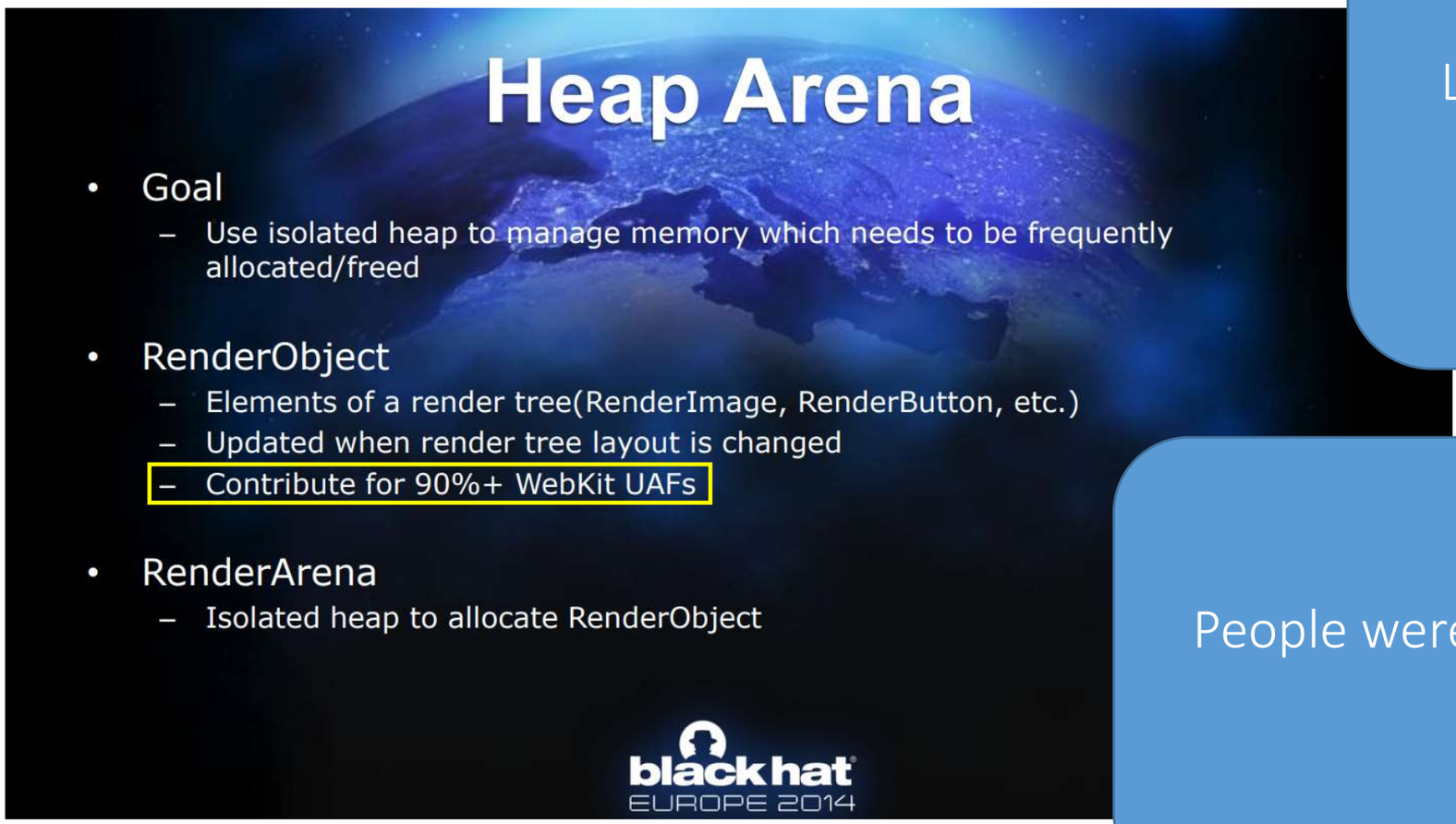
- Exploitation becomes very bug dependent
- Generic techniques that abuse heap metadata become less viable due to hardening from vendors
- Some attack surfaces become barren, either due to decreasing bug density or mitigations. Attackers shift to different attack surfaces
- SBX becomes more relevant (especially in android)

Browsers

Webkit Anonymous Frame Remote Code Execution Vulnerability						
ZDI-11-138	ZDI-CAN-1036	Apple	CVE-2011-0234	9.0	2011-04-19	2020-07-30
Webkit Undefined DOM Prototype Attach Remote Code Execution Vulnerability						
ZDI-11-135	ZDI-CAN-1168	WebKit	CVE-2011-1344	9.0	2011-04-14	
(Pwn2Own) WebKit WBR Tag Removal Remote Code Execution Vulnerability						
ZDI-11-104	ZDI-CAN-1107	WebKit	CVE-2011-1290	9.0	2011-04-14	
(Pwn2Own) Webkit CSS Text Element Count Remote Code Execution Vulnerability						
ZDI-11-101	ZDI-CAN-918	Apple	CVE-2011-0154	9.0	2011-03-02	
Apple iPhone Webkit Library Javascript Array sort Method Remote Code Execution Vulnerabi						
ZDI-11-100	ZDI-CAN-969	Apple	CVE-2011-0149			
Apple Webkit Root HTMLBRElement Style Remote Code Execution Vulnerability						
ZDI-11-099	ZDI-CAN-968	Apple	CVE-2011-0133			
Apple Webkit Font Glyph Layout Remote Code Execution Vulnerability						
ZDI-11-098	ZDI-CAN-987	Apple	CVE-2011-0132			
Apple Safari Webkit Runin Box Promotion Remote Code Execution Vulnerability						

People were pounding hard on the DOM…

Browsers



Let’s do something about the DOM…

People were pounding hard on the DOM…

Bring idea from Internet Explorer

The Isolated Heap in Internet Explorer

Most of the DOM objects and supporting objects are now allocated on a separate heap. This will prevent an attacker from easily allocating arbitrary data in the space of a freed DOM object. The separate heap is allocated during the initialisation phase of mshtml.dll:

```
; START OF FUNCTION CHUNK FOR __DllMainStartup@12

loc_63C94EA9: ; dwMaximumSize
push 0
push 0 ; dwInitialSize
push 0 ; flOptions
call ds:__imp__HeapCreate@12 ; HeapCreate(x,x,x)
mov _g_hIsolatedHeap, eax
```

DATAFLOW SECURITY

Webkit isolated heap

IsoHeap

IsoHeap is another new memory allocation API, every object allocated using IsoHeap will be allocated in dedicated memory pages. This stops objects of another type being allocated in the same memory, helping to prevent to exploitation of UAFs by preventing attackers from turning them into Type Confusion issues. This feature is explained pretty well in this commit.

In order to use IsoHeap, the WTF_MAKE_ISO_AL
added to each protected object, this is defined in I
changes the define based on whether IsoHeap is e
the macro expands to the code defined here and

Did this kill DOM exploits in webkit?

Maybe not entirely. (i.e. Isoheap bypass, non isolated objects, etc…)

Safari

==Phrack Inc.==

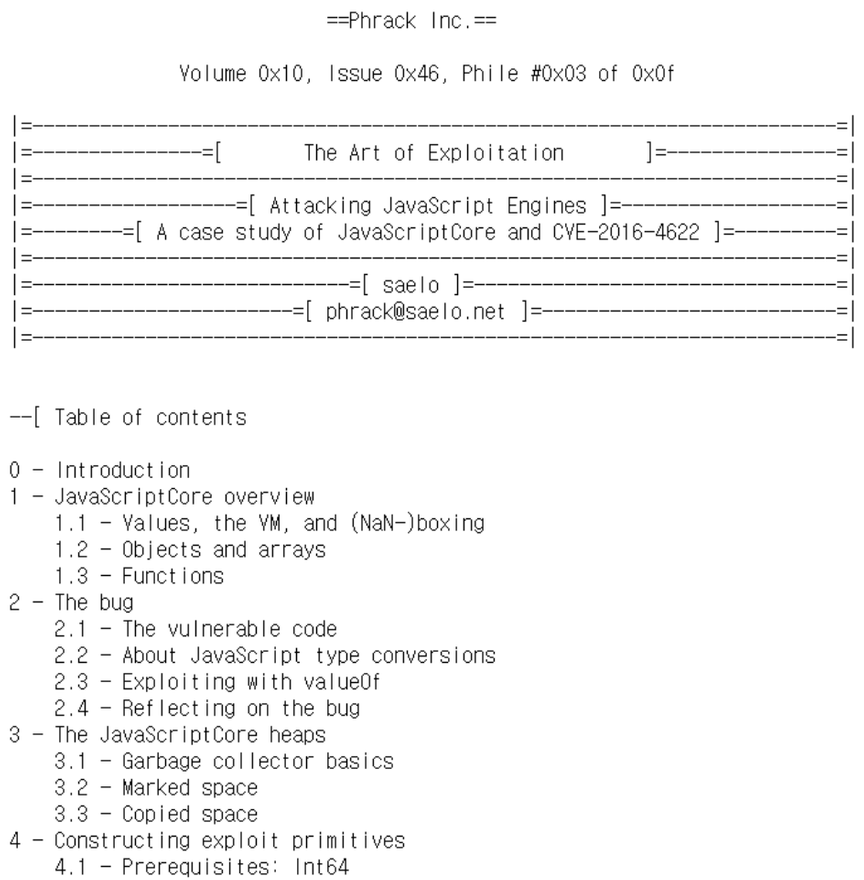
Volume 0x10, Issue 0x46, Phile #0x03 of 0x0f

|=---=|
|=----------------=[The Art of Exploitation]=----------------=|
|=---=|
|=-------------------=[Attacking JavaScript Engines]=--------------------=|
|=--------=[A case study of JavaScriptCore and CVE-2016-4622]=---------=|
|=---=|
|=---------------------------=[saelo]=-----------------------------=|
|=-----------------------=[phrack@saelo.net]=-------------------------=|
|=---=|

--[Table of contents

DATAFLOW SECURITY

Safari



==Phrack Inc.==

Volume 0x10, Issue 0x46, Phile #0x03 of 0x0f

|=---=|
|=----------------=[The Art of Exploitation]=----------------=|
|=---=|
|=-------------------=[Attacking JavaScript Engines]=--------------------=|
|=---------=[A case study of JavaScriptCore and CVE-2016-4622]=---------=|
|=---=|
|=----------------------------=[saelo]=-----------------------------=|
|=-----------------------=[phrack@saelo.net]=-------------------------=|
|=---=|

--[Table of contents

0 - Introduction
1 - JavaScriptCore overview
    1.1 - Values, the VM, and (NaN-)boxing
    1.2 - Objects and arrays
    1.3 - Functions
2 - The bug
    2.1 - The vulnerable code
    2.2 - About JavaScript type conversions
    2.3 - Exploiting with valueOf
    2.4 - Reflecting on the bug
3 - The JavaScriptCore heaps
    3.1 - Garbage collector basics
    3.2 - Marked space
    3.3 - Copied space
4 - Constructing exploit primitives
    4.1 - Prerequisites: Int64

Seminal whitepaper released by Samuel GroB, 2016

I feel like there was a profound increase of interest in JS engines among researchers after this article dropped

DATAFLOW SECURITY

Javascript becomes prime target for RCE

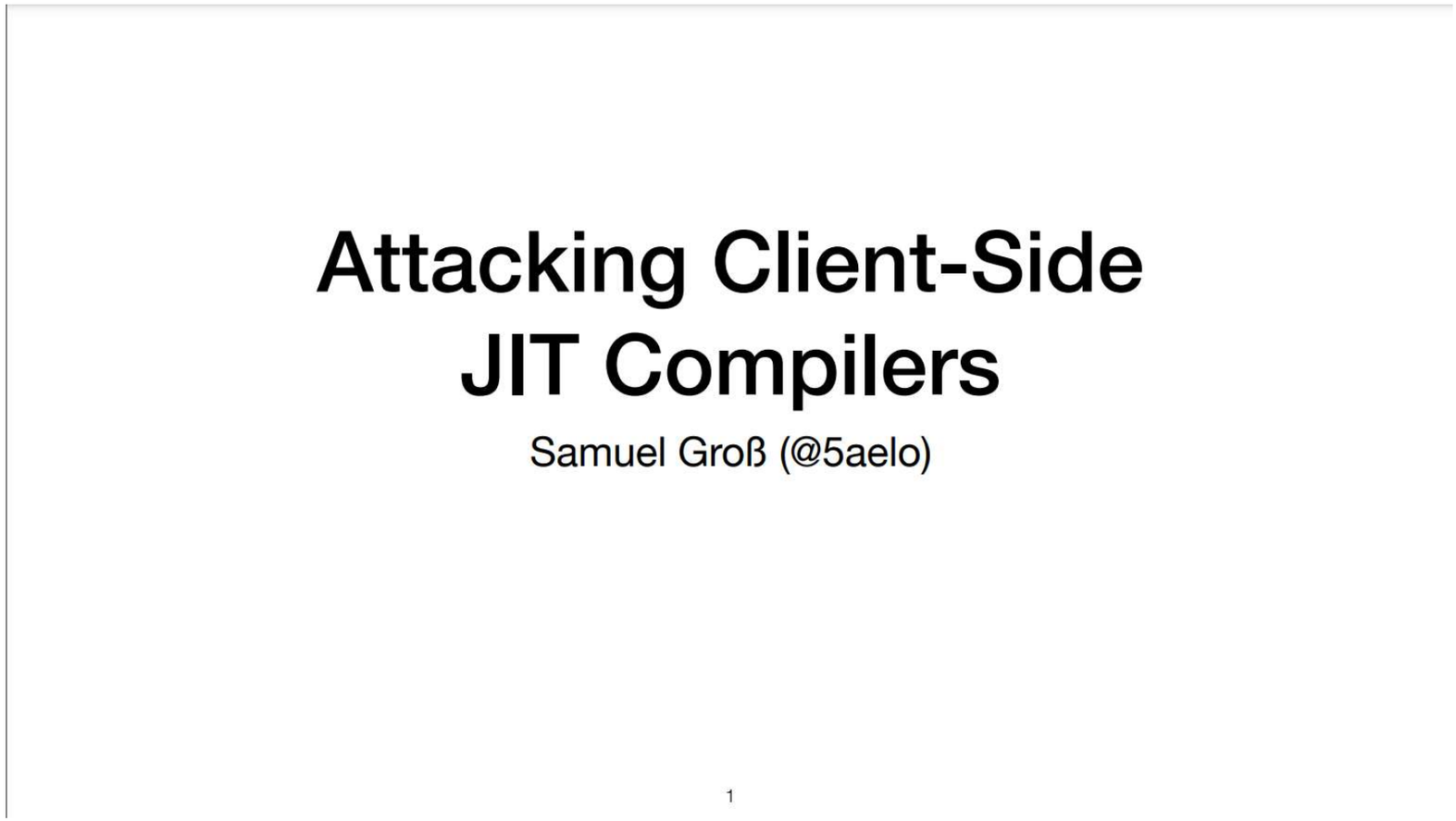
Every year some of the greatest security researchers around the globe gather together for the Pwn2Own event to demonstrate their skills by compromising widely used applications. This year's event recently completed and did not disappoint. On the second day of the competition, the Fluoroacetate team and Niklas Baumstark successfully demonstrated attacks against Mozilla Firefox, earning themselves $50,000 USD and $40,000 USD respectively. Now that Mozilla has patched the underlying bugs, I will provide you with details of the two vulnerabilities used by these two teams to compromise the renderer process of Mozilla Firefox.

DATAFLOW SECURITY

Early Browser RCE

- Re-entrency bugs (concept comes from Adobe flash, other browsers). Will be re-introduced every once in a while because of new features implemented (Proxy, Species, Regex, etc…). Very popular target
- Edge cases in weird Javascript constructs (Integer overflow, UAF, uninitialized variable, …)
- Javascript API's (WebSQL, WebAudio, …)
- WebAssembly
- And more…

JIT bugs become mainstream



DATAFLOW SECURITY

JIT bugs become mainstream

Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JavascriptGeneratorFunction::GetPropertyBuiltIns exposes scriptFunction CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Incorrect scope handling CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Deferred parsing makes wrong scopes #2 CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: stack-to-heap copy bug CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Loop analysis bug CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: AsmJSByteCodeGenerator::EmitCall call handling bug CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Incomplete fix for issue 1365 CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Escape analysis bug #2 CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: ImplicitCallFlags checks bypass CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: CallRegExSymbolFunction doesn't check the return type CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Array type confusion via InitProto instructions CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Array type confusion via Array.prototype.reverse CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Array type confusion via NewScObjectNoCtor CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: LdThis type confusion CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: The fix for issue 1420 is incomplete. CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: The fix for issue 1420 is incomplete #2 CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: A bound check elimination bug CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Magic value can cause type confusion CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Cross context bug CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: EntrySimpleObjectSlotGetter can have side effects CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: Type confusion with hoisted SetConcatStrMultiItemBE instructions CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: OOB reads/writes CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: JIT: ImplicitCallFlags check bypass with Intl CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: A bug in BoundFunction::NewInstance CCProjectZeroMembers
Microsoft	Edge	lokihardt	Microsoft Edge: Chakra: Parameter scope parsing bug CCProjectZeroMembers

Common techniques for JIT exploitation appears (eliminate bounds check)

Issue 762874: Security: off by one in TurboFan range optimization for String.indexOf

Reported by sroet...@google.com on Thu, Sep 7, 2017, 6:44 PM GMT+9 **Project Member**

VULNERABILITY DETAILS

The typer returns a range from -1 to String::kMaxLength - 1 for String.indexOf:

https://cs.chromium.org/chromium/src/v8/src/compiler/typer.cc?rcl=8cd4009c5b7072ad224f19a9e668ec0ed7430599&l=1456

However, indexOf can actually return String::kMaxLength (2**28-16):

'A'.repeat(2**28-16).indexOf("", 2**28).toString(16)

Since the optimizer is confused about the potential return values, you can use it to optimize away array accesses and get an out of bounds read-write primitive.

The fix should simply be to remove the - 1 in that line.

VERSION

Chrome Version: 60.0.3112.113 + stable

Operating System: Ubuntu 17.04

REPRODUCTION CASE

Here's how you trigger an out of bounds read. I can also share a full exploit to UXSS/code exec if you're interested.

DATAFLOW SECURITY

v8 killing common exploit primitives

Issue 8806: Harden turbofan's bounds check against typer bugs
Reported by jarin@google.com on Thu, Feb 7, 2019, 11:52 PM GMT+9 **Project Member**

Rather than eliminating bounds checks, just abort if they do not pass.

Comment 1 by bugdroid on Sat, Feb 9, 2019, 1:15 AM GMT+9 **Project Member**

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8.git/+/7bb6dc0e06fa158df508bc8997f0fce4e33512a5

commit 7bb6dc0e06fa158df508bc8997f0fce4e33512a5
Author: Jaroslav Sevcik <jarin@chromium.org>
Date: Fri Feb 08 16:14:23 2019

[turbofan] Introduce aborting bounds checks.

Instead of eliminating bounds checks based on types, we introduce
an aborting bounds check that crashes rather than deopts.

~~Bug: v8:8806~~
Change-Id: Icbd9c4554b6ad20fe4135b8622590093679dac3f
Reviewed-on: https://chromium-review.googlesource.com/c/1460461
Commit-Queue: Jaroslav Sevcik <jarin@chromium.org>
Reviewed-by: Tobias Tebbi <tebbi@chromium.org>

DATAFLOW SECURITY

On the webkit side…
(Pointer poisoning, Butterfly gigacage check)

Timestamp: Jan 8, 2018 1:05:17 PM (5 years ago)
Author: mark.lam@apple.com

Message: Apply poisoning to more pointers in JSC.
https://bugs.webkit.org/show_bug.cgi?id=181096
<rdar://problem/36182970>

Reviewed by JF Bastien.

Source/JavaScriptCore:

- assembler/MacroAssembler.h:

(JSC::MacroAssembler::xorPtr):

- assembler/MacroAssemblerARM64.h:

(JSC::MacroAssemblerARM64::xor64):

- assembler/MacroAssemblerX86_64.h:

(JSC::MacroAssemblerX86_64::xor64):

- Add xorPtr implementation.

- bytecode/CodeBlock.cpp:

Changeset 220165 in webkit

Timestamp: Aug 2, 2017 6:32:07 PM (6 years ago)
Author: fpizlo@apple.com

Message: All C++ accesses to JSObject::m_butterfly should do caging
https://bugs.webkit.org/show_bug.cgi?id=175039

Reviewed by Keith Miller.

Source/JavaScriptCore:

Makes JSObject::m_butterfly a AuxiliaryBarrier<CagedPtr<Butter
This ensures that you can't cause C++ code to access a butterfly
outside the gigacage.

- runtime/JSArray.cpp:

(JSC::JSArray::setLength):
(JSC::JSArray::pop):
(JSC::JSArray::push):
(JSC::JSArray::shiftCountWithAnyIndexingType):
(JSC::JSArray::unshiftCountWithAnyIndexingType):
(JSC::JSArray::fillArgList):
(JSC::JSArray::copyToArguments):

- runtime/JSObject.cpp:

(JSC::JSObject::heapSnapshot):

DATAFLOW SECURITY

On the webkit side…
(Pointer poisoning, Butterfly gigacage check)

Timestamp: Jan 8, 2018 1:05:17 PM (5 years ago)
Author: mark.lam@apple.com

Message: Apply poisoning to more pointers in JSC.
https://bugs.webkit.org/show_bug.cgi?id=181096
<rdar://problem/36182970>

Reviewed by JF Bastien.

Source/JavaScriptCore:

- assembler/MacroAssembler.h:

(JSC::MacroAssembler::xorPtr):

- assembler/MacroAssemblerARM64.h:

(JSC::MacroAssemblerARM64::xor64):

- assembler/MacroAssemblerX86_64.h:

(JSC::MacroAssemblerX86_64::xor64):

- Add xorPtr implementation.
- bytecode/CodeBlock.cpp:

They were both removed…??!!

Changeset 220165 in webkit

Timestamp: Aug 2, 2017 6:32:07 PM (6 years ago)
Author: fpizlo@apple.com

Message: All C++ accesses to JSObject::m_butterfly should do caging
https://bugs.webkit.org/show_bug.cgi?id=175039

Reviewed by Keith Miller.

Source/JavaScriptCore:

Makes JSObject::m_butterfly a AuxiliaryBarrier<CagedPtr<Butter
This ensures that you can't cause C++ code to access a butterfly
outside the gigacage.

- runtime/JSArray.cpp:

(JSC::JSArray::setLength):
(JSC::JSArray::pop):
(JSC::JSArray::push):
(JSC::JSArray::shiftCountWithAnyIndexingType):
(JSC::JSArray::unshiftCountWithAnyIndexingType):
(JSC::JSArray::fillArgList):
(JSC::JSArray::copyToArguments):

- runtime/JSObject.cpp:

DATAFLOW SECURITY

Webkit - common object faking technique

```
----[ 6.1 - Predicting structure IDs

Unfortunately, structure IDs aren't necessarily static across different
runs as they are allocated at runtime when required. Further, the IDs of
structures created during engine startup are version dependent. As such we
don't know the structure ID of a Float64Array instance and will need to
determine it somehow.

Another slight complication arises since we cannot use arbitrary structure
IDs. This is because there are also structures allocated for other garbage
collected cells that are not JavaScript objects (strings, symbols, regular
expression objects, even structures themselves). Calling any method
referenced by their method table will lead to a crash due to a failed
assertion. These structures are only allocated at engine startup though,
resulting in all of them having fairly low IDs.

To overcome this problem we will make use of a simple spraying approach: we
will spray a few thousand structures that all describe Float64Array
instances, then pick a high initial ID and see if we've hit a correct one.
```

```
for (var i = 0; i < 0x1000; i++) {
    var a = new Float64Array(1);
    // Add a new property to create a new Structure instance.
    a[randomString()] = 1337;
}
```

DATAFLOW SECURITY

Webkit’s response - StructureID randomization

[Re-landing] Add some randomness into the StructureID.

https://bugs.webkit.org/show_bug.cgi?id=194989
<rdar://problem/47975563>

Reviewed by Yusuke Suzuki.

1. On 64-bit, the StructureID will now be encoded as:

1 Nuke Bit	24 StructureIDTable index bits	7 entropy bits

   The entropy bits are chosen at random and assigned when a StructureID is allocated.

2. Instead of Structure pointers, the StructureIDTable will now contain encodedStructureBits, which is encoded as such:

7 entropy bits	57 structure pointer bits

DATAFLOW SECURITY

Webkit’s response - StructureID randomization

[Re-landing] Add some randomness into the StructureID.

https://bugs.webkit.org/show_bug.cgi?id=194989
<rdar://problem/47975563>

Reviewed by Yusuke Suzuki.

1. On 64-bit, the StructureID will now be encoded as:

1 Nuke Bit	24 StructureIDTable index bits	7 entropy bits

   The entropy bits are chosen at random and assigned when a StructureID i
   allocated.

2. Instead of Structure pointers, the StructureIDTable will now contain
   encodedStructureBits, which is encoded as such:

7 entropy bits	57 structure pointer bits

Bypassing was not hard, multiple bypasses published in the public

DATAFLOW SECURITY

PAC lands in late 2018

PAC lands in late 2018

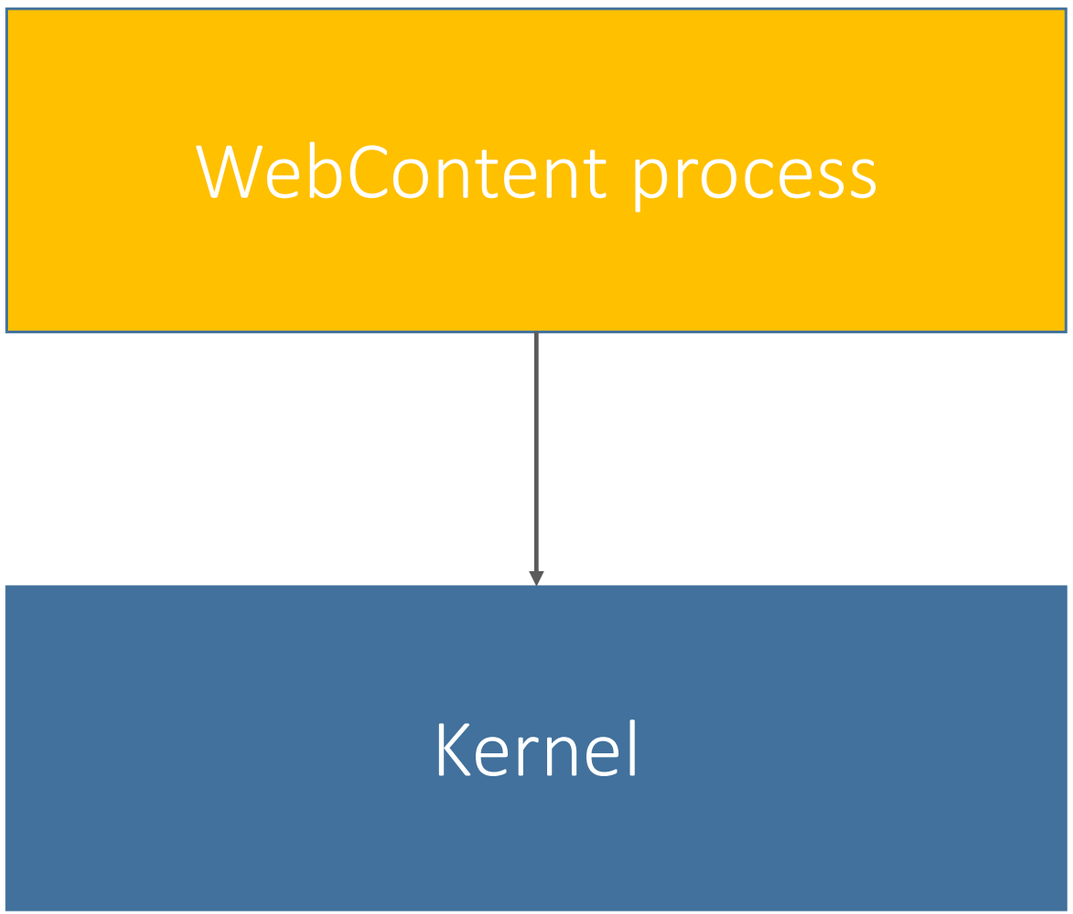
Great impact on bugs and exploitation. All virtual function calls are authenticated.

Most UAF bugs are dead

JOP'ing significantly harder (without a PAC bypass)

DATAFLOW SECURITY

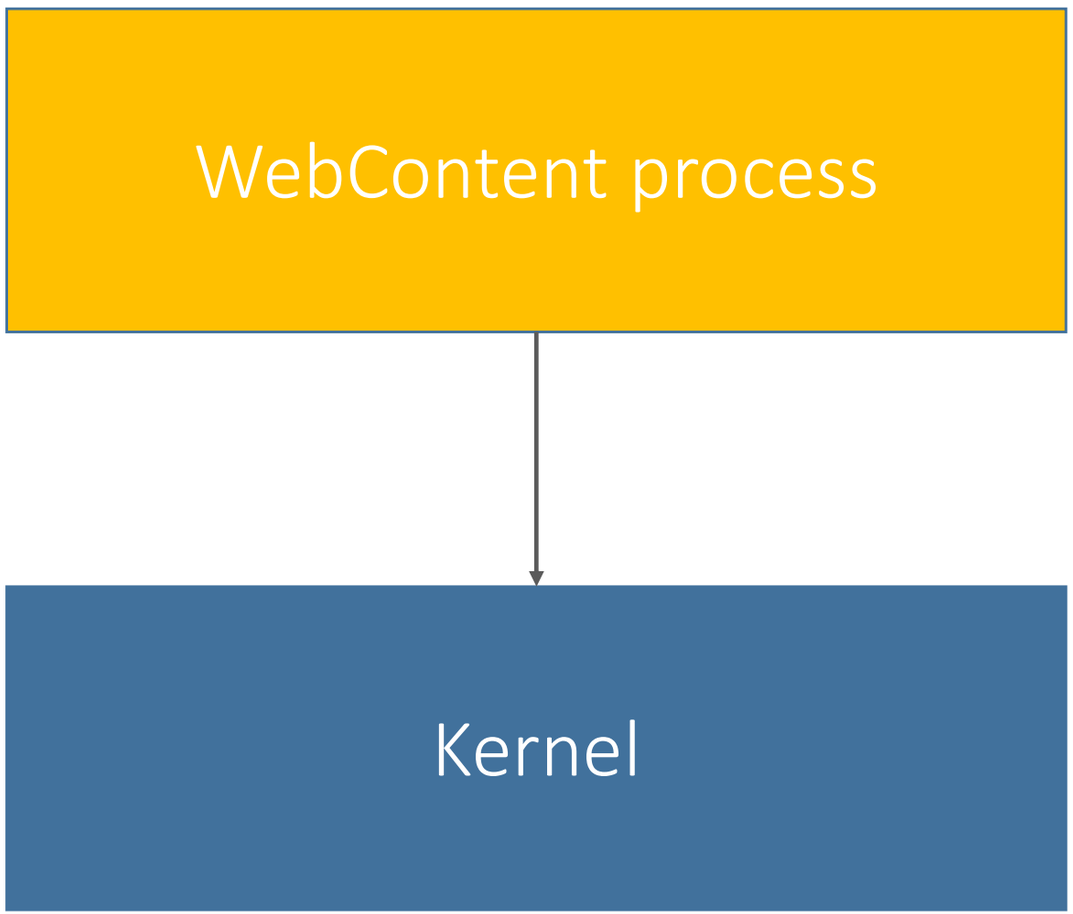
iOS Full chain



Before :

- Webkit exploit
- Execute Shellcode
- Kernel exploit

iOS Full chain

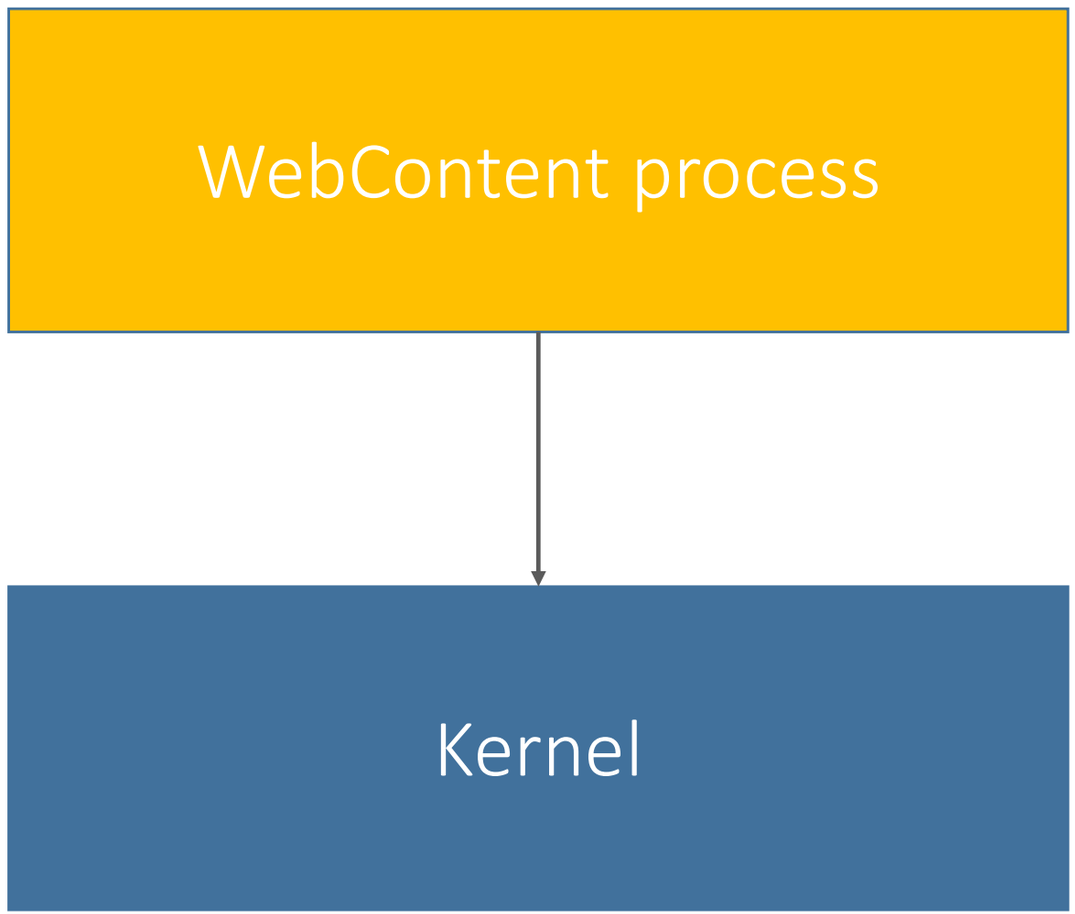


After PAC :

- Webkit arbitrary R/W
- Userland PAC bypass (function call primitive)
- Execute Shellcode
- Kernel arbitrary R/W
- Kernel PAC bypass

DATAFLOW SECURITY

iOS Full chain



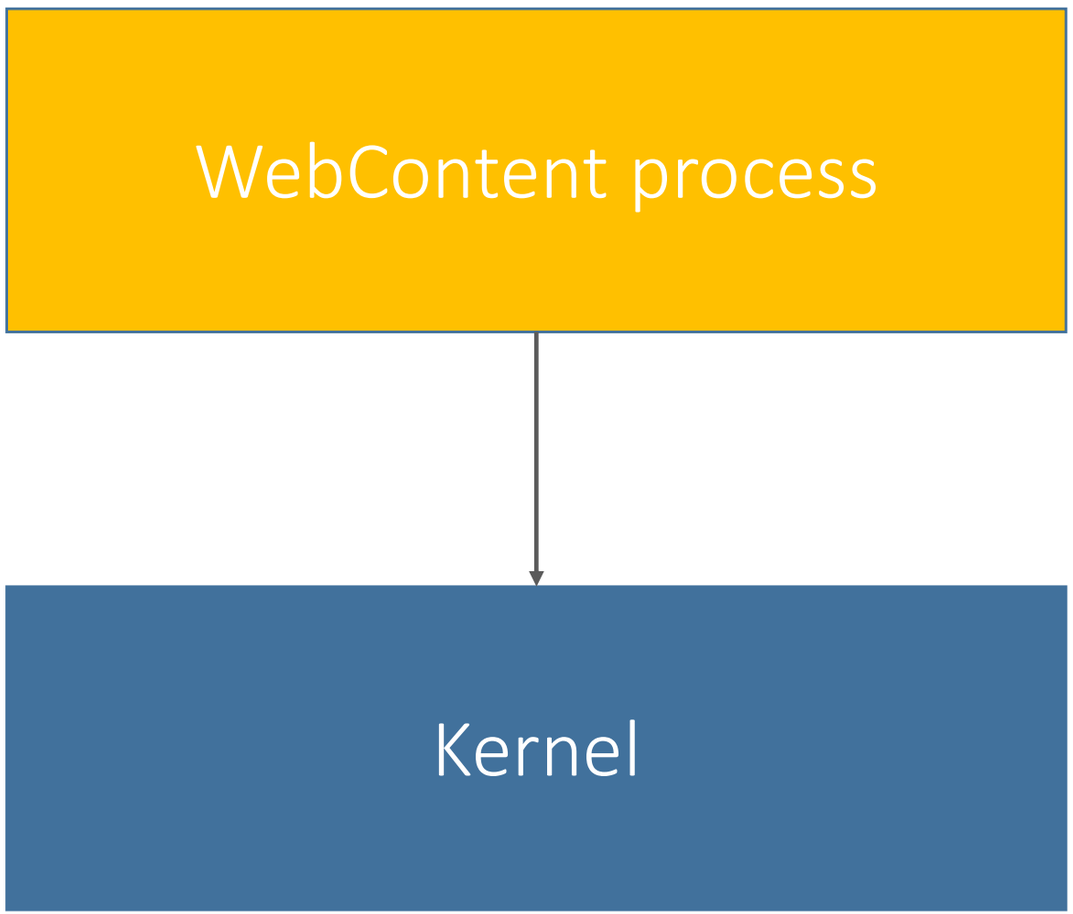
After PAC :

- Webkit arbitrary R/W
- Userland PAC bypass (function call primitive)
- Execute Shellcode
- Kernel arbitrary R/W
- Kernel PAC bypass

De-facto for PAC bypass becomes :

- First gain AAR/AAW
- Work forwards from there for PAC bypass

iOS Full chain

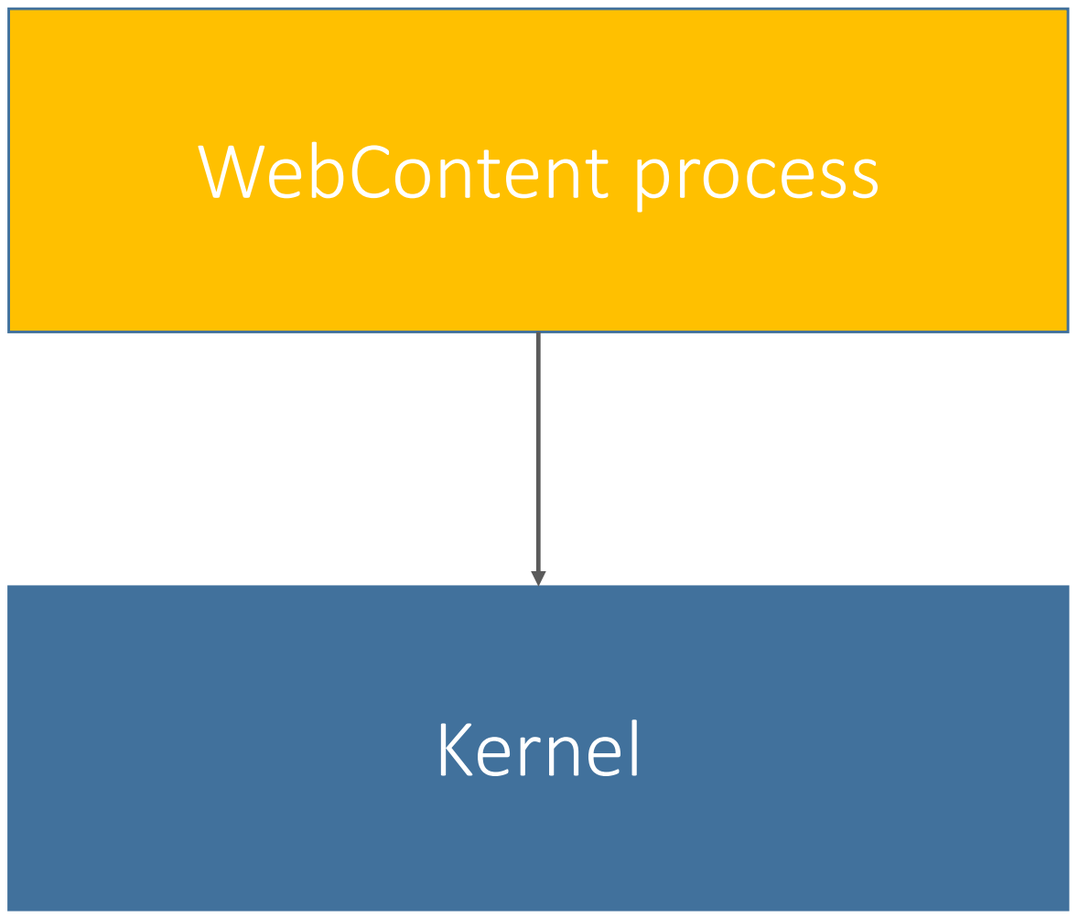


After PAC :

- Webkit arbitrary R/W
- Userland PAC bypass (function call primitive)
- Execute Shellcode
- Kernel arbitrary R/W
- Kernel PAC bypass

Do you realize that iOS browser chains never really required an SBX…??
→ Why even bother with SBX, go straight to the kernel with a kernel bug

iOS Full chain



After PAC :

- Webkit arbitrary R/W
- Userland PAC bypass (function call primitive)
- Execute Shellcode
- Kernel arbitrary R/W
- Kernel PAC bypass

 obviously realized this. They wanted to make it harder to go from
Webcontent → directly to Kernel

Tighten the Sandbox profile (2021 ~ 2022)

Bug 221959 - [iOS] Remove access to the Mobile asset service

Status: RESOLVED FIXED

Alias: None

Bug 221967 - [macOS] Remove access to IOAccelerationUserClient

Status: RESOLVED FIXED

Alias: None

Bug 221946 - Remove unneeded sandbox access to some file paths

Status: RESOLVED FIXED

Alias: None

Product: WebKit

Bug 221610 - [macOS] Deny mach-lookup to the fonts service

Status: RESOLVED FIXED

Alias: None

Bug 221458 - [iOS] Remove access to AppleJPEGDriverUserClient

Status: RESOLVED FIXED

Alias: None

DATAFLOW SECURITY

Tighten the Sandbox profile (2021 ~ 2022)

Bug 221959 - [iOS] Remove access to the Mobile asset service

Status: RESOLVED FIXED

Alias: None

Bug 221967 - [macOS] Remove access to IOAccelerationUserClient

Status: RESOLVED FIXED

Alias: None

Bug 221946 - Remove unneeded sandbox access to some file paths

Status: RESOLVED FIXED

Alias: None

Product: WebKit

Bug 221610

Bug 221458 - [iOS] Remove access to AppleJPEGDriverUserClien

Status: RESOLVED FIXED

Alias: None

 gradually removes access to everything that's unnecessary for Webcontent, to the bare minimum. Some of them are just moved to other processes (GPU process)

iokit method number filters

You can only call iokit methods that the sandbox profile allows

```
(allow iokit-open
    (require-all
        (extension "com.apple.webkit.extension.iokit")
        (iokit-user-client-class "AGXDeviceUserClient") ;; Used by WebGL
    )
    (when (defined? 'iokit-external-method)
        (apply-message-filter
            (deny (with telemetry)
                iokit-external-trap)
            (deny (with telemetry) (with message "AGXDeviceUserClient")
                iokit-async-external-method
                iokit-external-method
            )
            (allow iokit-async-external-method
                (iokit-method-number
                    43
                )
            )
            (allow iokit-external-method
                (iokit-method-number
                    0
                    2
                    4
                    5
                    6
                    7
                    8
                    9
                    10
                    11
                    12
                    13
                    14
                    15
                    16
                    25
                    26
                    27
                    36
                    38
                    44
```

mach_msg “msgh_id filters”

```
(allow mach-lookup
    (global-name "com.apple.system.notification_center")
    (apply-message-filter
        (deny mach-message-send)
        (deny mach-message-send (with no-report) (message-number 1023))
        (allow mach-message-send (message-number
            1002 1011 1012 1016 1017 1018 1021 1025 1026 1028 1029 1030 1031 1032))))
```

You can only send a mach message with a certain msgh_id

(They seemed to also introduce xpc message filtering briefly…? However, it’s no longer there)

DATAFLOW SECURITY

KTRR/APRR/SPRR, AMCC, PPL, GFX

- Early KPP/KTRR bypassed by racing, unprotected MSR register (Pangu, Siguza, Luca, Ian Beer)
- Supplemented by hardware based mitigations (AMCC / PPL). Protects modification of Trust cache, Kernel .text .const, Page tables, MMIO…
- APRR supplemented again by SPRR, GFX
- Described in great detail in Siguza, Jonathan Levin, Sven Peter's articles
- (Very abbreviated) General exploit flow :

Find Kernel PAC bypass → Arbitrary function call primitive → Find PPL bypass → Ultimately try to gain control over page tables

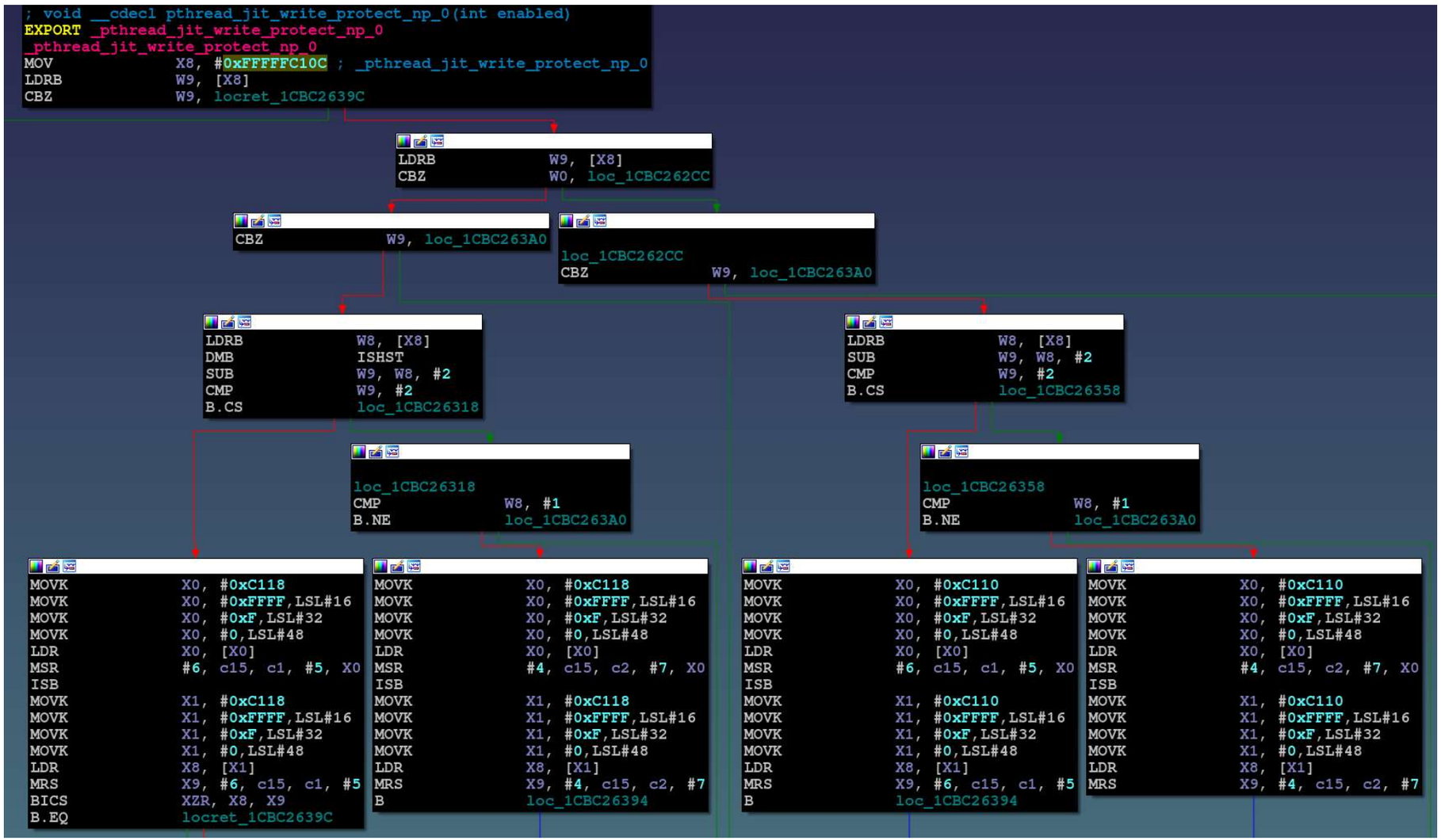
Userland APRR (JIT region)

```
int readPtr = 0;
int writePtr = 0;
unsigned jumpCount = jumpsToLink.size();

if (useFastJITPermissions())
    threadSelfRestrictRWXToRW();

if (m_shouldPerformBranchCompaction) {
    for (unsigned i = 0; i < jumpCount; ++i) {
```

```
static ALWAYS_INLINE void threadSelfRestrictRWXToRW()
{
    ASSERT(useFastJITPermissions());

#if USE(PTHREAD_JIT_PERMISSIONS_API)
    pthread_jit_write_protect_np(false);
#elif USE(APPLE_INTERNAL_SDK)
    os_thread_self_restrict_rwx_to_rw();
#else
    bool tautologyToIgnoreWarning = true;
    if (tautologyToIgnoreWarning)
        RELEASE_ASSERT_NOT_REACHED();
#endif
}
```

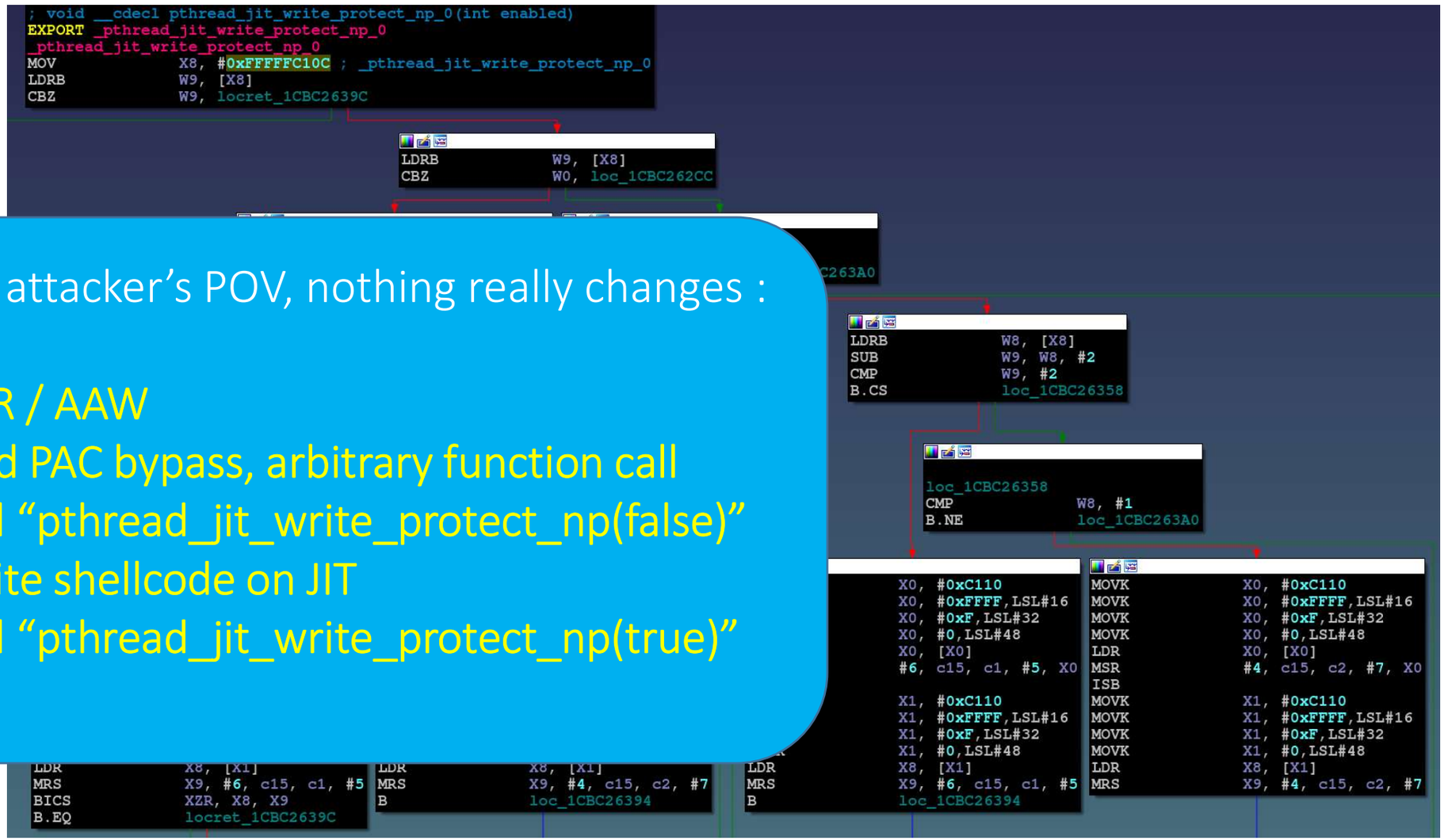
DATAFLOW SECURITY

Userland APRR (JIT region)



DATAFLOW SECURITY

Userland APRR (JIT region)



In the attacker's POV, nothing really changes :

1. AAR / AAW
2. Find PAC bypass, arbitrary function call
3. Call "pthread_jit_write_protect_np(false)"
4. Write shellcode on JIT
5. Call "pthread_jit_write_protect_np(true)"

DATAFLOW SECURITY

Safari and Kernel PAC bypasses

iOS kernel PAC bypasses

	iOS 12	iOS 13
PAC signing gadget	1	1
PAC bruteforce gadget	1	
Thread state signing gadget	2	2
Unprotected indirect branch	1	
Implementation bug	1	
Reusing signed states (design issue)		2

Issue 2044: PAC bypass due to unprotected function pointer imports
Reported by saelo@google.com on Wed, May 20, 2020, 8:59 PM GMT+9

PAC [1] aims to prevent an attacker with the ability to read and write memory from executir validating code pointers (as well as some data pointers) at runtime. However, it seems that i properly protected by PAC, allowing an attacker to sign arbitrary pointers and thus bypass P for ~~issue #2042~~, thus I'm reporting it via our tracker.

llowing code from WebKit:

ıt::doublePow(LValue xOperand, LValue yOperand)

powDouble)(double, double) = pow;
lWithoutSideEffects(B3::Double, powDouble, xOperand, yOperand);

Comment 1 by saelo@google.com on Tue
There is a potential variant of this bypass:

WebKit's pointer tagging infrastructure (im attacker. It uses different tags as PAC discri tag (e.g. due to memory corruption), it call

```
void reportBadTag(const void* ptr, PtrTag expectedTag)
{
    dataLog("PtrTag ASSERTION FAILED on pointer ", RawPointer(ptr), ", actual tag = ", tagForPtr(ptr));
    ...
}
```

Due to the dataLog, tagForPtr [5] is called, which ends up traversing a linked list:

d JIT Hardening Bypass in WebKit on iOS
ogle.com on Fri, May 15, 2020, 9:39 PM GMT+9

product-security@apple.com, Apple would like to treat the PAC bypass described here as a ut a deadline on May 6. After receiving the reply that they will treat it as a separate vulnera lay).

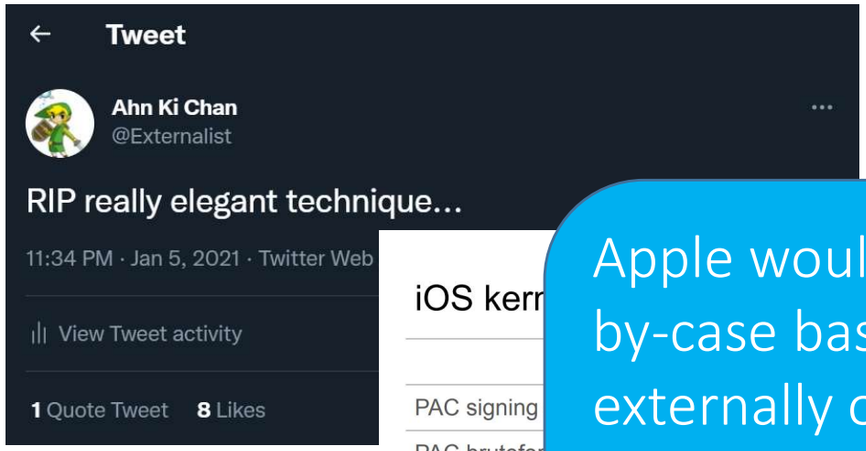
ellcode execution from arbitrary memory read/write in a WebKit renderer only requires findi a combination of APRR [1] and PAC [2] protect the JIT region from an attacker with arbitrary read/write.

WebKit has support for in-process signal handling. This is for example used by some JIT optimizations in JSC [3]. T `catch_mach_exception_raise_state` in Signals.cpp [4], which will traverse a linked list of handlers and call each one the signal is treated as handled and the thread will continue.

This enables the following attack:

DATAFLOW SECURITY

Safari and Kernel PAC bypasses



iOS kern

PAC signing
PAC brutefor
Thread state
Unprotected
Implementati
Reusing sign

Apple would patch these in a case-by-case basis, either if it's reported externally or used in a Pwning competition
There weren't any further hardenings on the JIT mechanism itself

Issue 2044: PAC bypass due to unprotected function pointer imports
Reported by saelo@google.com on Wed, May 20, 2020, 8:59 PM GMT+9

PAC [1] aims to prevent an attacker with the ability to read and write memory from executir validating code pointers (as well as some data pointers) at runtime. However, it seems that i properly protected by PAC, allowing an attacker to sign arbitrary pointers and thus bypass P it via our tracker.

erand, LValue yOperand)

le) = pow;
uble, powDouble, xOperand, yOperand);

Comment 1 by saelo@google.com on Tue

There is a potential variant of this bypass:

WebKit's pointer tagging infrastructure (im attacker. It uses different tags as PAC discri tag (e.g. due to memory corruption), it call

```
void reportBadTag(const void* ptr, PtrTag expectedTag)
{
    dataLog("PtrTag ASSERTION FAILED on pointer ", RawPointer(ptr), ", actual tag = ", tagForPtr(ptr));
    ...
}
```

Due to the dataLog, tagForPtr [5] is called, which ends up traversing a linked list:

s in WebKit on iOS
020, 9:39 PM GMT+9

om, Apple would like to treat the PAC bypass described here as a ter receiving the reply that they will treat it as a separate vulnera

bitrary memory read/write in a WebKit renderer only requires findi ct the JIT region from an attacker with arbitrary read/write.

WebKit has support for in-process signal handling. This is for example used by some JIT optimizations in JSC [3]. T `catch_mach_exception_raise_state` in Signals.cpp [4], which will traverse a linked list of handlers and call each one the signal is treated as handled and the thread will continue.

This enables the following attack:

DATAFLOW SECURITY

More and more Data PAC on critical structures…

Changeset 269020 in webkit

Timestamp: Oct 26, 2020 7:13:14 PM (2 years ago)
Author: Tadeu Zagallo

Message: Sign MacroAssembler::jumpsToLink
https://bugs.webkit.org/show_bug.cgi?id=217774
<rdar://problem/69433058>

Reviewed by Saam Barati.

Source/JavaScriptCore:

Changeset 272191 in webkit

Timestamp: Feb 1, 2021 11:46:20 PM (2 years ago)
Author: sbarati@apple.com

Message: Sign m_offset in AssemblerLabel
https://bugs.webkit.org/show_bug.cgi?id=221237

Reviewed by Mark Lam.

- assembler/ARM64Assembler.h:

Changeset 269016 in webkit

Timestamp: Oct 26, 2020 6:56:16 PM (2 years ago)
Author: Tadeu Zagallo

Message: Validate addresses returned by LinkBuffer::locationOf
https://bugs.webkit.org/show_bug.cgi?id=217786
<rdar://problem/69887913>

Reviewed by Saam Barati.

- assembler/LinkBuffer.h:

(JSC::LinkBuffer::locationOf):
(JSC::LinkBuffer::locationOfNearCall):
(JSC::LinkBuffer::getLinkerAddress):

DATAFLOW SECURITY

Some early signs of JIT Cage

Changeset 269349 in webkit

Timestamp: Nov 3, 2020 6:31:56 PM (2 years ago)
Author: ysuzuki@apple.com

Message: [JSC] Add JITCage support
https://bugs.webkit.org/show_bug.cgi?id=218143

Reviewed by Saam Barati.

Source/JavaScriptCore:

Towards software verified JIT, this patch adds partial JIT-Caging support which cages JIT call / jumps in a certain format.
This is currently only enabled when internal SDK is enabled. And it is only enabled in ARM64E for now.
Currently, this patch does not have CSS JIT support. Subsequent patch will add it.
We ensured that JS2 and RAMification are neutral.

- CMakeLists.txt:
- JavaScriptCore.xcodeproj/project.pbxproj:
- assembler/JITOperationList.cpp:

(JSC::addPointers):
(JSC::JITOperationList::populatePointersInJavaScriptCoreForLLInt):

- assembler/JITOperationList.h:

(JSC::JITOperationList::map const):
(JSC::JITOperationList::assertIsHostFunction):
(JSC::JITOperationList::assertIsJITOperation):
(JSC::JITOperationList::contains const): Deleted.

DATAFLOW SECURITY

Some early signs of ISA pointer PAC (late 2020)

(Upcoming?) ObjectiveC ISA PAC

The PoC exploit against iMessage on iOS 12.4 relied heavily on faking ObjectiveC objects to gain a form of arbitrary code execution despite the presence of pointer authentication (PAC). This was mainly possible because the ISA field, containing the pointer to the Class object and thus making a piece of memory appear like a valid ObjectiveC object, was not protected through PAC and could thus be faked. With iOS 14, this now seems to be changing: while previously, the top 19 bits of the ISA value contained the inline refcount, it now appears that this field has been reduced to 9 bits (of which the LSB appears to be reserved for some purpose, leaving an 8-bit inline refcount, see the bit shifting logic in `objc_release` or `objc_retain`), while the freed-up bits now hold a PAC, as can be seen in `objc_rootAllocWithZone` in libobjc.dylib:

```
; Allocate the object
BL              j__calloc_3
CBZ             X0, loc_1953DA434
MOV             X8, X0
; “Tag” the address with a constant to get a PAC modifier value
MOVK            X8, #0x6AE1,LSL#48
MOV             X9, X19
; Compute PAC of Class pointer with tagged object address as modifier
PACDA           X9, X8
; Clear top 9 bits (inline refcnt) and bottom 3 bits (other bitfields)
AND             X8, X9, #0x7FFFFFFFFFFFF8
; Set LSB and inline refcount to one
MOV             X9, #0x100000000000001
ORR             X9, X8, X9
; Presumably, the refcnt isn’t used for all types of classes...
TST             W20, #0x2000
CSEL            X8, X9, X8, EQ
; Store the resulting value into the ISA field
STR             X8, [X0]
```

DATAFLOW SECURITY

Separate IA keys (Early 2021)

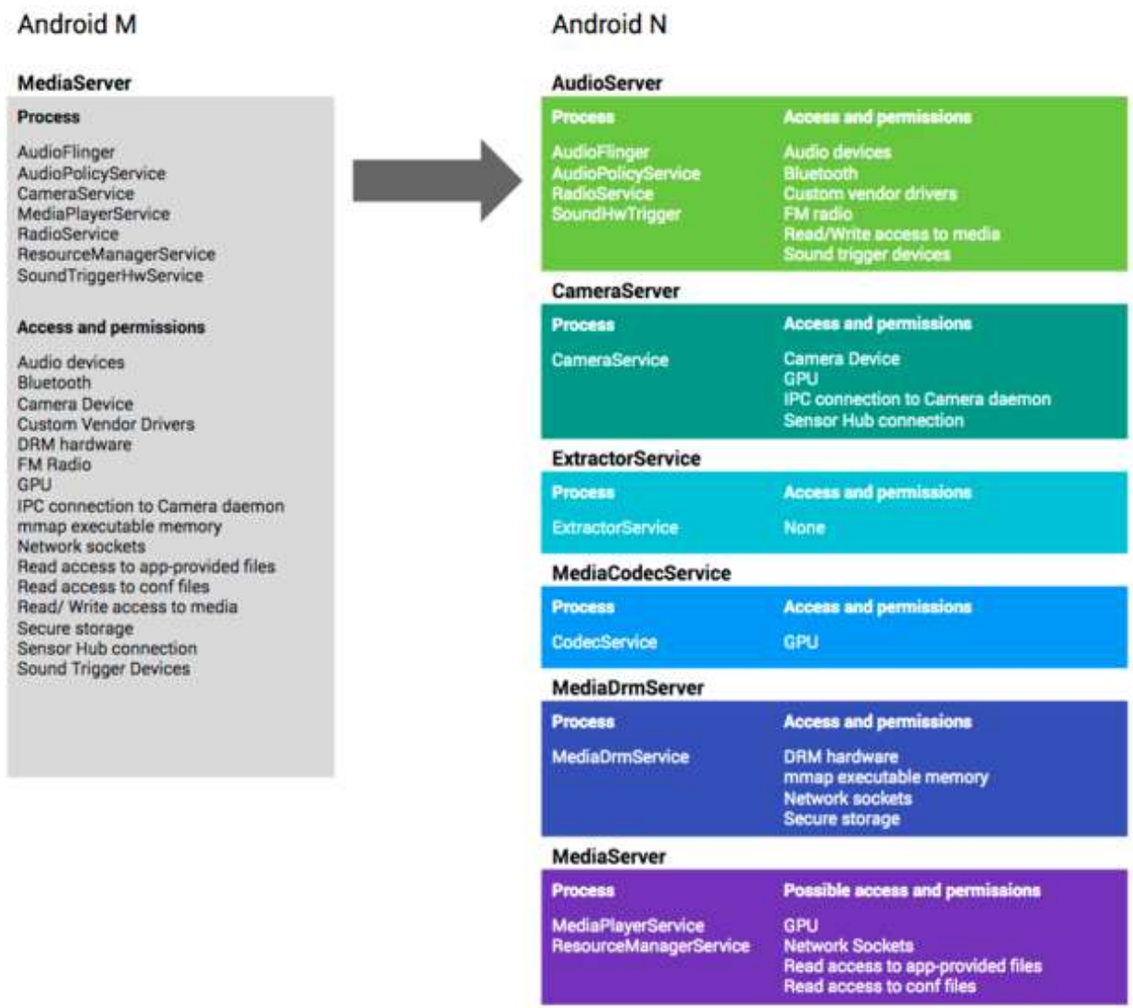
- Different IA keys based on Team ID
- WebContent uses separate IA keys now. Can’t sign arbitrary instruction pointers in other processes (Wasn’t really an issue because exploit would directly target the kernel)
- Will probably expand to more “usual suspect processes”

Now a brief detour to Android…

Some environment changes

- dlmalloc → jemalloc
- 64 bit devices
- 64 bit Chrome (Early 2021)
- Maintaining both 32-bit, 64-bit becomes a headache… :(

The end of Stagefright era (Mediaserver)



SEAndroid (isolated_app)

```
###
### Services with isolatedProcess=true in their manifest.
###
### This file defines the rules for isolated apps. An "isolated
### app" is an APP with UID between AID_ISOLATED_START (99000)
### and AID_ISOLATED_END (99999).
###

typeattribute isolated_app coredomain;

app_domain(isolated_app)

# Access already open app data files received over Binder or local socket IPC
allow isolated_app { app_data_file privapp_data_file sdk_sandbox_data_file}:f

# Allow access to network sockets received over IPC. New socket creation is n
# permitted.
allow isolated_app { ephemeral_app priv_app untrusted_app_all }:{ tcp_socket

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

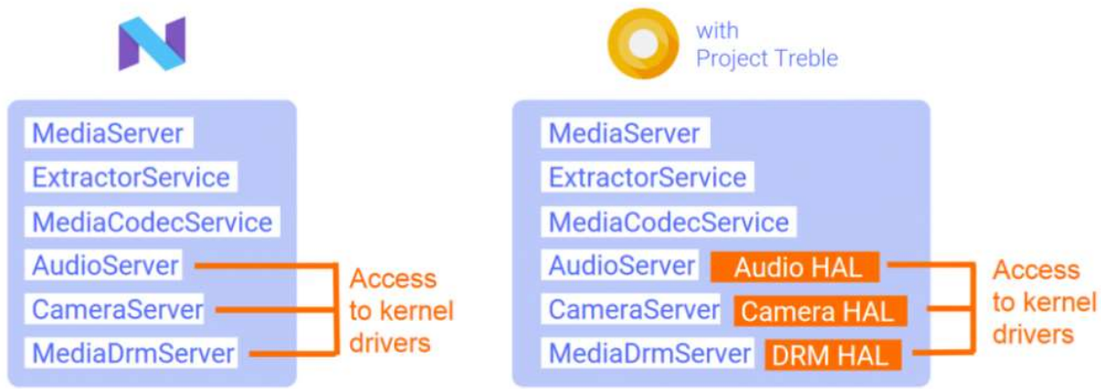
Extremely constrained.
Can barely access anything from “isolated_app” SEAndroid context (Chrome renderer)

While Apple was busy building a castle on top of hardware based mitigations, Google put a lot of effort on proper sandboxing/isolation

DATAFLOW SECURITY

Project Treble – More and more sandboxing

In the old model, hackers were able to achieve remote code execution via MediaServer by bypassing SELinux with chained vulnerabilities. That changed in Android 7 (Nougat), where MediaServer functionality split into seven components such as MediaExtractor and MediaDrmServer, preventing format string vulnerabilities.



In Project Treble, Google accelerates the compartmentalizing of components and has introduced a bevy of new hardware abstraction layers (HALs) for the audio, camera and DRM servers inside the media framework. With those HALs in place more pieces of the Android framework are isolated in separate processes and sandboxes and no longer have access to the OS kernel—making it harder for hackers to chain vulnerabilities and compromise an Oreo device.

DATAFLOW SECURITY

Android brings seccomp via minijail

```
ioctl: 1
futex: 1
prctl: 1
write: 1
getpriority: 1
mmap2: 1
close: 1
10munmap: 1
dupe: 1
mprotect: 1
getuid32: 1
setpriority: 1
```

First lands on MediaServer (Android N)
Then Zygote (Android O)
Then expands to more and more processes

Chrome SBX before ~2017

Issue 610600: sandbox escape using ppapi broker
Reported by 70696...@gmail.com on Tue, May 10, 2016, 3:37 PM

UserAgent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) Ap

Steps to reproduce the problem:
run attached code

What is the expected behavior?
it fails

What went wrong?
Attached is a proof of concept exploit demonstrating how a co
broker. I believe that this affects all operating systems, but I ha
(canary). Some modification is required to get it to work on other versions

Clipboard IPC bug
Pinkie's sandbox escape bug is in the Clipboard IPC code, specifically in the `Clipboard::Di`
function which is reachable via the `ClipboardHostMsg_WriteObjectsAsync` IPC message

(Both `type` and `params` are controlled here)
[src/ui/base/clipboard/clipboard.cc]

```
void Clipboard::DispatchObject(ObjectType type, const ObjectMapParams&
...
  switch (type) {
...
    case CBF_SMBITMAP: {
...
      const char* raw_bitmap_data_const =
          reinterpret_cast<const char*>(&params[0].front());

      char* raw_bitmap_data = const_cast<char*>(raw_bitmap_data_const);
```

ONE CLASS TO RULE THEM ALL
0-DAY DESERIALIZATION VULNERABILITIES IN ANDROID

Or Peles, Roee Hay
IBM Security
{orpeles,roeeh}@il.ibm.com

Abstract

We present previously unknown high severity vulnerabilities in Android.

The first is in the Android Platform and Google Play Services. The Platform instance affects Android 4.3-5.1, M (Preview 1) or 55% of Android devices at the time of writing[1]. This vulnerability allows for arbitrary code execution in the context of many apps and services and results in elevation of privileges. In this

I. Introduction

Android is the most popular mobile operating system with 78% of the worldwide smartphone sales to end users in Q1 2015 [1].

Android apps are executed in a sandboxed environment to protect both the system and the hosted applications from malware [2]. The Android sandbox relies on the Linux kernel's isolation facilities. While

From: Jann Horn <jann () thejh net>
Date: Wed, 19 Nov 2014 02:31:15 +0100

In Android <5.0, java.io.ObjectInputStream did not check wh
is being deserialized is actually serializable. That issue
5.0 with this commit:
<https://android.googlesource.com/platform/libcore/+/738c83

This means that when ObjectInputStream is used on untrusted
can cause an instance of any class with a non-private param
to be created. All fields of that instance can be set to ar
malicious object will then typically either be ignored or c
which it doesn't fit, implying that no methods will be call
from it will be used. However, when it is collected by the
the object's finalize method.

Analyze Chain of Bugs

Use-After-Unmap bug in Android's libgralloc r
- hardware/qcom/display/msm8996/libgralloc

map and unmap mismatch in function gralloc_
gralloc_unmap

Thursday, December 1, 2016

BitUnmap: Attacking Android Ashmem

Posted by Gal Beniamini, Project Zero

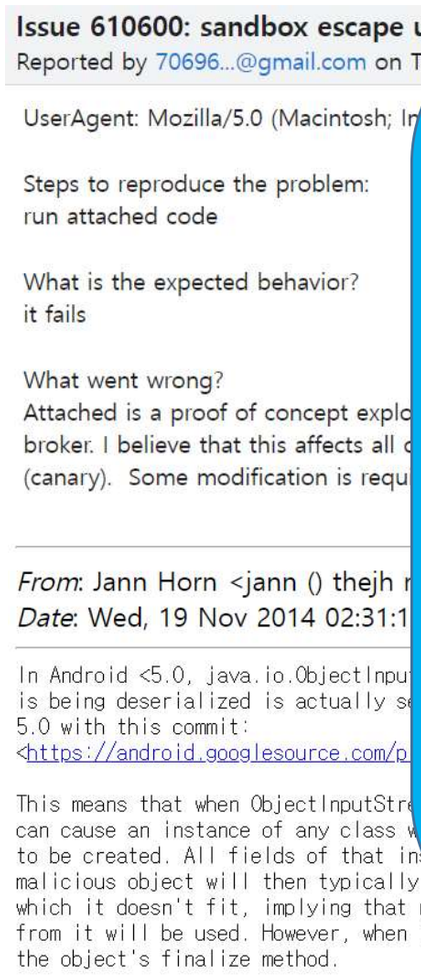
The law of leaky abstractions states that "all non-trivial abstractions,
post we'll explore the *ashmem* shared memory interface provided by
about its internal operation can result in security vulnerabilities affect

We'll walk through the process of discovering and exploiting a vulner
abstraction, which will allow us to elevate our privileges from any An
privileged contexts, including the highly-privileged "system_server".
core Android platform code for the Marshmallow and Nougat version
Android bulletin. For a detailed disclosure timeline, see the "Timeline

Chrome SBX before ~2017

Many pioneers targeting different attack surfaces :

Message Filters (Chrome IPC)
Parcel Deserialization
PPAPI
system_server (libraries)
GPU Process
QUIC
Hop to SBrowser, use N-Days
Logic bugs (Invoke Intent, content providers, deep links...)
...


Issue 610600: sandbox escape using
Reported by 70696...@gmail.com on Tue

UserAgent: Mozilla/5.0 (Macintosh; In

Steps to reproduce the problem:
run attached code

What is the expected behavior?
it fails

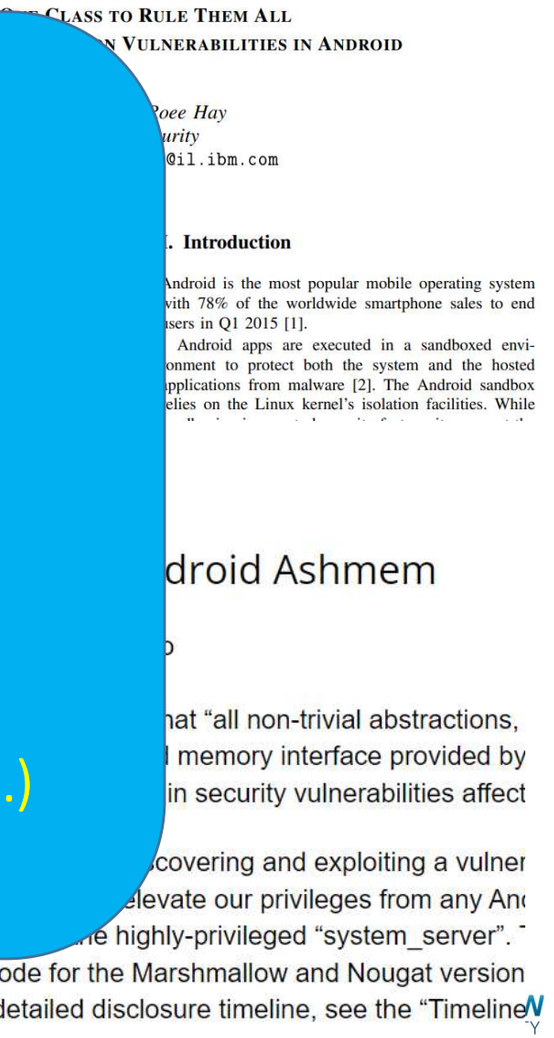
What went wrong?
Attached is a proof of concept explo
broker. I believe that this affects all
(canary). Some modification is requ

From: Jann Horn <jann () thejh
Date: Wed, 19 Nov 2014 02:31:1

In Android <5.0, java.io.ObjectInput
is being deserialized is actually s
5.0 with this commit:
<https://android.googlesource.com/p

This means that when ObjectInputStre
can cause an instance of any class w
to be created. All fields of that ins
malicious object will then typically
which it doesn't fit, implying that no
from it will be used. However, when it is
the object's finalize method.


grallOC_unmap


CLASS TO RULE THEM ALL
N VULNERABILITIES IN ANDROID

oee Hay
urity
@il.ibm.com

. Introduction

Android is the most popular mobile operating system
with 78% of the worldwide smartphone sales to end
sers in Q1 2015 [1].
Android apps are executed in a sandboxed envi-
onment to protect both the system and the hosted
pplications from malware [2]. The Android sandbox
elies on the Linux kernel's isolation facilities. While

droid Ashmem

hat "all non-trivial abstractions,
l memory interface provided by
in security vulnerabilities affect

covering and exploiting a vulner
levate our privileges from any An
he highly-privileged "system_server".
core Android platform code for the Marshmallow and Nougat version
Android bulletin. For a detailed disclosure timeline, see the "Timeline

gzobqq on Chrome mojo bugs (2015)

Security: AppCacheUpdateJob UaF allpublic	michaeln@chromium.org	----	48, 47	gzo...@gmail.com	Nov 20, 2015
Security: AppCacheDispatcherHost UaF with host transfer allpublic	michaeln@chromium.org	----	48, 47	gzo...@gmail.com	Nov 12, 2015
Security: AppCacheUpdateJob accesses map::end() allpublic	michaeln@chromium.org	----	48, 47	gzo...@gmail.com	Nov 4, 2015
Security: MidiHostMsg_SendData vector OOB on Android allpublic	yhirano@chromium.org	----	41	gzo...@gmail.com	Feb 8, 2015

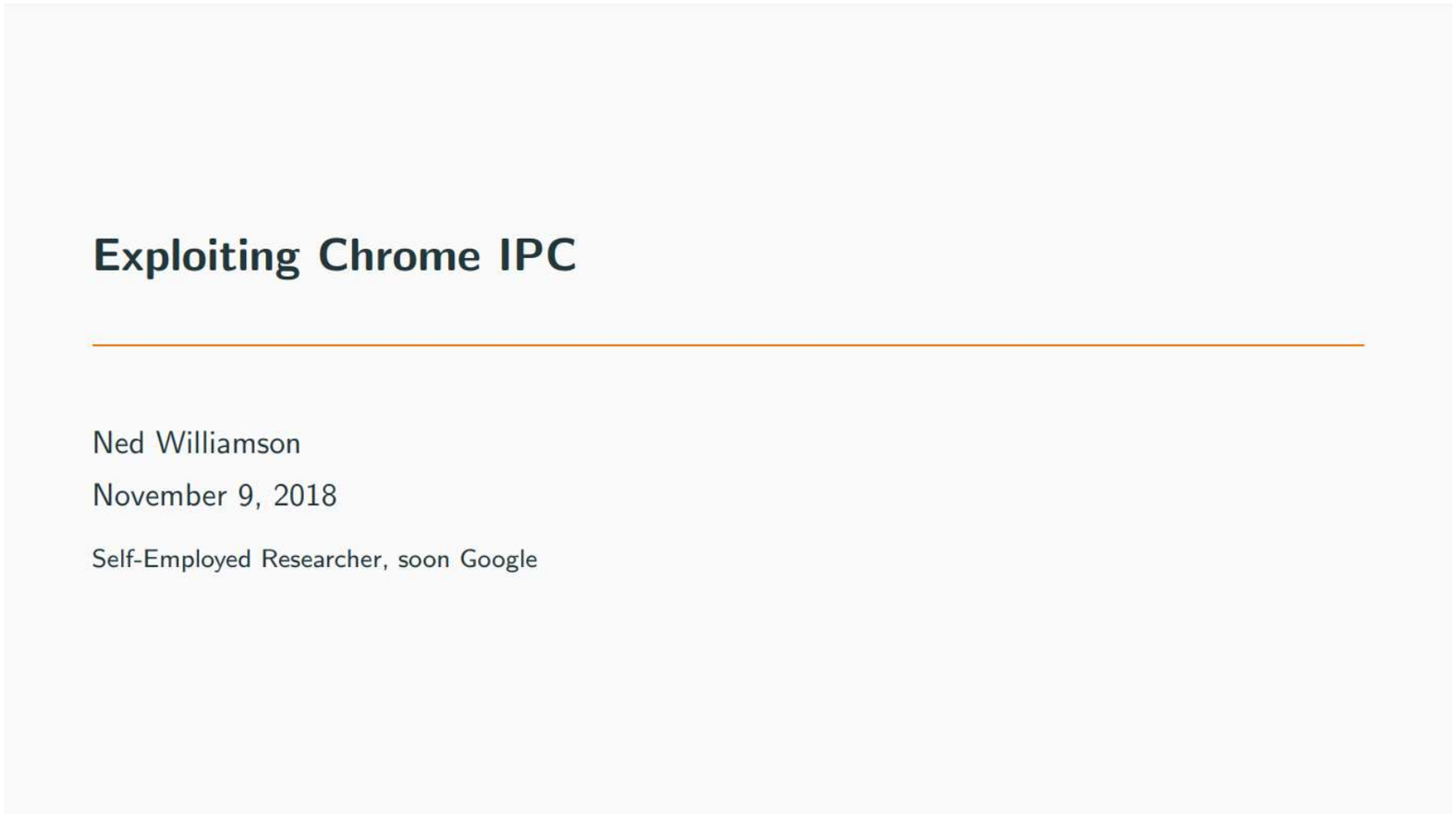
DATAFLOW SECURITY

Ned Williamson's take on mojo bughunting

Security: In-memory Cache UaF 2 allpublic	morlovich@chromium.org	----	67, 66, 68	nedwi...@gmail.com	Jun 27, 2019	Apr 12, 2018
Security: In-memory Cache UaF allpublic	jkarlin@chromium.org	----	66	nedwi...@gmail.com	Dec 4, 2018	Mar 30, 2018
Security: Blockfile Media Cache UaF allpublic	morlovich@chromium.org	----	66	nedwi...@gmail.com	Dec 4, 2018	Mar 28, 2018
Security: global-buffer-overflow in SkBitmap IPC Deserialization allpublic	dcheng@chromium.org	----	65	nedwi...@gmail.com	Nov 14, 2018	Oct 30, 2017
Security: OOB Read in BlobStorageContext::BlobFlattener::BlobFlattener allpublic	dmu...@chromium.org	----	65	nedwi...@gmail.com	Oct 5, 2018	Oct 28, 2017
Security: OOB Write in QuicStreamSequencerBuffer::OnStreamData allpublic	rch@chromium.org	----	65	nedwi...@gmail.com	May 1, 2020	Oct 26, 2017
Security: Stack Buffer Overflow in QuicClientPromisedInfo::OnPromiseHeaders allpublic	rch@chromium.org	68, 69, 70, 71, 72, 73, 74, 75, 76	76	nedwi...@gmail.com	Aug 14, 2019	Oct 24, 2017
Security: IndexedDB OpenCursor UaF allpublic	dmu...@chromium.org	----	60	nedwi...@gmail.com	Apr 26, 2018	Jun 2, 2017
Security: Use-after-free in IndexedDB Transactions allpublic	dmu...@chromium.org	----	58, 60, 61	nedwi...@gmail.com	Apr 26, 2018	May 22, 2017

DATAFLOW SECURITY

Ned Williamson's research disclosed



Exploiting Chrome IPC

Ned Williamson

November 9, 2018

Self-Employed Researcher, soon Google

Ned Williamson's research disclosed

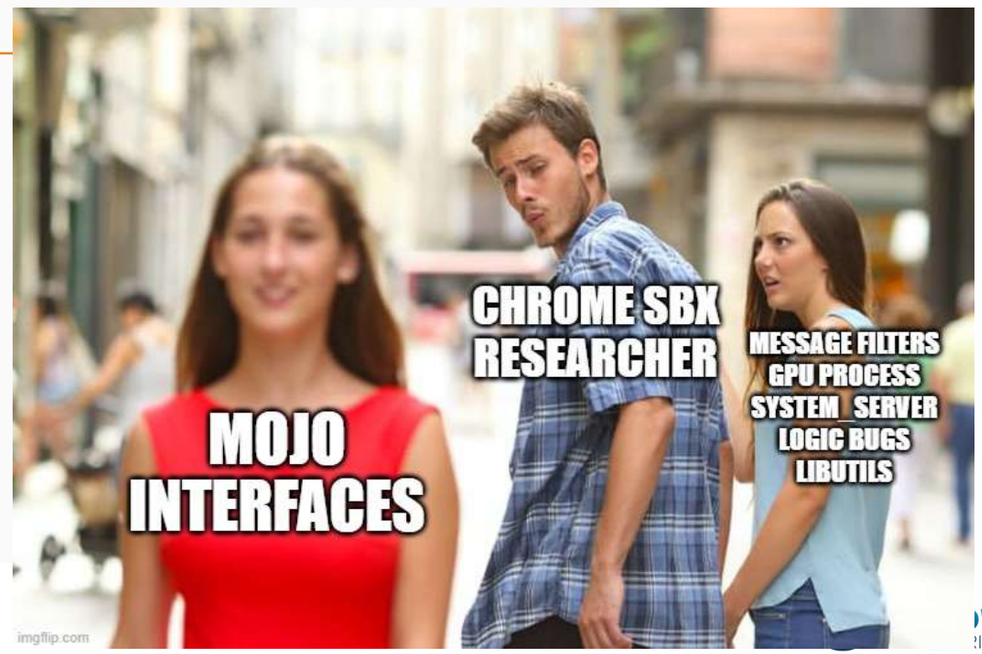
Exploiting Chrome IPC

Ned Williamson

November 9, 2018

Self-Employed Researcher, soon Google

Everyone seemed to have ditched all other attack surfaces in favor of Mojo 😂



Ned Williamson's research disclosed

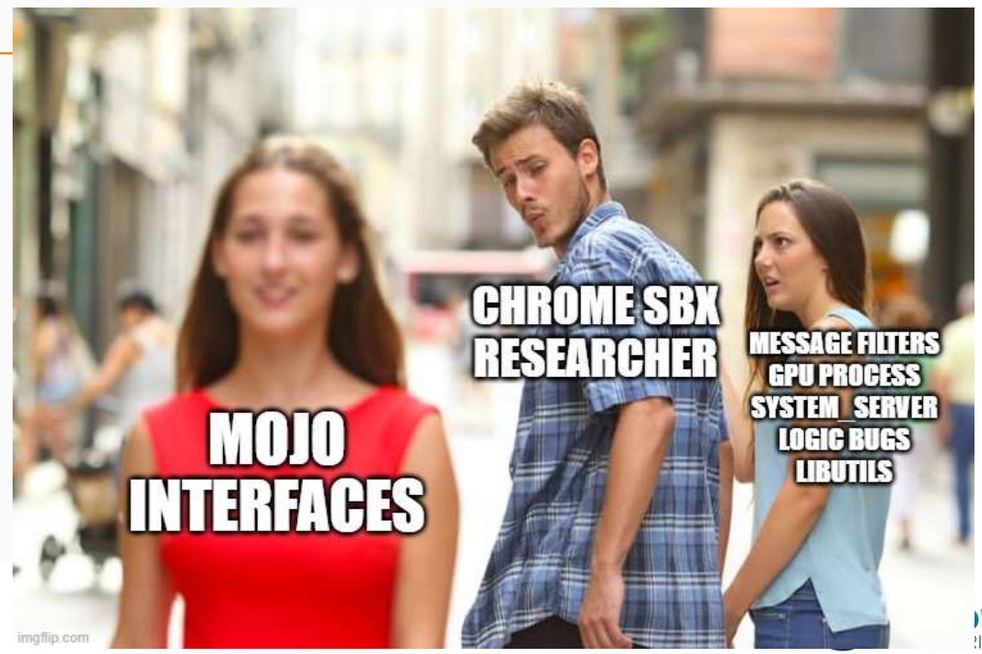
Exploiting Chrome IPC

Ned Williamson

November 9, 2018

Self-Employed Researcher, soon Google

Mojo based SBX research will flourish for several years after this (even until now)



Android kernel exploits

- Too many to fit in a slide…
- In the early days, there were very generic exploits that target syscalls accessible from any SEAndroid context (towelroot, iovyroot, …)
- These generic bugs become nearly extinct later on (except for binder exploits that appear in 2019)
- Many attackers would shift to audit vendor specific code/drivers

General kernel exploit flow

- Use bug to gain AAR / AAW (either by hitting addr_limit, fast return on set_fs(KERNEL_DS), or some simple JOP)
- Write 0 on global variable selinux_enforcing
- Write 0 on creds, enable all capabilities
- Enjoy # !
- This flow is still generally true even to this day for non Samsung phones

General kernel exploit flow

- Use bug to gain AAR / AAW (either by hitting addr_limit, fast return on set_fs(KERNEL_DS), or some simple JOP)
- Write 0 on global variable selinux_enforcing
- Write 0 on creds, enable all capabilities
- Enjoy # !

Samsung wasn't happy about this…

- This flow is still generally true even to this day for non Samsung phones

DATAFLOW SECURITY

Samsung’s CFI

Executable File | 2001 lines (1778 sloc) | 70.5 KB

```
#!/usr/bin/env python
#
# Instrument vmlinux STP, LDP and BLR instructions to protect RA and restrict jumpping
#
# Depends on:
# 1) a modified gcc that
#    - outputs 2 nop's before stp x29, x30 instructions
#    - outputs 1 nop after ldp x29, x30 instructions
# 2) a kernel built using gcc command-line options to prevent allocation of registers x16, x17, and x
#
# Copyright (c) 2015 Samsung Electronics Co., Ltd.
# Authors:      James Gleeson <jagleeso@gmail.com>
#               Wenbo Shen <wenbo.s@samsung.com>

# Set to False to have vmlinux instrumented during kernel build.
# OFF = True
OFF = False

# If true, skip instrumenting functions in hyperdrive/resource/debug/skip.txt
# (for debugging).
# SKIP_INSTRUMENTING = True
SKIP_INSTRUMENTING = False
```

Python script to insert NOP’s in certain places, insert instructions before function call instruction, and a hash right before the function prologue

DATAFLOW SECURITY

Samsung’s CFI

Executable File | 2001 lines (1778 sloc) | 70.5 KB

```
#!/usr/bin/env python
#
# Instrument vmlinux STP, LDP and BLR instructions to protect RA and restrict jumpping
#
# Depends on:
# 1) a modified gcc that
#    - outputs 2 nop's before stp x29, x30 instructions
#    - outputs 1 nop after ldp x29, x30 instructions
# 2) a kernel built using gcc command-line options to prevent allocation of registers x16, x17, and x
#
# Copyright (c) 2015 Samsung Electronics Co., Ltd.
# Authors:      James Gleeson <jagleeso@gmail.com>
#               Wenbo Shen <wenbo.s@samsung.com>

# Set to False to have vmlinux instrumented during kernel build.
# OFF = True
OFF = False

# If true, skip instrumenting functions in hyperdrive/resource/debug/skip.txt
# (for debugging).
# SKIP_INSTRUMENTING = True
SKIP_INSTRUMENTING = False
```

However basically you could jump to the beginning of every function. Very weak CFI

DATAFLOW SECURITY

General kernel exploit flow

- Use bug to gain AAR / AAW (either by hitting addr_limit, fast return on set_fs(KERNEL_DS), or some simple JOP)

- Write 0 on global variable selinux_enforcing
- Write 0 on creds, enable all capabilities

Samsung wasn't happy about these too…

- Enjoy # !

- This flow is still generally true even to this day for non Samsung phones

DATAFLOW SECURITY

Samsung’s RKP (Realtime Kernel Protection)

Knox White Paper

Real-time Kernel Protection (RKP)

The Knox Platform's patented Real-time Kernel Protection (RKP) is the industry's strongest protection against kernel threats and exploits. RKP works seamlessly out-of-the-box, with no setup required. Simply powering on a Samsung Knox device provides world-class threat protection and attack mitigation. RKP supports the rest of the Knox security offerings to provide full security coverage without the typical gaps anticipated with mobile devices.

Why does kernel protection matter?

Kernel protection is central to device security and enterprise data protection. When attackers find software vulnerabilities, they often escalate privileges and compromise the core of the OS: the kernel.

A compromised kernel can leak sensitive data and even allow remote monitoring and control of the affected device. Other more commonplace protections like Secure Boot or hardware-backed keystores are of little value if the kernel itself is controlled at runtime. After a device boots and decrypts sensitive content, a kernel compromise can result in data leaks that directly impact an enterprise’s data integrity.

RKP design and structure

As part of the Knox Platform's security offerings, RKP employs a security monitor within an isolated execution environment. Depending on the device model, either a dedicated hypervisor or the hardware-backed secure world provided by ARM TrustZone technology provides the isolated execution environment.

DATAFLOW SECURITY

Samsung’s RKP (Realtime Kernel Protection)

Knox White Paper

Real-time Kernel Prote

The Knox Platform's patented Real-time Kernel Pr
threats and exploits. RKP works seamlessly out-of
Knox device provides world-class threat protectio
offerings to provide full security coverage without

Why does kernel protection matter

Kernel protection is central to device security and
vulnerabilities, they often escalate privileges and

A compromised kernel can leak sensitive data and
Other more commonplace protections like Secure B
itself is controlled at runtime. After a device boots an
data leaks that directly impact an enterprise’s data integrity.

RKP design and structure

As part of the Knox Platform's security offerings, RKP employs a security monitor within an isolated execution environment. Depending on the device model, either a dedicated hypervisor or the hardware-backed secure world provided by ARM TrustZone technology provides the isolated execution environment.

Protects important things on heap from hypervisor :

Cred structures and related
Important globals (selinux_enforcing, …)
Page tables
And other things…

DATAFLOW SECURITY

Samsung’s RKP (Realtime Kernel Protection)

Knox White Paper

Real-time Kernel Prote

The Knox Platform's patented Real-time Kernel Pr
threats and exploits. RKP works seamlessly out-of
Knox device provides world-class threat protectio
offerings to provide full security coverage without

Why does kernel protection matter

Kernel protection is central to device security and
vulnerabilities, they often escalate privileges and

A compromised kernel can leak sensitive data and
Other more commonplace protections like Secure B
itself is controlled at runtime. After a device boots an
data leaks that directly impact an enterprise’s data integrity.

RKP design and structure

As part of the Knox Platform's security offerings, RKP employs a security monitor within an isolated execution environment. Depending on the device model, either a dedicated hypervisor or the hardware-backed secure world provided by ARM TrustZone technology provides the isolated execution environment.

There were some very trivial bypasses for a couple years. Also, RKP kept (inadvertently?) removing/reintroducing the .RO enforcement every once in a while.

Bypass RKP…?

Or don’t even deal with RKP

DATAFLOW SECURITY

General kernel exploit flow – Galaxy devices

- Use bug to gain AAR / AAW (Some minor annoyances with Samsung's custom CFI)
- ~~Write 0 on global variable selinux_enforcing~~
- ~~Write 0 on creds, enable all capabilities~~
- Migrate to a userland process running as root, with a very powerful SEAndroid context, and inject attacker code there
- Now you have both code running as root, and kernel R/W
- Enjoy # !

DATAFLOW SECURITY

LLVM Control Flow Integrity

10 October 2018

Control Flow Integrity in the Android kernel

Posted by Sami Tolvanen, Staff Software Engineer, Android Security

Android's security model is enforced by the Linux kernel, which makes it a tempting target for attackers. We have put a lot of effort into hardening the kernel in previous Android releases and in Android 9, we continued this work by focusing on compiler-based security mitigations against code reuse attacks.

Google's Pixel 3 will be the first Android device to ship with LLVM's forward-edge Control Flow Integrity (CFI) enforcement in the kernel, and we have made CFI support available in Android kernel versions 4.9 and 4.14. This post describes how kernel CFI works and provides solutions to the most common issues developers might run into when enabling the feature.

LLVM Control Flow Integrity

10 October 2018

Control Flow Integrity in the Android kernel

Posted by Sami Tolvanen, Staff Software Engineer,

Android's security model is enforced by the Linux
attackers. We have put a lot of effort into hardeni
Android 9, we continued this work by focusing on
reuse attacks.

Google's Pixel 3 will be the first Android device to
Integrity (CFI) enforcement in the kernel, and we h
versions 4.9 and 4.14. This post describes how ke
common issues developers might run into when e

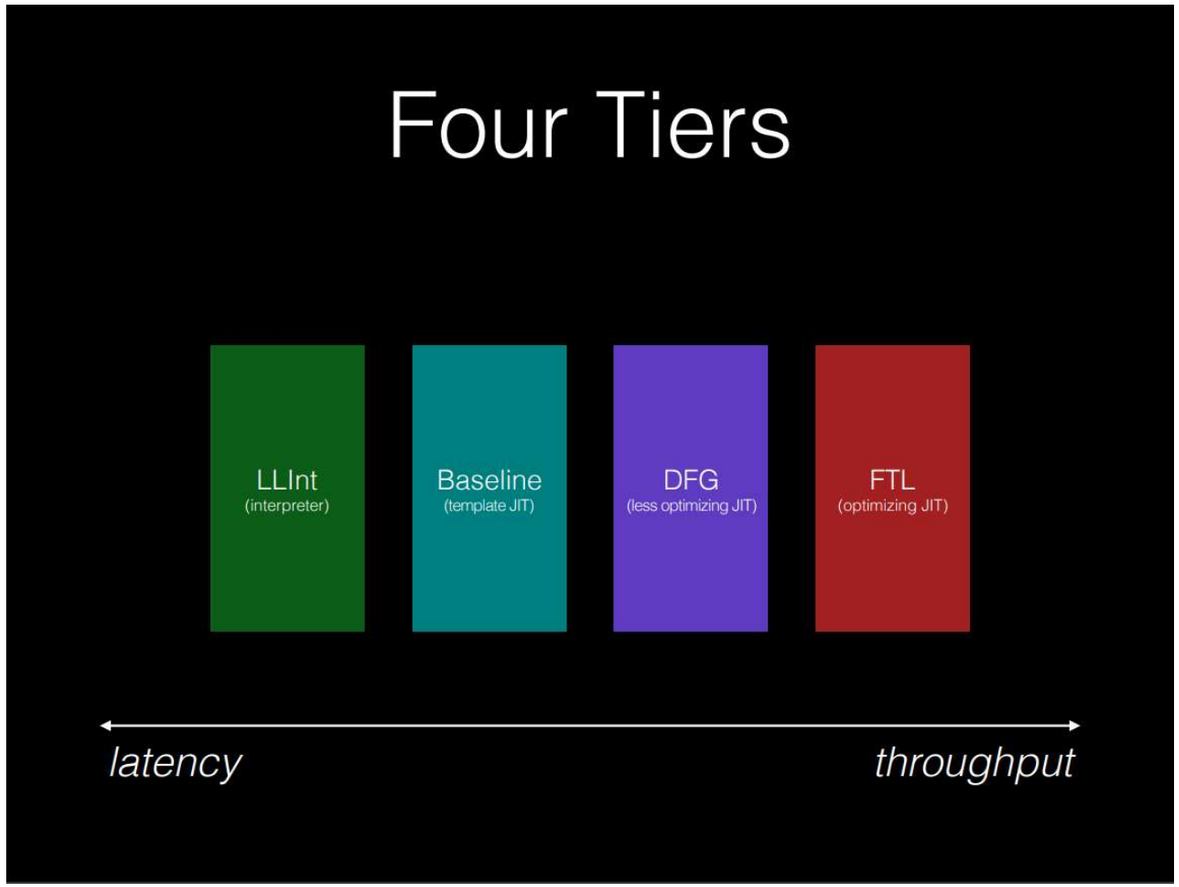
Finally lands on Pixel 3

Later on picked up by Samsung and replaces their own CFI

Makes kernel exploitation more challenging

The present

Safari RCE - JavaScriptCore



JavaScriptCore RCE scene has been very dormant for the longest time.

Bugs can be very long living, little changes in architecture(except libpas) in terms of exploitation, and little advancements in mitigation

DATAFLOW SECURITY

v8 “still” a prime target for RCE

Security Fixes and Rewards

Note: Access to bug details and links may be kept restricted until a majority of users are u

This update includes 2
security fixes. Below, we highlight fixes that were contributed by external researchers. Ple
for more information.

[$NA][1315901]
High CVE-2022-1364: Type Confusion in V8. *Reported by Clément Lecigne of Google's Threat Analysis Group on 2022-04-13*

We would also like to thank all security researchers that worked with us during the development cycle to prevent security bugs from eve

Nothing really changed from the attacker side

However…

DATAFLOW SECURITY

V8 Heap Sandbox

V8 Sandbox

Aka. “Ubercage”

Author: saelo@
First Published: July 2021
Last Updated: July 2022
Status: Living Doc
Visibility: **PUBLIC**

This document is part of the V8 Sandbox Project and covers the high-level design of the sandbox.

Summary

Objective: build a low-overhead, in-process sandbox for V8.

Motivation: V8 bugs typically allow for the construction of unusually powerful exploits. Furthermore, these bugs are unlikely to be mitigated by memory safe languages or upcoming hardware-assisted security features such as MTE or CFI. As a result, V8 is especially attractive for real-world attackers.

DATAFLOW SECURITY

V8 Heap Sandbox

V8 Sandbox

Aka. “Ubercage”

Author: saelo@
First Published: July 2021
Last Updated: July 2022
Status: Living Doc
Visibility: **PUBLIC**

This document is part of the V8 Sandbox Project

Summary

Objective: build a low-overhead, in-process sand

Motivation: V8 bugs typically allow for the const
bugs are unlikely to be mitigated by memory safe
features such as MTE or CFI. As a result, V8 is esp

Basically “Gigacage” for v8

In essence :
You can get R/W with your v8 exploit,
but the R/W will never escape the cage

DATAFLOW SECURITY

V8 Heap Sandbox

V8 Sandbox

Aka. “Ubercage”

Author: saelo@
First Published: July 2021
Last Updated: July 2022
Status: Living Doc
Visibility: **PUBLIC**

This document is part of the V8 Sandbox Project

Summary

Objective: build a low-overhead, in-process sand

Motivation: V8 bugs typically allow for the const
bugs are unlikely to be mitigated by memory safe
features such as MTE or CFI. As a result, V8 is esp

Now you need a “V8 Heap Sandbox bypass” along with the v8 exploit

More components in exploit chain, more maintenance costs

DATAFLOW SECURITY

MiraclePtr lands

We successfully rewrote more than 15,000 raw pointers in the Chrome codebase into raw_ptr<T>, then enabled BackupRefPtr for the browser process on Windows and Android (both 64 bit and 32 bit) in Chrome 102 Stable. We anticipate that MiraclePtr meaningfully reduces the browser process attack surface of Chrome by protecting ~50% of use-after-free issues against exploitation. We are now working on enabling BackupRefPtr in the network, utility and GPU processes, and for other platforms. In the end state, our goal is to enable BackupRefPtr on *all* platforms because that ensures that a given pointer is protected for *all* users of Chrome.

Balancing Security and Performance

MiraclePtr lands

We successfully rewrote more than 15,000 raw pointers in the Chrome codebase into raw_ptr<T>, then enabled BackupRefPtr for the brov
Android (both 64 bit and 32 bit) in Chrome 102 Sta
meaningfully reduces the browser process attack
~50% of use-after-free issues against exploitation.
BackupRefPtr in the network, utility and GPU proce
end state, our goal is to enable BackupRefPtr on a
that a given pointer is protected for *all* users of Ch

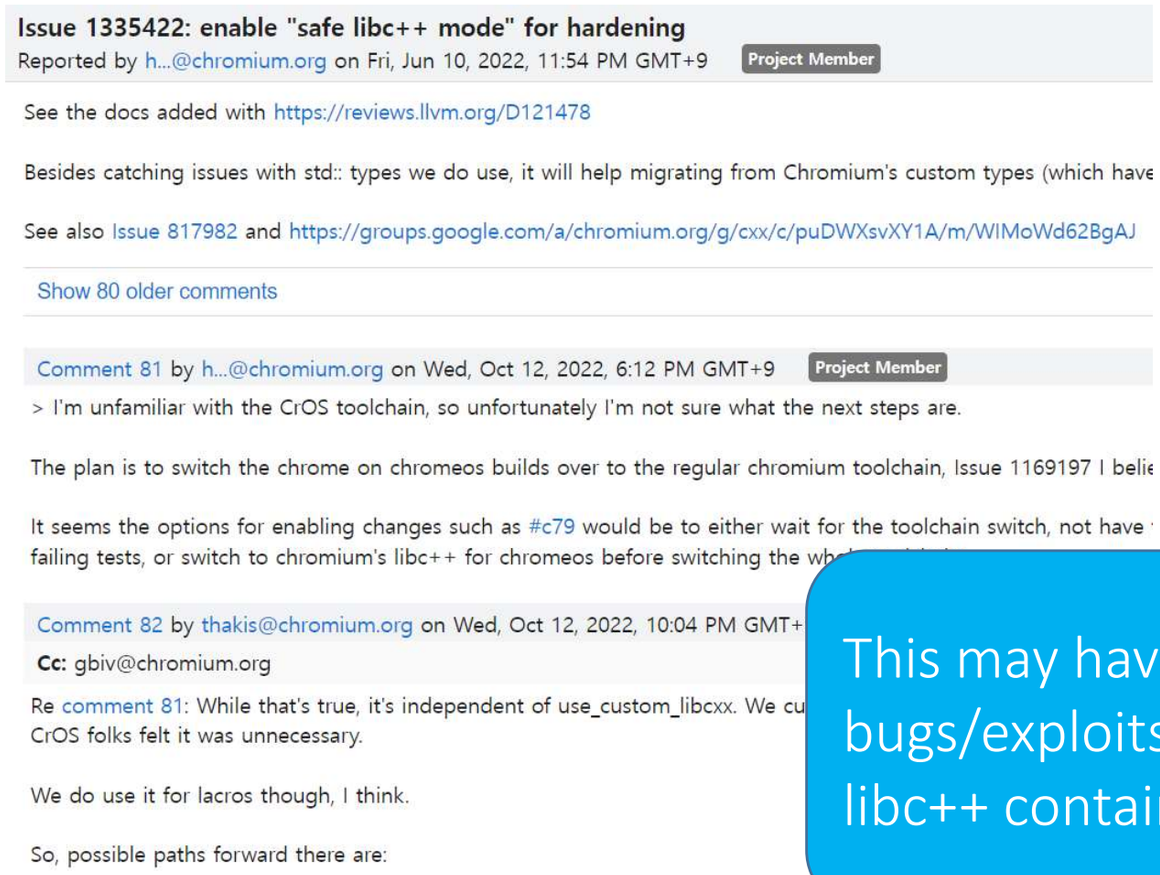
Balancing Security and Performance

Significant implications on Chrome SBX research, as the majority of bugs are UAF. MiraclePtr is a mitigation specifically designed to kill the UAF bug class

Strategic shift from attackers due to this mitigation

DATAFLOW SECURITY

safe libc++ mode



Issue 1335422: enable "safe libc++ mode" for hardening
Reported by h...@chromium.org on Fri, Jun 10, 2022, 11:54 PM GMT+9 Project Member

See the docs added with https://reviews.llvm.org/D121478

Besides catching issues with std:: types we do use, it will help migrating from Chromium's custom types (which have

See also Issue 817982 and https://groups.google.com/a/chromium.org/g/cxx/c/puDWXsvXY1A/m/WIMoWd62BgAJ

Show 80 older comments

Comment 81 by h...@chromium.org on Wed, Oct 12, 2022, 6:12 PM GMT+9 Project Member

> I'm unfamiliar with the CrOS toolchain, so unfortunately I'm not sure what the next steps are.

The plan is to switch the chrome on chromeos builds over to the regular chromium toolchain, Issue 1169197 I belie

It seems the options for enabling changes such as #c79 would be to either wait for the toolchain switch, not have
failing tests, or switch to chromium's libc++ for chromeos before switching the wh

Comment 82 by thakis@chromium.org on Wed, Oct 12, 2022, 10:04 PM GMT+

Cc: gbiv@chromium.org

Re comment 81: While that's true, it's independent of use_custom_libcxx. We cu
CrOS folks felt it was unnecessary.

We do use it for lacros though, I think.

So, possible paths forward there are:

This may have disturbed bugs/exploits that rely on libc++ containers

DATAFLOW SECURITY

BTI lands

Issue 1145581: Armv8.5-A BTI enablement in Chrome

Reported by jonat...@arm.com on Thu, Nov 5, 2020, 12:21 AM GMT+9 **Project Member**

https://chromium.googlesource.com/chromium/src/+/9db3d451259db5140cc1a6733ecdc01d057eeaa2

commit 9db3d451259db5140cc1a6733ecdc01d057eeaa2
Author: Richard Townsend <Richard.Townsend@arm.com>
Date: Tue Jun 07 03:45:29 2022

security: enable BTI by default for Arm platforms

BTI (Branch Target Identification) is a control flow integrity measure which hardens indirect branches. Under BTI, all indirect branches must land on one of these instructions:

- bti c
- bti j
- paciasp

If you're a maintainer and you've bisected back to this CL, this indicates that a component is missing one of these landing pads.

Exploit may need to bypass BTI, COOP style

DATAFLOW SECURITY

Also PAC

```
} else if (current_cpu == "arm64" || v8_current_cpu == "arm64") {
  # arm64 supports only "hard".
  arm_float_abi = "hard"
  arm_use_neon = true
  declare_args() {
    # Enables the new Armv8 branch protection features. Valid strings are:
    # - "pac": Enables Pointer Authentication Code (PAC, featured in Armv8.3)
    # - "standard": Enables both PAC and Branch Target Identification (Armv8.5).
    # - "none": No branch protection.
    arm_control_flow_integrity = "standard"
```

Has minimal influence on exploitation

Scudo allocator

Scudo is now Android's default native allocator

In Android 11, Scudo replaces jemalloc as the default native allocator for Andro
Scudo is a hardened memory allocator designed to help detect and mitigate me
corruption bugs in the heap, such as:

- Double free,
- Arbitrary free,
- Heap-based buffer overflow,
- Use-after-free

Scudo does not fully prevent exploitation but it does add a number of sanity checks which are effective at strengthening the heap against some memory corruption bugs.

Default allocator for userland processes

DATAFLOW SECURITY

On the Safari side…

Removal of StructureID randomization

Changeset 286502 in webkit

Timestamp: Dec 3, 2021 10:30:11 AM (16 months ago)
Author: keith_miller@apple.com

Message: Remove StructureIDBlob
https://bugs.webkit.org/show_bug.cgi?id=233723

Reviewed by Yusuke Suzuki.

StructureIDBlob isn't very useful now that StructureIDs are just the
bottom bits of the pointer on 64 bit platforms. In a follow up patch
I'll change the layout of JSCell and Structure so that TypeInfo creation
can be a single load platforms that allow (and don't penalize) misaligned loads.

Now the StructureID is just part of the Structure object's address

JIT Cage

JIT Cage

Thou shall not… :

- RET, BLR, BR, SVC etc…
- Sign with IA key
- Restricted signing with IB key

On instructions executed on the JIT region.
Basically JIT Cage wants to remove ability to create “arbitrary function call primitive”

DATAFLOW SECURITY

JIT Cage

Need more effort to get around this…

Or try to launch PE without arbitrary code execution

DATAFLOW SECURITY

Safari WebContent sandbox

```
; CRCopyRestrictionsDictionary periodically tries to CFPreferencesAppSynchronize com.apple.springboard.plist
; which will attempt to create the plist if it doesn't exist -- from any application.  Only SpringBoard is
; allowed to write its plist; ignore all others, they don't know what they are doing.
; See <rdar://problem/9375027> for sample backtraces.
(deny file-write* (with no-report)
    (home-prefix "/Library/Preferences/com.apple.springboard.plist"))

;; <rdar://problem/34986314>
(mobile-preferences-read "com.apple.indigo")

;;;
;;; End UIKit-apps.sb content
;;;

(mobile-preferences-read "com.apple.AdLib.plist")

(deny file-read* (with no-report)
    (home-literal
        "/Library/Preferences/com.apple.WebKit.WebContent.plist"
        "/Library/Preferences/com.apple.CFNetwork.plist"
        "/Library/Preferences/com.apple.AppSupport.plist"
    )
)

(with-filter (system-attribute apple-internal)
    (mobile-preferences-read "com.apple.CFNetwork"))
```

DATAFLOW SECURITY

Safari WebContent sandbox

```
634  ; CRCopyRestrictionsDictionary periodically tries to CFPreferencesAppSynchronize com.apple.springboard.plist
635  ; which will attempt to create the plist if it doesn't exist -- from any application.  Only SpringBoard is
636  ; allowed to write its plist; ignore all others, they don't know what they are doing.
637  ; See <rdar://problem/9375027> for sample backtraces.
638  (deny file-write* (with no-report)
639      (home-prefix "/Library/Preferences/com.apple.springboard.plist"))
640
641  ;; <rdar://problem/34986314>
642  (mobile-preferences-read "com.apple.indigo")
643
644  ;;;
645  ;;; End UIKit-apps.sb content
646  ;;;
647
648  (mobile-preferences-read "com.apple.AdLib.plist")
649
650  (deny file-read* (with no-report)
651      (home-literal
652          "/Library/Preferences/com.apple.WebKit.WebContent.plist"
653          "/Library/Preferences/com.apple.CFNetwork.plist"
654          "/Library/Preferences/com.apple.AppSupport.plist"
655      )
656  )
657
658  (with-filter (system-attribute apple-internal)
659      (mobile-preferences-read "com.apple.CFNetwork"))
```

Stripped down to minimum…

A lot of attractive IPC/XPC endpoints and IOKit userclients are no longer there

Only the absolutely necessary syscall / mach-traps remain

DATAFLOW SECURITY

Safari WebContent sandbox

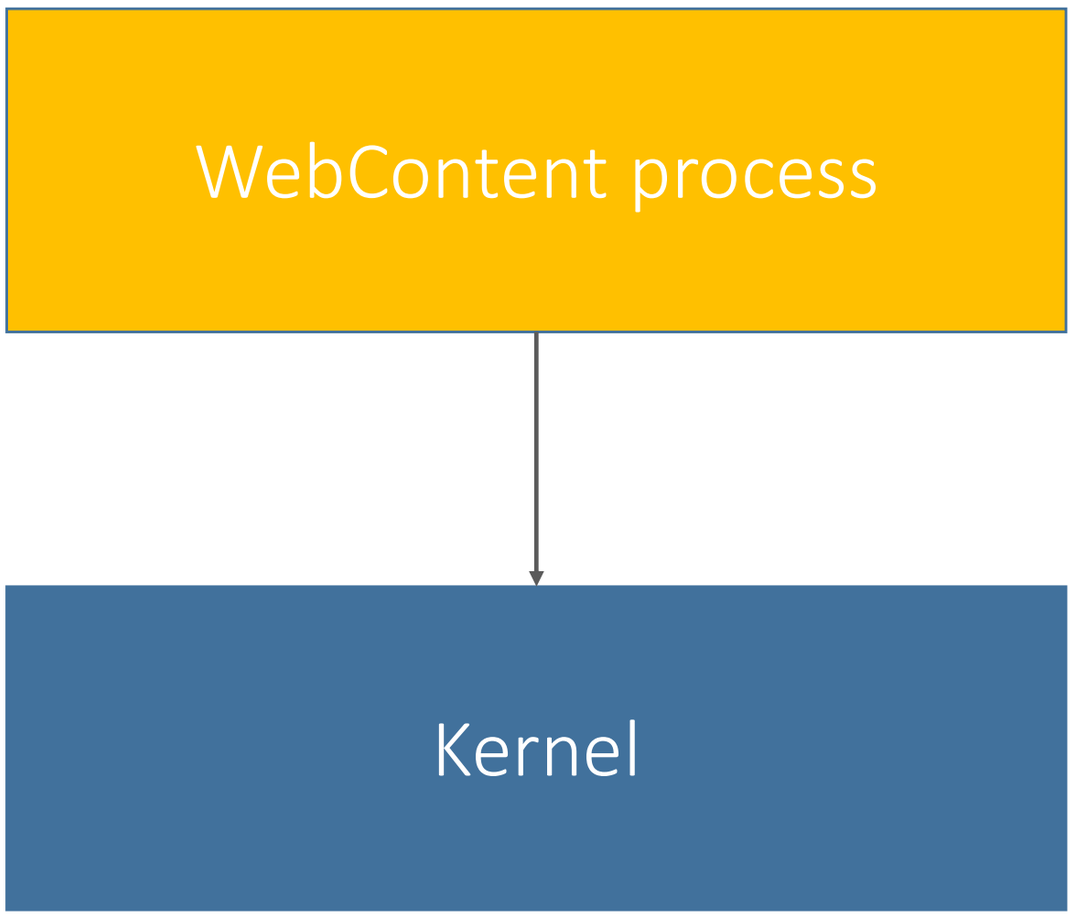
```
634  ; CRCopyRestrictionsDictionary periodically tries to CFPreferencesAppSynchronize com.apple.springboard.plist
635  ; which will attempt to create the plist if it doesn't exist -- from any application.  Only SpringBoard is
636  ; allowed to write its plist; ignore all others, they don't know what they are doing.
637  ; See <rdar://problem/9375027> for sample backtraces.
638  (deny file-write* (with no-report)
639      (home-prefix "/Library/Preferences/com.apple.springboard.plist"))
640
641  ;; <rdar://problem/34986314>
642  (mobile-preferences-read "com.apple.indigo")
643
644  ;;;
645  ;;; End UIKit-apps.sb content
646  ;;;
647
648  (mobile-preferences-read "com.apple.AdLib.plist")
649
650  (deny file-read* (with no-report)
651      (home-literal
652          "/Library/Preferences/com.apple.WebKit.WebContent.plist"
653          "/Library/Preferences/com.apple.CFNetwork.plist"
654          "/Library/Preferences/com.apple.AppSupport.plist"
655      )
656  )
657
658  (with-filter (system-attribute apple-internal)
659      (mobile-preferences-read "com.apple.CFNetwork"))
```

End result :

- You may still have a kernel bug that triggers on WebContent
- Extremely challenging to create exploit primitives

DATAFLOW SECURITY

iOS Full chain (~2021)

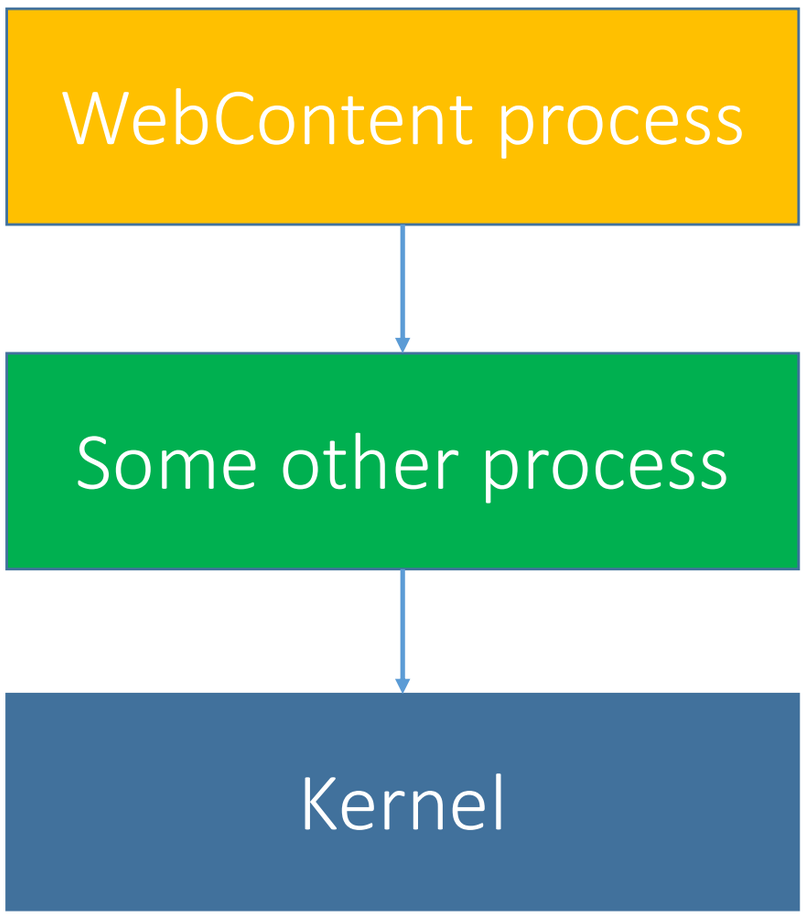


After PAC :

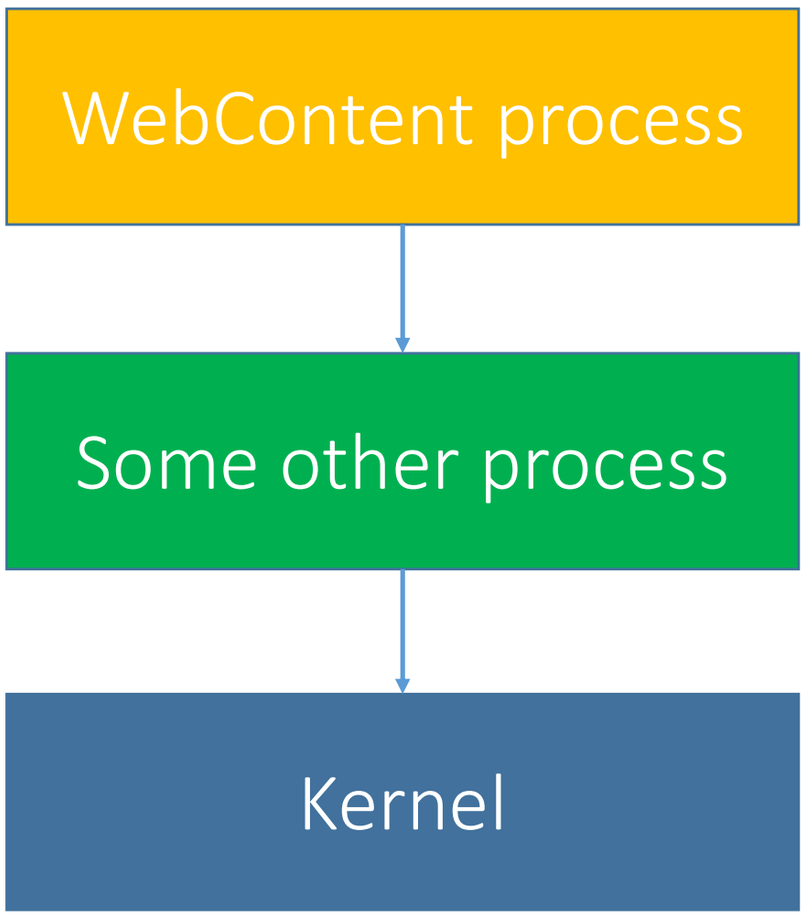
- Webkit arbitrary R/W
- Userland PAC bypass (function call primitive)
- Execute Shellcode
- Kernel arbitrary R/W
- Kernel PAC bypass

Do you realize that iOS browser chains never really required an SBX…??
→ Why even bother with SBX, go straight to the kernel with a kernel bug

iOS Full chain (now)



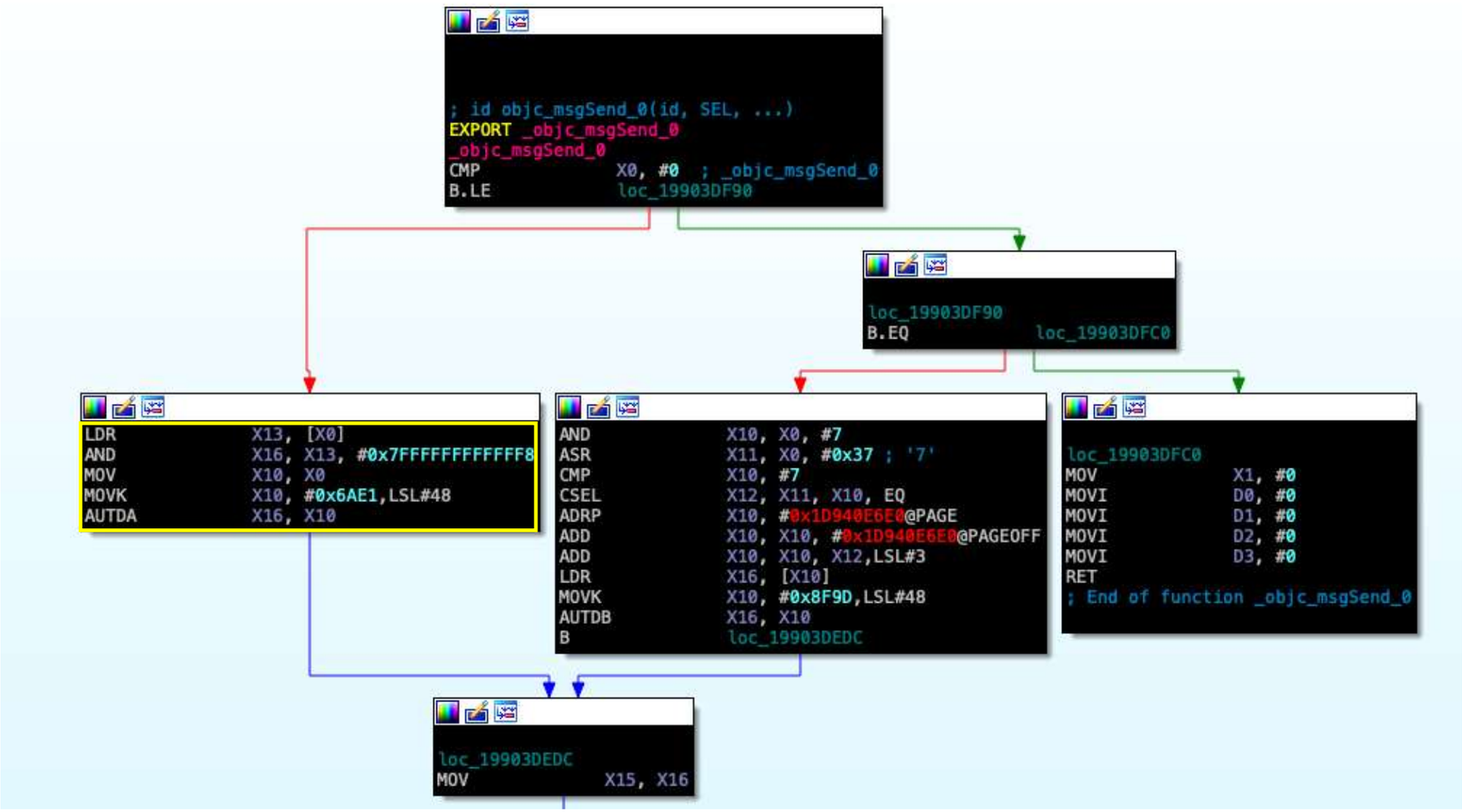
iOS Full chain (now)



- Webkit arbitrary R/W
- Userland PAC bypass
- Execute Shellcode
- SBX to different userland process
- Userland PAC bypass
- Kernel arbitrary R/W
- Kernel PAC bypass

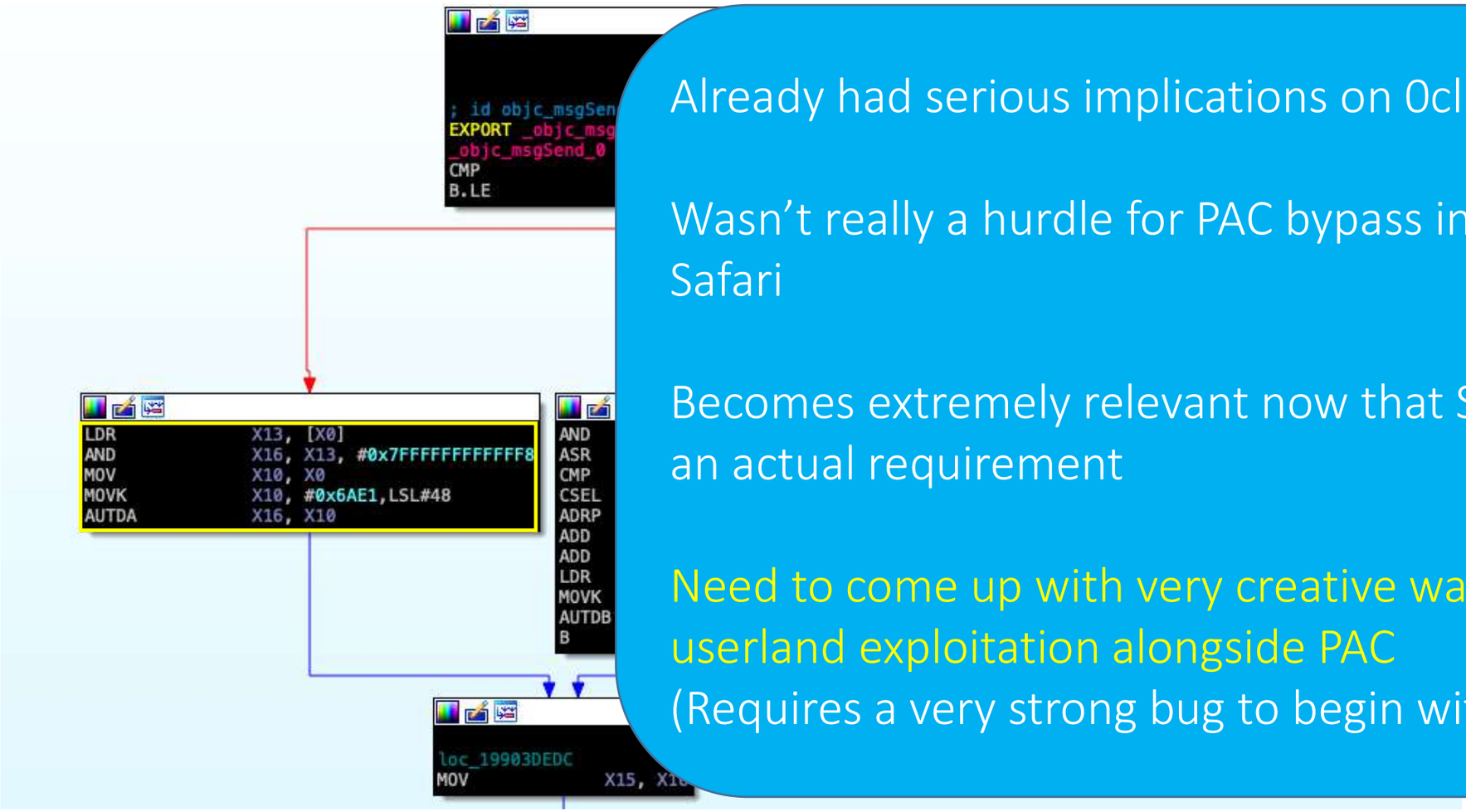
SBX is now actually a critical component in iOS 1click full chains

ISA PAC lands in early 2021



DATAFLOW SECURITY

ISA PAC lands in early 2021



Already had serious implications on 0click

Wasn't really a hurdle for PAC bypass in Safari

Becomes extremely relevant now that SBX is an actual requirement

Need to come up with very creative ways for userland exploitation alongside PAC
(Requires a very strong bug to begin with…)

DATAFLOW SECURITY

Zero on free

- The system memory allocator `free` operation zeroes out all deallocated blocks in iOS 16.1 beta or later. Invalid accesses to free memory might result in new crashes or corruption, including:
  - Read-after-free bugs that previously observed the old contents of a block may now observe zeroes instead
  - Write-after-free bugs
    zero memory

  To debug these issues, u
    3). (97449075)

This may affect certain bugs or exploit techniques that relied on uninitialized memory

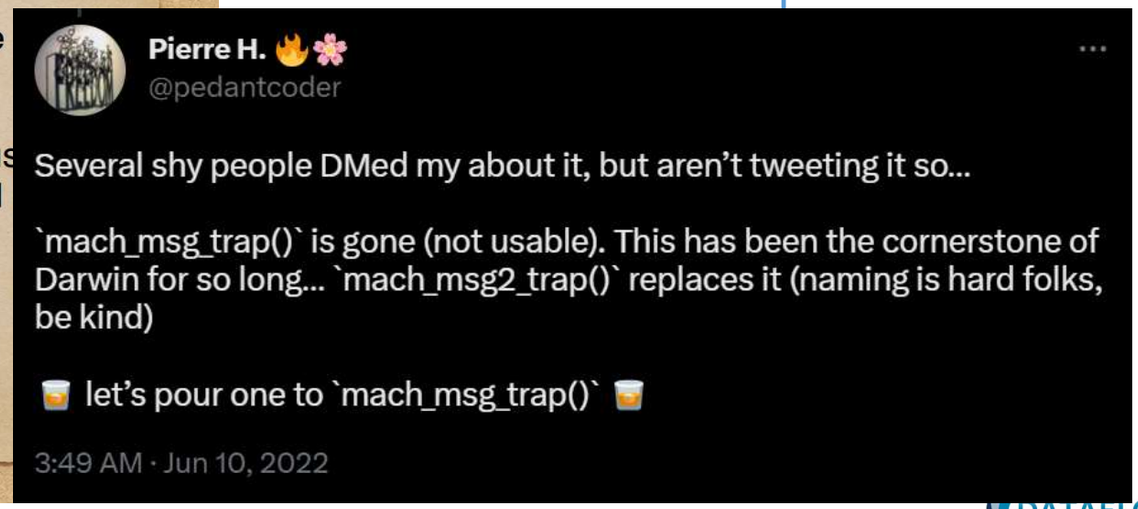
DATAFLOW SECURITY

mach_msg2_trap

Mach and CFI

- Sending a Mach message gives a lot of control over your process, the kernel and other processes too
- The mach_msg function is used for this
  - 'mach_msg' requires multiple arguments
    - With a function call with single controlled argument primitive tricky to call it
      - Fortunately, mach_msg_send and mach_msg_receive jus one argument and will put together the arguments based read from that argument

I think this was a “sniper mitigation” to kill primitive hinted in this talk…



The slaughter of “ipc_kmsg” primitive

Conclusion

- New “good” bugs are still being introduced in iOS - the ones in this talk were new in iOS 15 and not patched until a few months ago
- When engaged in full time security research, new bugs/techniques/attack surfaces often come to you as you go while working on something else
- Apple Security Bounty could use further improvements
- kmsg will forever be remembered as an iOS exploit SDK
- Finally, I’m at Dataflow Forensics, a new company founded in collaboration with Dataflow Security - we are hiring! (reach me on twitter - @jaakerblom)

DATAFLOW SECURITY

Data PAC everywhere to kill primitives

- We split `ipc_kmsg`, so that it no longer stores kernel pointers in the data submap for user-to-user messages. `ipc_kmsg` has been very useful for building various exploit primitives. This split landed in iOS 16.
- We aggressively PAC-protected pointers from typed allocations to data allocations to minimize opportunities for second-order type confusion. While any pointer/pointer overlap is subject to potentially exploitable type confusion, data allocations are particularly attractive for all the reasons we outlined in the introduction.
- We made `kfree_type()` and other free APIs zero the pointer itself that's being

Apple aggressively makes majority of objects with inline attacker controlled data out-of-line, and pure data is allocated on KHEAP_DATA_BUFFERS

The pointers to these buffers will be PAC'd

DATAFLOW SECURITY

Zone sequestering

When zones are sequestered, the zone GC behaves slightly differently. Instead of returning both the physical memory and the VA range to the `zone_map`, it returns only the physical memory and remembers the VA in the new fourth list. Keeping the virtual address range allocated to the zone even as the range is depopulated of physical pages ensures that the VA cannot be reused by any other zone. Allocations in single-type zones (e.g. the “proc” and “thread” zones) are no longer susceptible to direct type confusion via UAF, as their VA can’t be reused for another type. And in the traditional kalloc zones, objects can
the same size class. GC attacks across z
longer possible.

Cross zone attacks are no longer viable
(Free all objects of type-A on Zone A→ GC → Reallocate and with type-B object in Zone B)

DATAFLOW SECURITY

kalloc_type (Sad Feng Shui)

```
#if ZSECURITY_CONFIG(SAD_FENG_SHUI)
        /*
         * Randomly assign zones to one of the 4 general submaps,
         * and pick whether they allocate from the begining
         * or the end of it.
         *
         * A lot of OOB exploitation relies on precise interleaving
         * of specific types in the heap.
         *
         * Woops, you can't guarantee that anymore.
         */
        for (zone_id_t i = 1; i < MAX_ZONES; i++) {
                uint32_t r = zalloc_random_uniform(0,
                    ZSECURITY_CONFIG_GENERAL_SUBMAPS * 2);

                zone_security_array[i].z_submap_from_end = (r & 1);
                zone_security_array[i].z_submap_idx += (r >> 1);
        }
#endif /* ZSECURITY_CONFIG(SAD_FENG_SHUI) */
```

Randomly assigns allocation zones based on object type “signatures”.

Exploit needs to somehow compensate for this object grouping randomization

DATAFLOW SECURITY

Firebloom 🔥🌸

Memory safe iBoot implementation

In iOS 14 and iPadOS 14, Apple modified the C compiler toolchain used to build the iBoot bootloader to improve its security. The modified toolchain implements code designed to prevent memory- and type-safety issues that are typically encountered in C programs. For example, it helps prevent most vulnerabilities in the following classes:

- Buffer overflows, by ensuring that all pointers carry bounds information that is verified when accessing memory
- Heap exploitation, by separating heap data from its metadata and accurately detecting error conditions such as double free errors
- Type confusion, by ensuring that all pointers carry runtime type information that's verified during pointer cast operations
- Type confusion caused by use after free errors, by segregating all dynamic memory allocations by static type

This technology is available on iPhone with Apple A13 Bionic or later, and iPad with the A14 Bionic chip.

Published Date: February 18, 2021

Other mitigations…

- Can no longer use predictable addresses for objects with huge heap sprays
- PPL based read-only allocator
- Removal of primitives (i.e. mach voucher user data)
- Entitlements required to access IOKit userclient methods (i.e. WebContent)
- IOUserClient2022, kmem_guard_t
- …

DATAFLOW SECURITY

The Future

v8 (Maglev)

Maglev

Attention: Externally visible, non-confidential

Author: jgruber@chromium.org, leszeks@chromium.org, verwaest@chromium.org
Status: Draft | Final
Created: 2022-02-04
Tracking Bug: v8:7700
Link: go/v8-maglev

LGTMs needed

Name	Write (not) LGTM in this row
hpayer	LGTM
tebbi	LGTM
saelo	LGTM
<your name here>	

Note: This is a long doc and incomplete on the design details. LGTMs are for the summary and milestones.

TL;DR

Let's add a mid-tier optimising compiler designed mainly for compilation speed that can still generate good code for straightforward JS.

v8 (Turboshaft)

Issue 12783: Tracking bug: TurboShaft

Reported by tebbi@chromium.org on Thu, Apr 7, 2022, 8:46 PM GMT+9 **Project Member**

TurboShaft is a new CFG-based IR for TurboFan.

Show 37 older comments

Comment 38 by Git Watcher on Fri, Oct 28, 2022, 3:37 PM GMT+9 **Project Member**

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/bfda81d1a1faa7ab31a6b1a377b565d67b037907

commit bfda81d1a1faa7ab31a6b1a377b565d67b037907
Author: Tobias Tebbi <tebbi@chromium.org>
Date: Thu Oct 27 16:38:01 2022

[turboshaft] introduce SnapshotTable

v8 (Turboshaft)

Issue 12783: Tracking bug: TurboShaft

Reported by tebbi@chromium.org on Thu, Apr 7, 2022, 8:46 PM GMT+9 **Project Member**

TurboShaft is a new CFG-based IR for TurboFan.

Show 37 older comments

Comment 38 by Git Watcher on Fri, Oct 28, 2022, 3:37 PM GMT+9 **Project Member**

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/bfda81d1a1faa7ab31a6b1a377b565d67b037907

commit bfda81d1a1faa7ab31a6b1a377b565d67b037907
Author: Tobias Tebbi <tebbi@chromium.org>
Date: Thu Oct 27 16:38:01 2022

[turboshaft] introduce SnapshotTable

However some researchers may be going back to blink, due to various environmental changes

v8 MTE based Address Sanitizer

Memory Tagging and a JS address sanitizer

Attention - this doc is public and shared with the world!

Author: pierre.langlois@arm.com
Contacts: pierre.langlois@arm.com, martyn.capewell@arm.com
Status: Draft | Accepted | Done

LGTMs needed

Name	Write (not) LGTM in this rov
mstarzinger	
ulan	
ishell	
rmcilroy	
verwaest	
<your name here>	

Plans to land MTE on v8 allocator

DATAFLOW SECURITY

MTE in PartitionAlloc

Comment 19 by Git Watcher on Fri, Oct 15, 2021, 2:23 AM GMT+9 **Project Member**

The following revision refers to this bug:
https://chromium.googlesource.com/chromium/src/+/4c4507e2c48b15daf608660c12dea706d09e03d8

commit 4c4507e2c48b15daf608660c12dea706d09e03d8
Author: Richard Townsend <Richard.Townsend@arm.com>
Date: Thu Oct 14 17:22:19 2021

feat: basic MTE support for the partition allocator

This CL introduces minimal support for Arm's memory tagging extension (MTE) in the partition allocator. MTE is designed to improve memory safety by assigning each 16-byte region of memory one of 16 tags. This is used to detect temporal (e.g. double-free, use-after-free) and spatial memory safety violations by tagging each partition allocator slot on initialization, and then incrementing that tag upon each free.

ipcz

```
MojoIpcz: Enable by default on supported platforms

Enable MojoIpcz by default on platforms other than Chrome OS, macOS,
and Fuchsia. It's been enabled throughout M109 for 50% of the
population across all channels, it's stable, and it improves
WCV metrics consistently on Android (-1% FCP / -0.6% LCP).

Bug: 1299283
Change-Id: Ibd525a4c61c484e4adc17ce62b7bea3d2d11ef99
```

ipcz

Issue 1371860: Security: UAF in mojo::SimpleWatcher::Context in MojoIpcz feature (browser process)
Reported by 0xasn...@gmail.com on Thu, Oct 6, 2022, 9:27 PM GMT+9

VULNERABILITY DETAILS
UAF in mojo::SimpleWatcher::Context in MojoIpcz feature in browser process

Issue 1379831: Security: stack-buffer-overflow in mojo::core::ipcz_driver::ObjectBase::PeekBox(browser process)
Reported by 0xasn...@gmail.com on Mon, Oct 31, 2022, 10:07 PM GMT+9

VULNERABILITY DETAILS
stack-buffer-overflow in mojo::core::ipcz_driver::ObjectBase::PeekBox in browser process
another one(different frome ~~Issue 1371860~~) in MojoIpcz feature

Bug: 1299283

Issue 1400054: Bad-cast to mojo::core::ipcz_driver::ObjectBase from ipcz::ParcelWrapper in mojo::core::ipcz_driver::Object<mojo::core::ipcz_driver::DataPipe>::FromHandle
Reported by ClusterFuzz on Sat, Dec 10, 2022, 6:41 PM GMT+9 Project Member

Detailed Report: https://clusterfuzz.com/testcase?key=4847992038424576

Go back to old attack surfaces

Issue 1371859: stack-use-after-return in gpu::gles2::ProgramInfoManager::Program::UpdateES2
Reported by emily...@gmail.com on Thu, Oct 6, 2022, 9:26 PM GMT+9

Steps to reproduce the problem:
os:ubuntu 22.04
chrome version
Chromium 108.0.5327.0
Chromium 106.0.5231.2

1. ./chrome --disable-gpu http://localhost:8001/crash.html

rn or heap-buffer-overflow.

h triggered an underflow in the previous operations, resulting in oob on this line[0].

<input->name_length;
gram_info_manager.cc(401)] UpdateES2input->name_length:24
gram_info_manager.cc(401)] UpdateES2input->name_length:1
gram_info_manager.cc(401)] UpdateES2input->name_length:1852795251

er.cc;l=401?q=%20progra

Issue 1283040: Security: (Android) Heap buffer overflow Vulnerability May Can Cause Chrome Sandbox Escape to system_server on Pixel 6
Reported by pinkperfect2021@gmail.com on Tue, Dec 28, 2021, 10:52 PM GMT+9

Hi Team,

Please Note!!: This vulnerability is pixel 6 issue, but can cause chrome sandbox escape to system privilege directly, so please consider the visibility of this issue.

VULNERABILITY DETAILS

1, Chrome Sandbox process can access activity_service according to isolated_app selinux policy.
2, Chrome APK has declared android.permission.REORDER_TASKS permission
So, chrome sandbox process can call IActivityManager.moveTaskToFront interface of system_server through binder

In the moveTaskToFront of system_server, it has an argument Bundle, which will cause unparcel when processing the Bundle passed by sandbox process.

ActivityTaskManagerService.moveTaskToFront -> SafeActivityOptions.fromBundle(bOptions) -> ActivityOptions.fromBundle(bOptions) -> new ActivityOptions(bOptions):

```
public ActivityOptions(Bundle opts) {
    // If the remote side sent us bad parcelables, they won't get the
    // results they want, which is their loss.
    opts.setDefusable(true);
```

Diversify attack surface as response to MiraclePtr

DATAFLOW SECURITY

Separate network service process (isolated_app)

Android Network Service Sandbox

This document is PUBLIC. Comment open to chromium-dev@chromium.org.

Status: *Under Review*
Author: *rsesek@*
Reviewers: *stamp here*
Date: *2020-01-09*
Updated: *2021-11-12*

Overview

Summary
Run the Network Service in a sandboxed process on Android.

Platforms
Android

Team
rsesek@chromium.org, chrome-security-guts@google.com

Bugs
https://bugs.chromium.org/p/chromium/issues/detail?id=1262395

Still ongoing for 5+ years...

DATAFLOW SECURITY

Android and rust

In Android 13, about 21% of all new native code (C/C++/Rust) is in Rust. There are approximately 1.5 million total lines of Rust code in AOSP across new functionality and components such as Keystore2, the new Ultra-wideband (UWB) stack, DNS-over-HTTP3, Android's Virtualization framework (AVF), and various other components and their open source dependencies. These are low-level components that require a systems language which otherwise would have been implemented in C++.

Security impact

To date, there have been zero memory safety vulnerabilities discovered in Android's Rust code.

DATAFLOW SECURITY

GPU Process in Safari

Message: [GPU Process] [iOS] Enable GPU Process for DOM rendering on iOS
https://bugs.webkit.org/show_bug.cgi?id=236508
rdar://83437844

Reviewed by Jon Lee.

- FeatureFlags/WebKit-appletvos.plist:
- FeatureFlags/WebKit-ios.plist:
- FeatureFlags/WebKit-watchos.plist:
- Shared/WebPreferencesDefaultValues.cpp:

(WebKit::defaultUseGPUProcessForDOMRenderingEnabled):

Location: trunk/Source/WebKit
Files: 5 edited

DATAFLOW SECURITY

Lockdown mode

How Lockdown Mode protects your device

When Lockdown Mode is enabled, some apps and features will function differently, including:

- Messages - Most message attachment types are blocked, other than certain images, video, and audio. Some features, such as links and link previews, are unavailable.
- Web browsing - Certain complex web technologies are blocked, which might cause some websites to load more slowly or not operate correctly. In addition, web fonts might not be displayed, and images might be replaced with a missing image icon.
- FaceTime - Incoming FaceTime calls are blocked unless you have previously called that person or contact.
- Apple services - Incoming invitations for Apple services, such as invitations to manage a home in the Home app, are blocked unless you have previously invited that person.

Browser exploits non-existent in competitions

Welcome to Pwn2Own Vancouver for 2023! This year's event promises some exciting research as we have 19 entries targeting nine different targets - including ***two*** Tesla attempts. For this year's event, every round will pay full price, which means if all exploits succeed, we'll award over $1,000,000 USD. As always, we began our contest with a random drawing to determine the order of attempts. If you missed it, you can watch the replay here.

The complete schedule for the contest is below (all times Pacific [GMT -7:00]).

DATAFLOW SECURITY

MTE (announced on 2018.9.17)

Memory Tagging

Armv8.5-A incorporates a new feature called Memory Tagging. When Memory Tagging is in use, a tag is assigned to each memory allocation. All accesses to memory must be made via a pointer with the correct tag. Use of an incorrect tag is noted and the operating system can choose to report it to the user immediately, or to note the process in which it occurred, for later investigation.

For instance, in the diagram below the access to the memory at 0x8000 will work because the pointer, used to make access, has the same tag (represented by a color) as the memory being accessed. However, the access to 0x9000 will fail, because the pointer has a different tag to the memory.

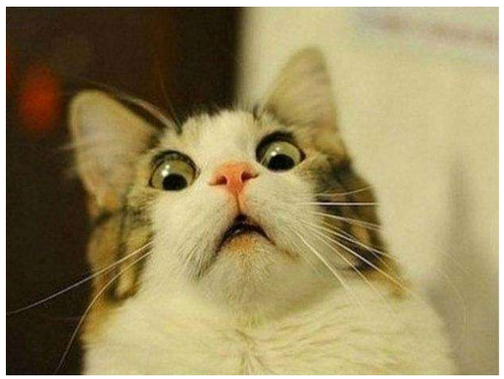
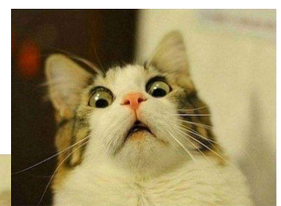
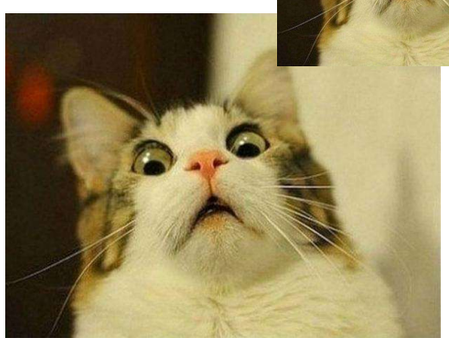
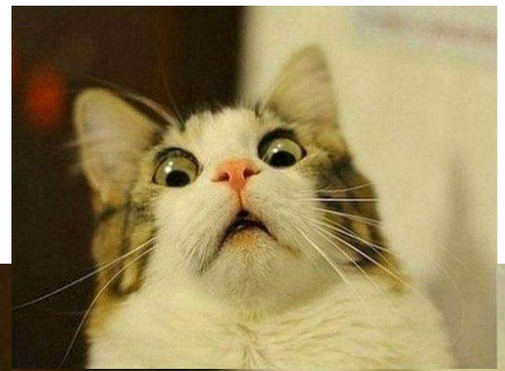
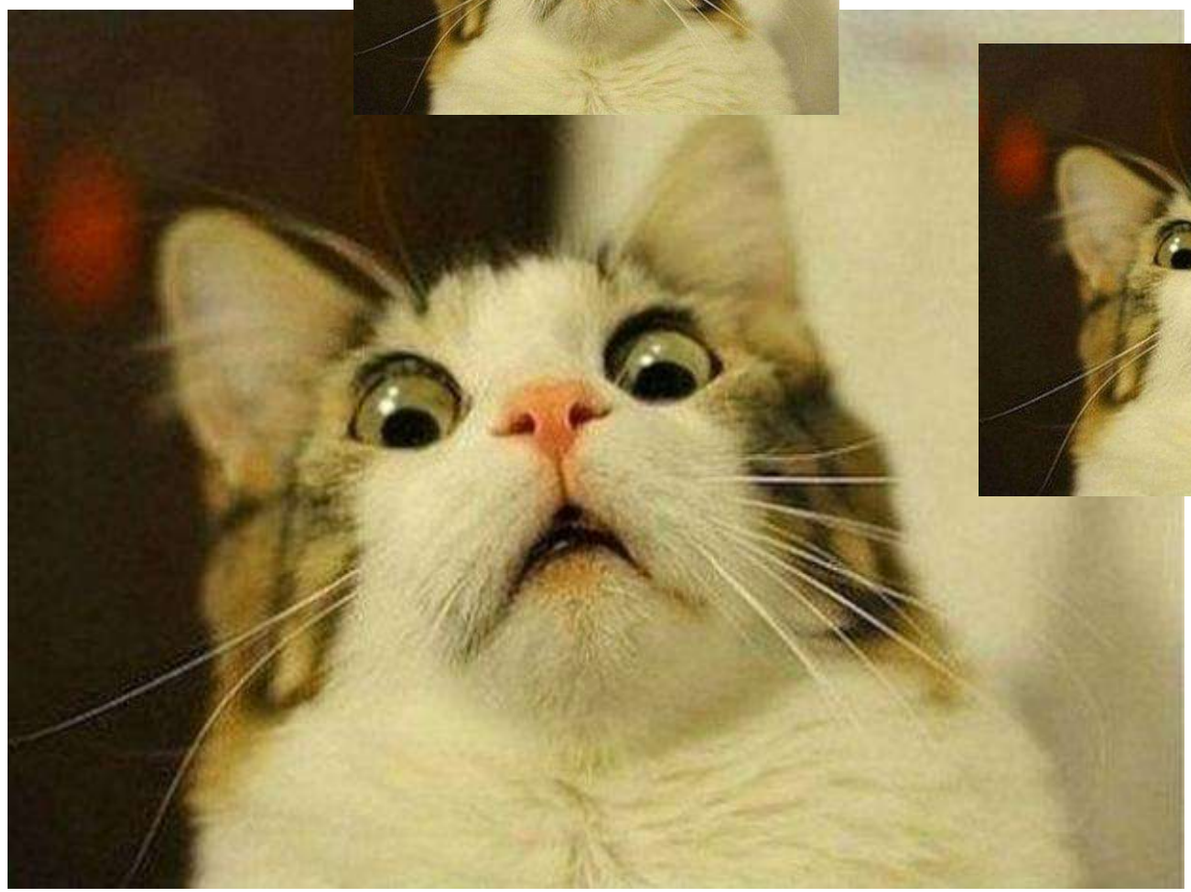
DATAFLOW SECURITY

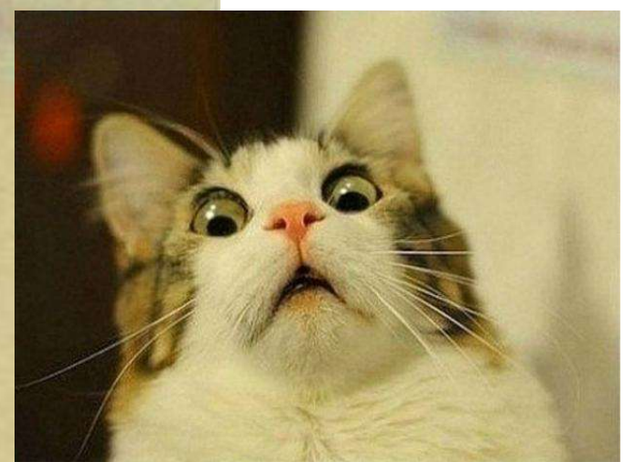
MTE (announced on 2018.9.17)

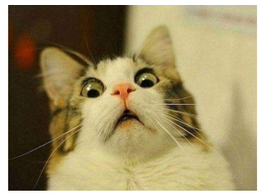
```
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
Build fingerprint: 'Android/fvp_mini/fvpbase:S/AOSP.MASTER/eng.kokmit.20210510.125925:eng/test-keys'
Revision: '0'
ABI: 'arm64'
Timestamp: 2021-04-27 03:28:27.555515850+0000
Cmdline: ./cjpeg-bad CVE-2020-13790
pid: 187, tid: 187, name: cjpeg-bad  >>> ./cjpeg-bad <<<
signal 11 (SIGSEGV), code 9 (SEGV_MTESERR), fault addr 0x600007a9b9e4f9f
Cause: [MTE]: Buffer Overflow, 104 bytes right of a 16151-byte allocation at 0x7a9b9e1020
    x0  0000000000003dc8  x1  0600007a9b9e2093  x2  0000000000002dd5  x3  ffffffffffffffff
    x4  ffffffffffffffff  x5  0600007a9b9e2093  x6  2929292929292929  x7  2929292929292929
    x8  0000000000000003  x9  0000000000000000  x10 0000000000000001  x11 0000000000000002
    x12 0600007a9b9e1262  x13 0100007a9b9e521f  x14 0000000000001402  x15 00000000000000ff
    x16 0000000000000000  x17 0000007afba807e4  x18 0000007afea4c000  x19 0000007fd7d2b8b8
    x20 0600007a9b9e1040  x21 0600007a9b9e4ea0  x22 0000000000000002  x23 0000000000000002
    x24 00000057d2238dc8  x25 00000057d2238e0c  x26 00000057d2238e50  x27 00000057d2238ed8
    x28 0000000000000008  x29 0000007fd7d2b750
    lr  00000057d2240c98  sp  0000007fd7d2b750  pc  00000057d2240ddc  pst 0000000020001000

backtrace:
      #00 pc 000000000000addc  /data/nativetest64/cjpeg/cjpeg-bad
      #01 pc 00000000000073b0  /data/nativetest64/cjpeg/cjpeg-bad
      #02 pc 000000000004842c  /apex/com.android.runtime/lib64/bionic/libc.so (__libc_init+96)
(BuildId: fecba5c6d9e898bdbe9f88fb1f8540d8)
```

DATAFLOW SECURITY

Panic ensues…






Attackers

- Year is 2018 : “We need to prepare for MTE next year”

DATAFLOW SECURITY

Attackers

- Year is 2018 : “We need to prepare for MTE next year”
- Year is 2019 : “We need to prepare for MTE next year”

DATAFLOW SECURITY

Attackers

- Year is 2018 : “We need to prepare for MTE next year”
- Year is 2019 : “We need to prepare for MTE next year”
- Year is 2020 : “We need to prepare for MTE next year”

DATAFLOW SECURITY

Attackers

- Year is 2018 : “We need to prepare for MTE next year”
- Year is 2019 : “We need to prepare for MTE next year”
- Year is 2020 : “We need to prepare for MTE next year”
- Year is 2021 : “We need to prepare for MTE next year”

Attackers

- Year is 2018 : “We need to prepare for MTE next year”
- Year is 2019 : “We need to prepare for MTE next year”
- Year is 2020 : “We need to prepare for MTE next year”
- Year is 2021 : “We need to prepare for MTE next year”
- Year is 2022 : “We need to prepare for MTE next year”

DATAFLOW SECURITY

Attackers

- Year is 2018 : “We need to prepare for MTE next year”
- Year is 2019 : “We need to prepare for MTE next year”
- Year is 2020 : “We need to prepare for MTE next year”
- Year is 2021 : “We need to prepare for MTE next year”
- Year is 2022 : “We need to prepare for MTE next year”
- …

DATAFLOW SECURITY

Attackers

- Year is 2018 : “We need to prepare for MTE next year”
- Year is 2019 : “We need to prepare for MTE next year”
- Year is 2020 : “We nee year”
- Year is 2021 : “We nee year”
- Year is 2022 : “We need to prepare for MTE next year”
- …

So…
Is MTE there yet? 🤔

DATAFLOW SECURITY

It’s already there, in the code

```
1246    auto* root = FromAddrInFirstSuperpage(object_addr);
1247    SlotSpan* slot_span = SlotSpan::FromObject(object);
1248    PA_DCHECK(FromSlotSpan(slot_span) == root);
1249
1250    uintptr_t slot_start = root->ObjectToSlotStart(object);
1251    PA_DCHECK(slot_span == SlotSpan::FromSlotStart(slot_start));
1252
1253  #if PA_CONFIG(HAS_MEMORY_TAGGING)
1254    if (PA_LIKELY(root->IsMemoryTaggingEnabled())) {
1255      const size_t slot_size = slot_span->bucket->slot_size;
1256      if (PA_LIKELY(slot_size <= internal::kMaxMemoryTaggingSize)) {
1257        // slot_span is untagged at this point, so we have to recover its tag
1258        // again to increment and provide use-after-free mitigations.
1259        internal::TagMemoryRangeIncrement(internal::TagAddr(slot_start),
1260                                          slot_size);
1261        // Incrementing the MTE-tag in the memory range invalidates the |object|'s
1262        // tag, so it must be retagged.
1263        object = internal::TagPtr(object);
1264      }
1265    }
```

It’s already there, in the code

arch > arm64 > kernel > C mte.c > ...

```
250     if (kasan_hw_tags_enabled())
251         *updptr = cpu_to_le32(aarch64_insn_gen_nop());
252 }
253
254 void mte_thread_init_user(void)
255 {
256     if (!system_supports_mte())
257         return;
258
259     /* clear any pending asynchronous tag fault */
260     dsb(ish);
261     write_sysreg_s(0, SYS_TFSRE0_EL1);
262     clear_thread_flag(TIF_MTE_ASYNC_FAULT);
263     /* disable tag checking and reset tag generation mask */
264     set_mte_ctrl(current, 0);
265 }
266
267 void mte_thread_switch(struct task_struct *next)
268 {
269     if (!system_supports_mte())
270         return;
271
272     mte_update_sctlr_user(next);
273     mte_update_gcr_excl(next);
```

DATAFLOW SECURITY

It’s already there, in the code

arch > arm64 > kernel > C mte.c > ...

```
250     if (kasan_hw_tags_enabled())
251         *updptr = cpu_to_le32(aarch64_insn_gen_nop());
252 }
253
254 void mte_thread_init_user(void)
255 {
256     if (!system_supports_mte())
257         return;
258
259     /* clear any pending asynchronous tag fault */
260     dsb(ish);
261     write_sysreg_s(0, SYS_TFSRE0_EL1);
262     clear_thread_flag(TIF_MTE_ASYNC_FAULT);
263     /* disable tag checking and reset tag generation mask */
264     set_mte_ctrl(current, 0);
265 }
266
267 void mte_thread_switch(struct task_struct *next)
268 {
269     if (!system_supports_mte())
270         return;
271
272     mte_update_sctlr_user(next);
273     mte_update_gcr_excl(next);
```

Just need to flip a build flag to turn it on (still actively being developed though)

Is there any hardware support yet…?

DATAFLOW SECURITY

MTE vendor adoption

What about silicon vendors and device manufacturers?

MTE is an intrinsic feature that is part of all of Arm's v9 CPUs. Multiple Arm partners who are committed to tackling memory safety bugs in the software ecosystem have already built and enable this feature across their chipsets. One device manufacturer who is leading the charge for MTE adoption is Honor, with the company announcing that its MTE enabled MagicOS 6.x and Magic OS 7 devices will be available to developers through its Honor SkyNet and in future DiagnosisKit tools. This is a significant indication that MTE could be switched on across mobile devices built on Armv9 technology that are coming to the consumer market.

Already this work is having a positive impact. Kuaishou – a leading content community and social platform for video sharing and live streams with over 360 million daily average users and 626 million monthly average users, making it the second-largest short video platform in the world – is partnering with Honor SkyNet and using Arm MTE to enhance the efficiency of memory safety across the development cycles of large-scale software projects. This has led to 90 percent of memory bugs being detected before release, alongside the following additional improvements:

MTE vendor adoption

What about silicon vendors and device manufacturers?

MTE is an intrinsic feature that is part of all of Arm's v9 CPUs. Multiple Arm partners who are committed to tackling memory safety bugs in the software ecosystem have already built and enable this feature across their chipsets. One device manufacturer who is leading the charge for MTE adoption is Honor, with the company announcing that its MTE enabled MagicOS 6.x and Magic OS 7 devices will be available to developers through its Honor SkyNet and in future DiagnosisKit tools. This is a significant indication that MTE could be switched on across mobile devices built on Armv9 technology that are coming to the consumer market.

Already this work is ha… video sharing and live streams with over … cond-largest short video platform in the w… f memory safety across the developmen… ng detected before release, alongside the f…

It seems to only be used to augment bug elimination during dev-cycle for now (MTE based HWAsan)

No signs of MTE on major vendors Pixel or Galaxy, iPhone (yet)

DATAFLOW SECURITY

MTE tag comparison failure mode

Trade-offs between lower overhead and more accuracy in reporting

You may be wondering what the performance overhead of MTE is. For this, the architecture provides both synchronous and asynchronous mode to report tag comparison failures. The following table compares the features of the two modes.

Synchronous Mode	Asynchronous Mode
Suitable for Development • Identify the precise instruction and address that caused failure • Slight performance impact	Suitable for Deployment • Acceptable performance overhead on production systems • Less precise information on where the failure occurred

Table 1: MTE's two modes to report tag comparison failures

MTE tag comparison failure mode

No idea what the runtime overhead would be yet.
It's plausible to think it'll be deployed as Asynchronous mode in runtime.

You may be wondering what the performance overhead of MTE is. For this, the architecture provides both synchronous and asynchronous mode to report tag comparison failures. The following table compares the features of the two modes.

Synchronous Mode	Asynchronous Mode
Suitable for Development • Identify the precise instruction and address that caused failure • Slight performance impact	Suitable for Deployment • Acceptable performance overhead on production systems • Less precise information on where the failure occurred

Table 1: MTE's two modes to report tag comparison failures

Attackers have been strategizing for 4+ years

- Strong infoleak bug to leak tags → MTE defeated
- CPU side channel attacks to leak tags (i.e. PACMAN)
- Exemption of tagging on certain edge-cases
- Attack surfaces that are not affected by MTE
- Logic bugs
- Some great insight on Saar Amar's presentation :
  "Security Analysis of MTE Through Examples – BlueHatIL 2022"

DATAFLOW SECURITY

Attackers have been strategizing for 4+ years

- Strong infoleak bug to leak tags → MTE defeated
- CPU side channel attacks to leak tags (i.e. PAC
- Exemption of tagging on certain edge-cases
- Attack surfaces that are not affected by MTE
- Logic bugs
- Some great insight on Saar Amar's presentation :
  "Security Analysis of MTE Through Examples – BlueHatIL 2022"

No doubt it's one of the strongest mitigations, but strong attackers will likely re-strategize and adapt (or shift to other domains)

DATAFLOW SECURITY

Some thoughts about exploit primitives…

DATAFLOW SECURITY

Some thoughts about exploit primitives…

- There was a time where attackers didn't take mitigations seriously, as one would just create a bypass and the mitigation is as good as non-existent.

- Mitigations start adding up and forming a castle. Some mitigations are laughable, others are very strong and requires quite an effort. Few mitigations would land to kill an entire bug class family.

- Disclose an exploit for public flex, then and would kill the primitives used in the exploit immediately. New mitigations make primitives more scarce and valuable.

DATAFLOW SECURITY

Some thoughts about exploit primitives…

- Now you don't really see researchers(in the mobile domain) dropping public exploits anymore, unless caught in the wild, or some rare drops from publicly disclosed bug reports

- Attackers are becoming more and more protective of exploit primitives or mitigation bypasses, and refrain from talking in public domain. Exploit primitive itself becomes very valuable.

- Cannot deny that mitigations increase costs for attackers, both manpower cost for research and maintenance overhead (Now everyone out there is teaming up for full chaining, less feasible as a one man team)

DATAFLOW SECURITY

Takeaways

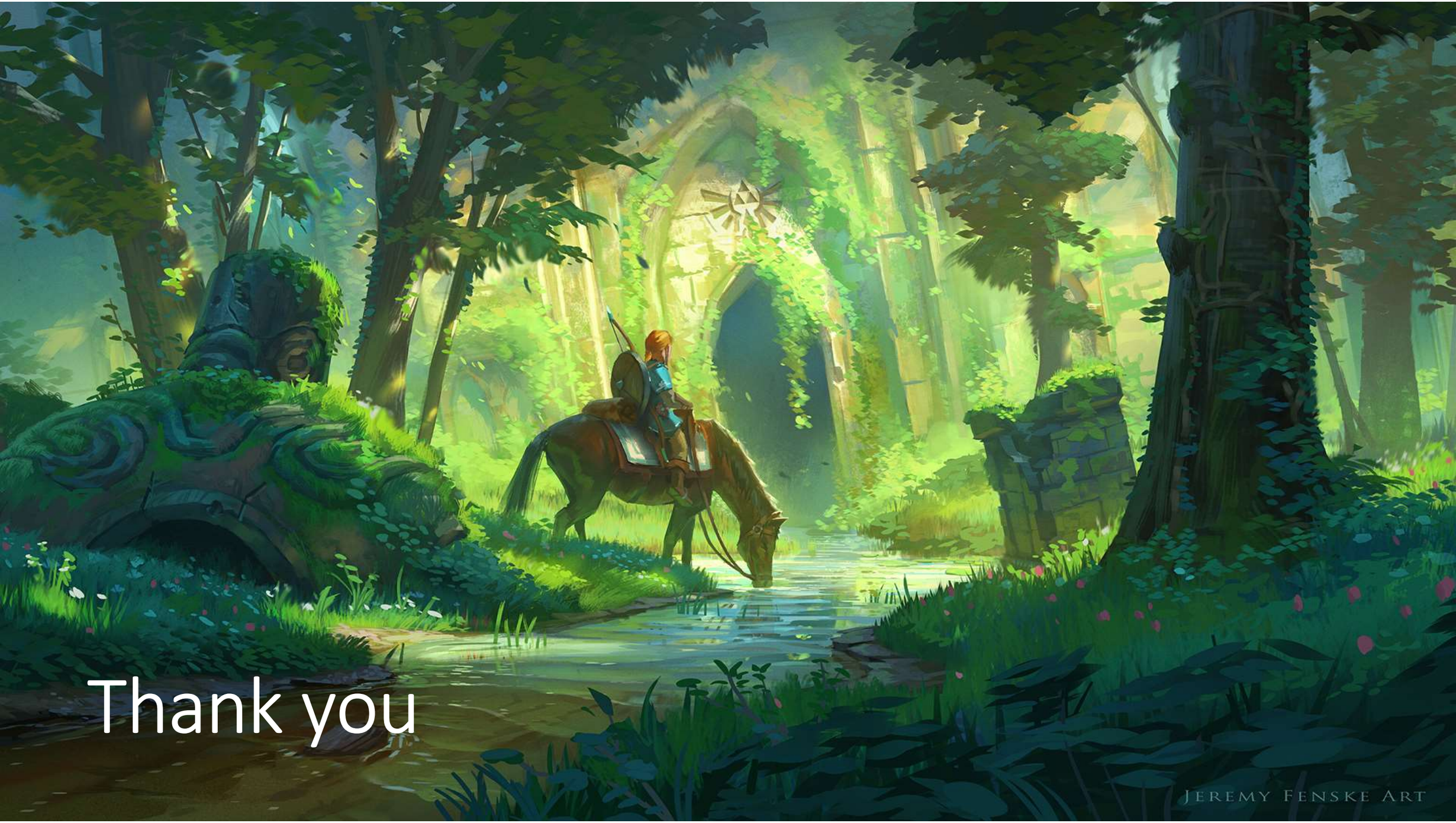
- Bughunting and exploitation becoming very challenging over time
- Strategic shift from attackers due to actions from vendors
- Curious how MTE would affect the landscape
- Recent mitigations make the game extremely challenging
- As Luca once said in a presentation… “We’re fighting a losing battle”, but attackers will re-strategize and adapt :)

DATAFLOW SECURITY

Special thanks for feedback

- Niklas
- Bruno
- And the entire Dataflow crew :)

DATAFLOW SECURITY


Thank you
JEREMY FENSKE ART