

ndays are also **0days**: Can hackers launch 0day RCE attack on popular software only with chromium ndays?

Bohan Liu, Zheng Wang, Guancheng Li

Tencent 腾讯



腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

> whoami



Bohan Liu

- @P4nda20371774
- Security Researcher at Tencent Security Xuanwu Lab
- Mainly Engaged in Browser Security
- Google Chrome Bug Hunter



Zheng Wang

- @xmzyshypnc
- Security Researcher at Tencent Security Xuanwu Lab
- Mainly Engaged in Browser Security and Kernel Security
- Found Several security bugs in Apple Safari, Linux kernel and VirtualBox



Guancheng Li

- @atuml1
- Security Researcher at Tencent Xuanwu Lab
- co-founder and former captain of r3kapig
- Previous focus on system/software security, currently focus on LLM x Security

Tencent 腾讯

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Agenda

- Introduction
- RCE-Window-period in Chromium-based software
- Attack Surface and Vulnerability Discovery
- Achieve RCE Without Sandbox Escape Bug
- Security Recommendations
- Conclusion & Takeaways



0. Introduction

Background

- I researched Chromium bugs for a while and wrote **some Nday exploits**.
- I know **little** web security.
- I can do **a little** reversing.



Can I find **0day RCE bugs** in popular software rather than Chrome?
How can I make it **faster and easier**?

How does chromium appears in software?



Chromium-based development



Electron framework



CEF



QtWebEngine

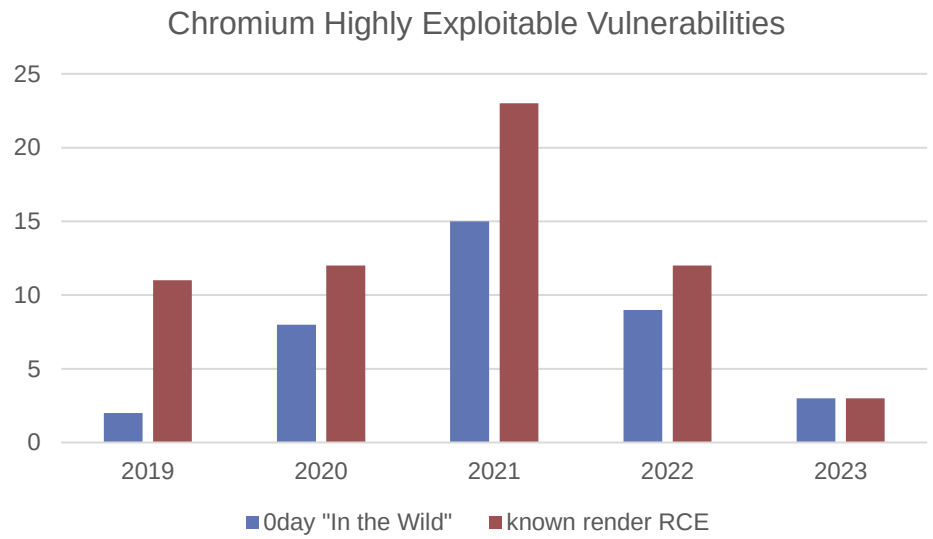


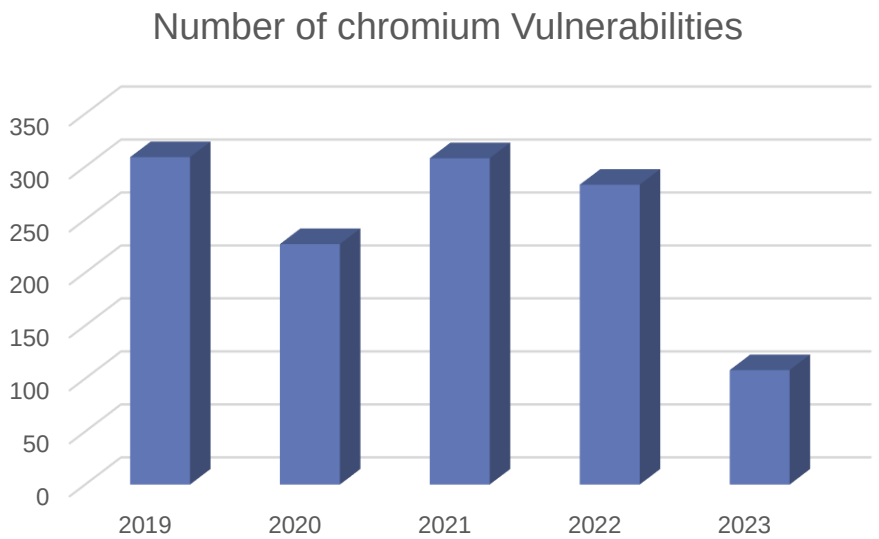
Other Supply Chain

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

The hidden dangers of chromium-based software

Chromium Highly Exploitable Vulnerabilities

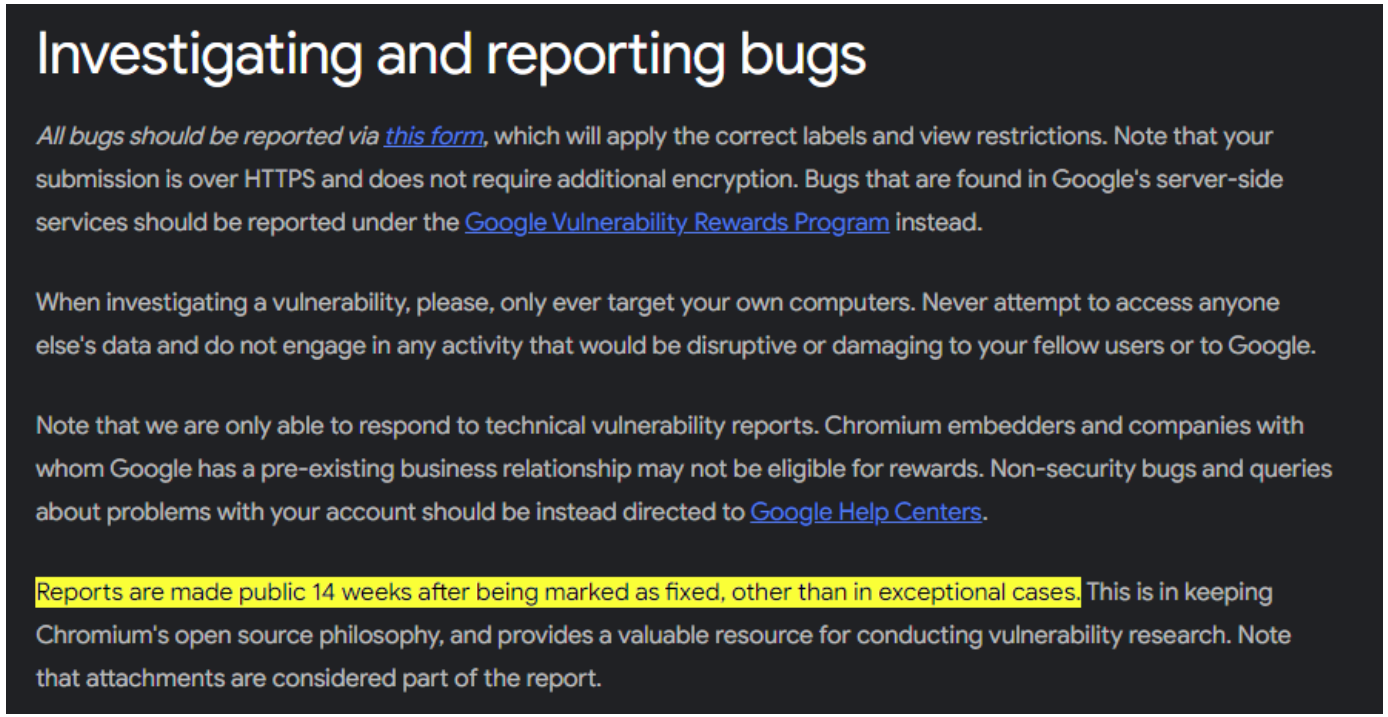




[1] https://www.cvedetails.com/product/15031/Google-Chrome.html?vendor_id=1224
[2] https://googleprojectzero.blogspot.com/p/0day.html

The hidden dangers of chromium-based software

- Google's vulnerability disclosure strategy



Investigating and reporting bugs

All bugs should be reported via this form, which will apply the correct labels and view restrictions. Note that your submission is over HTTPS and does not require additional encryption. Bugs that are found in Google's server-side services should be reported under the Google Vulnerability Rewards Program instead.

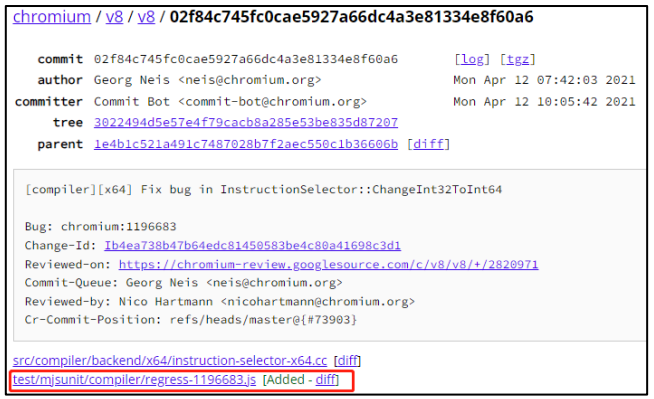
When investigating a vulnerability, please, only ever target your own computers. Never attempt to access anyone else's data and do not engage in any activity that would be disruptive or damaging to your fellow users or to Google.

Note that we are only able to respond to technical vulnerability reports. Chromium embedders and companies with whom Google has a pre-existing business relationship may not be eligible for rewards. Non-security bugs and queries about problems with your account should be instead directed to Google Help Centers.

Reports are made public 14 weeks after being marked as fixed, other than in exceptional cases. This is in keeping Chromium's open source philosophy, and provides a valuable resource for conducting vulnerability research. Note that attachments are considered part of the report.

[3] https://bughunters.google.com/about/rules/5745167867576320/chrome-vulnerability-reward-program-rules

The hidden dangers of chromium-based software



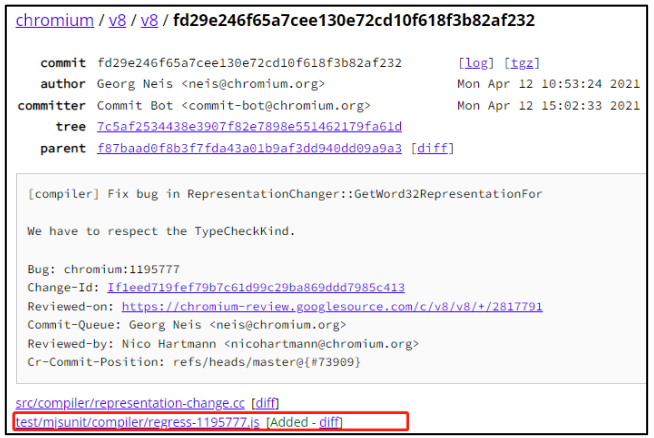
chromium / v8 / v8 / **02f84c745fc0cae5927a66dc4a3e81334e8f60a6**

commit 02f84c745fc0cae5927a66dc4a3e81334e8f60a6 [log] [tgz]
author Georg Neis <neis@chromium.org> Mon Apr 12 07:42:03 2021
committer Commit Bot <commit-bot@chromium.org> Mon Apr 12 10:05:42 2021
tree 3022494d5e57e4f79cacb8a285e53be835d87207
parent 1e4b1c521a491c7487028b7f2aec550c1b36606b [diff]

[compiler][x64] Fix bug in InstructionSelector::ChangeInt32ToInt64

Bug: chromium:1196683
Change-Id: Ib4ea738b47b64edc81450583be4c80a41698c3d1
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/2820971
Commit-Queue: Georg Neis <neis@chromium.org>
Reviewed-by: Nico Hartmann <nicohartmann@chromium.org>
Cr-Commit-Position: refs/heads/master@{#73903}

src/compiler/backend/x64/instruction-selector-x64.cc [diff]
test/mjsunit/compiler/regress-1196683.js [Added - diff]



chromium / v8 / v8 / **fd29e246f65a7cee130e72cd10f618f3b82af232**

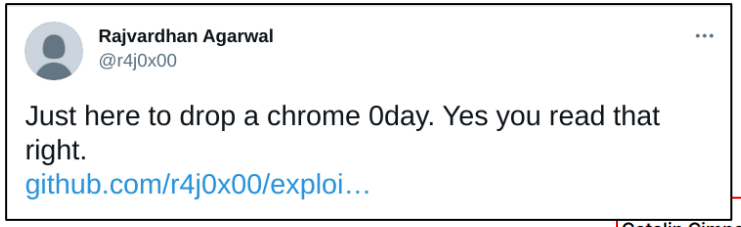
commit fd29e246f65a7cee130e72cd10f618f3b82af232 [log] [tgz]
author Georg Neis <neis@chromium.org> Mon Apr 12 10:53:24 2021
committer Commit Bot <commit-bot@chromium.org> Mon Apr 12 15:02:33 2021
tree 7c5af2534438e3907f82e7898e551462179fa61d
parent f87baad0f8b3f7fda43a01b9af3dd940dd09a9a3 [diff]

[compiler] Fix bug in RepresentationChanger::GetWord32RepresentationFor

We have to respect the TypeCheckKind.

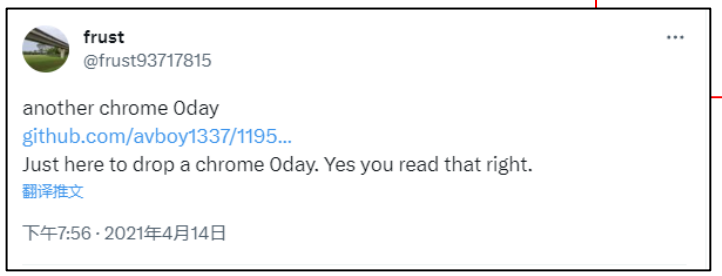
Bug: chromium:1195777
Change-Id: If1eed719fef79b7c61d99c29ba869ddd7985c413
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/2817791
Commit-Queue: Georg Neis <neis@chromium.org>
Reviewed-by: Nico Hartmann <nicohartmann@chromium.org>
Cr-Commit-Position: refs/heads/master@{#73909}

src/compiler/representation-change.cc [diff]
test/mjsunit/compiler/regress-1195777.js [Added - diff]



Rajvardhan Agarwal
@r4j0x00

Just here to drop a chrome 0day. Yes you read that right.
github.com/r4j0x00/exploi…



frust
@frust93717815

another chrome 0day
github.com/avboy1337/1195...
Just here to drop a chrome 0day. Yes you read that right.
翻译推文

下午7:56 · 2021年4月14日



Catalin Cimpanu
April 20th, 2021

News Technology

Recent Chromium bug used to attack Chinese WeChat users

A Chrome exploit published online last week has been weaponized and abused to attack WeChat users in China, a local security firm reported on Friday.

The attacks consisted of malicious links sent to WeChat users.

If users clicked the link, they'd trigger a piece of JavaScript code that would load and execute shellcode on their underlying operating systems.

Qingteng Cloud Security, which detected the attacks via one of its security products, said in a Weibo post that only users of the WeChat Windows app were attacked.

Hackers have lots of priori knowledge for chromium-based software attacks.



1. “RCE Window Period” in Chromium-based software

RCE Window Period in software

Definition : Patch gap between the fix of an exploitable bug on chromium and on software :

$$T_{RWPeriod=}\, t_{fix} - t_{exploitable} \qquad (1)$$

$$T_{UpdateGap=}\, t_{software_update} - t_{chromium_update} \qquad (2)$$







....







RCE Window Period in software

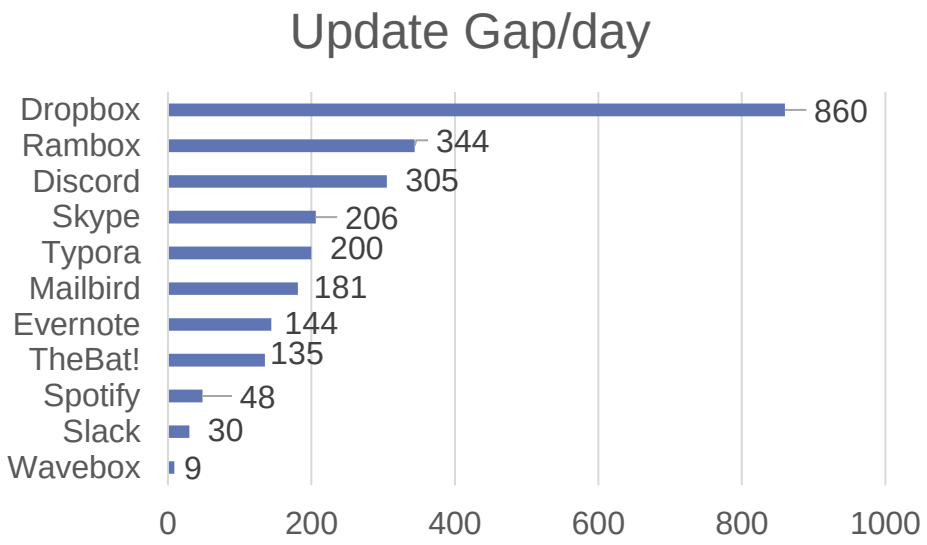
Method :

- Download the installation package for older versions
- Identify the type and version of the Chromium component in the software.
- Calculate the update cycle of software，the bug list and its RCE window period

RCE Window Period in software

Result :

The patch period varies from 9 to 800 days between software

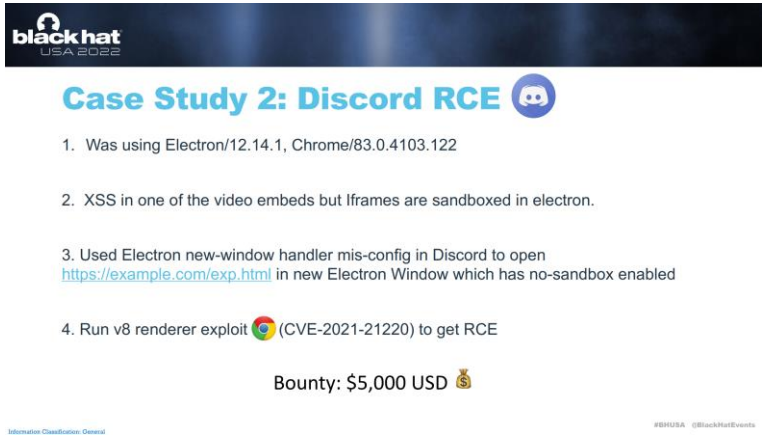


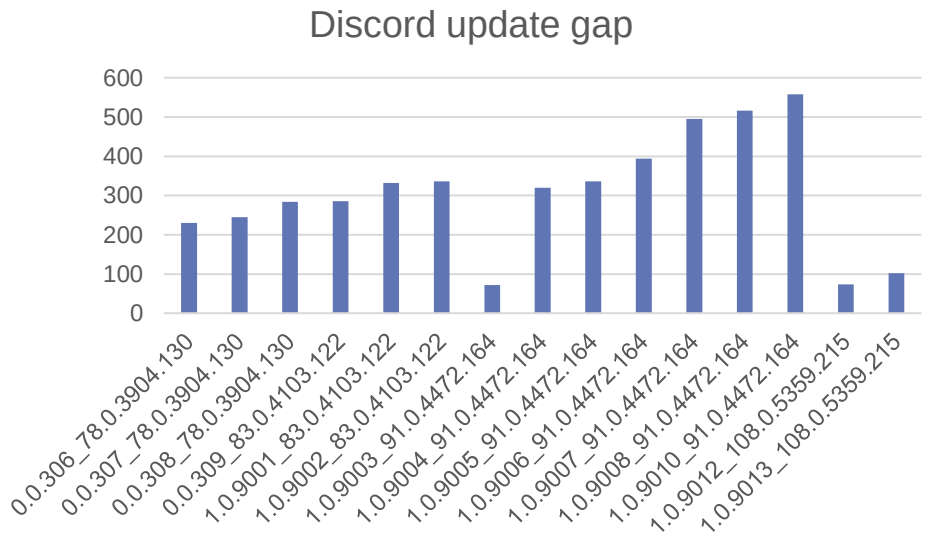
腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

RCE Window Period in software

Result :

For Discord, the update gap is as long as 350 days and there's lots of nday-exp for attackers to use.



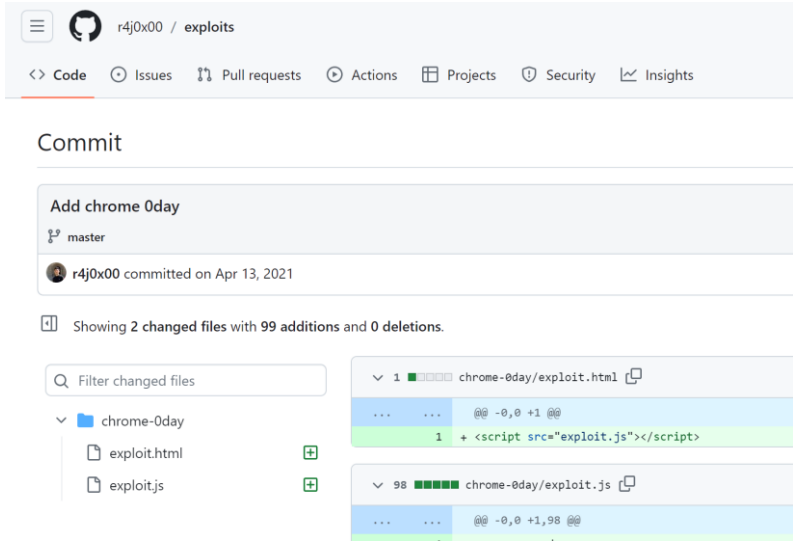


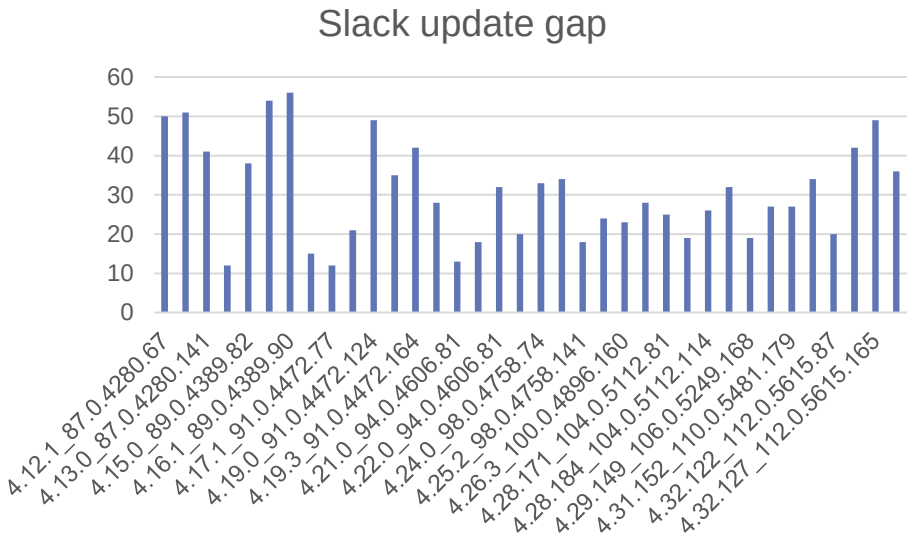
[4] https://blog.electrovolt.io/posts/discord-rce/

RCE Window Period in software

Result :

For Slack, the update gap is 30 days but it still at risk with unexpected 0-day disclosure





[5] https://github.com/r4j0x00/exploits/commit/7ba55e5ab034d05877498e83f144e187d3ddb160

RCE Window Period in software

Conclusion :

- The RCE Window Period exists universally in modern software
- The electron-based application updates more often than others
- Although many software tries to lower the patch period since a specific time, there’s still an opportunity for exploitation



2. Attack Surface and Vulnerability Discovery

Attack Vector



- Display existing web pages
- Trigger some client-side behaviors in specific scenarios.

Premise





During the “RCE window period”, it is possible to achieve **0-day** RCE when loading a page that hackers control.

Result



- Have many exploits with **N-day** vulnerabilities.
- Less limit, no domain-specific requirements

Advantage

What are the scenarios that developers tend to overlook?

Attack Vector



What are the scenarios that developers tend to overlook?

- What is the function of browser component?
- What code runs in the browser component?

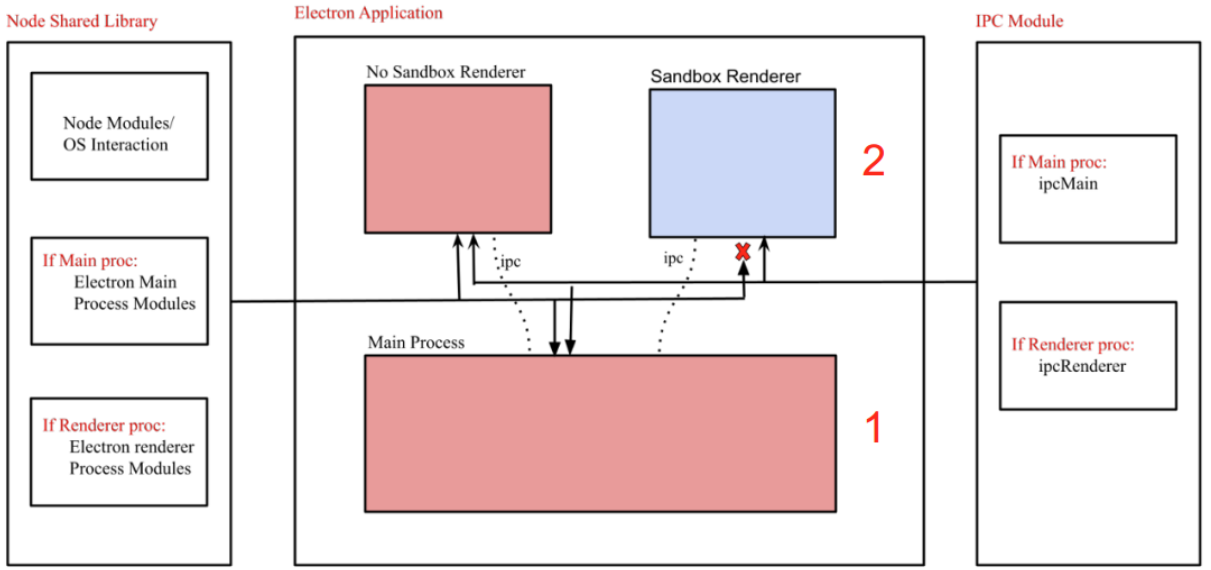


Let's take a look at how to develop it!

The development mode of the browser framework

Electron

- All codes are developed by Nodejs
- Divided into Main Process and Render process, communicate through IPC
- All code is stored in the resource folder
- The Electron framework is packaged in an executable file
- Less modification to Electron Framework code



[6] https://i.blackhat.com/USA-22/Thursd...sktop-Apps.pdf

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

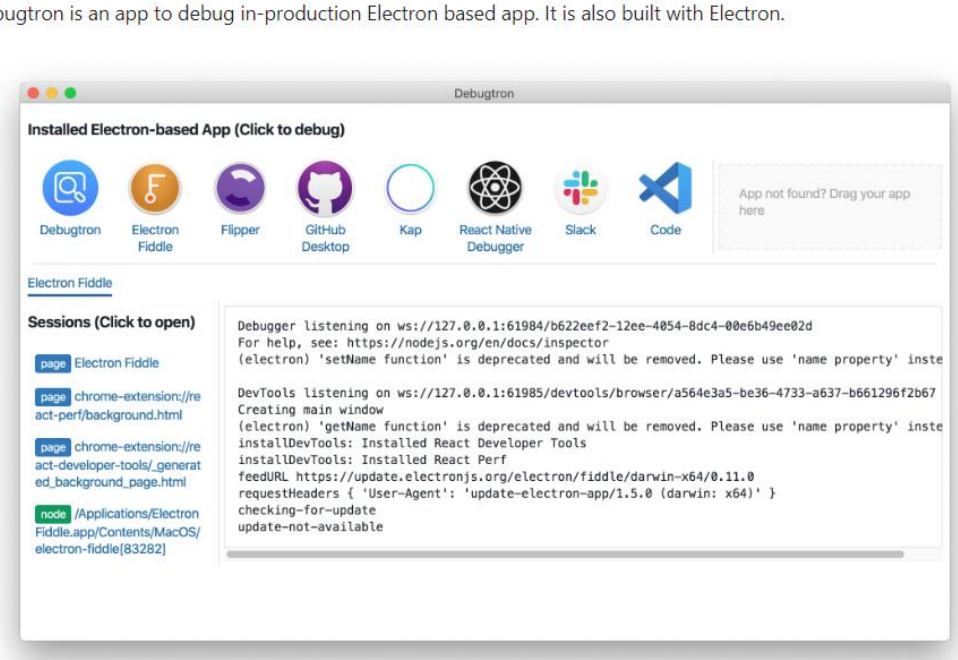
How to locate and debug

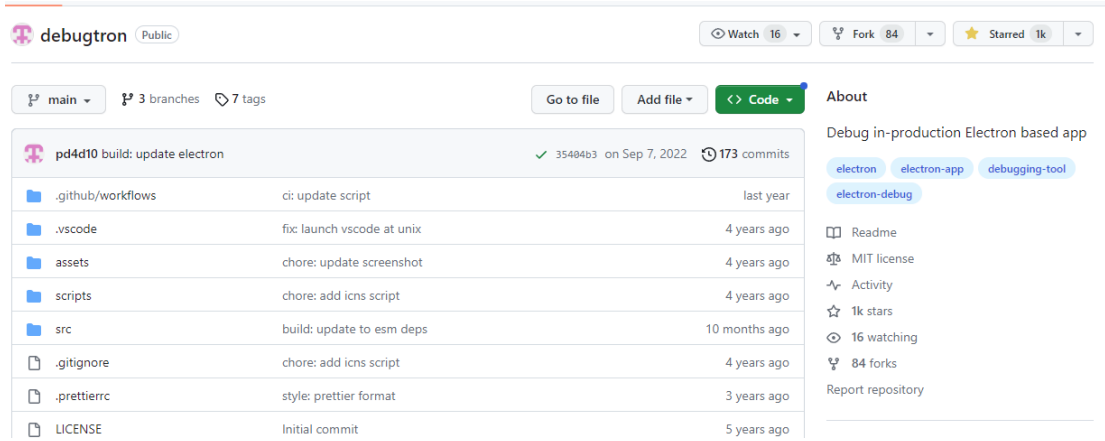
Electron

Locate：

asar extract app.asar xxx

Debug:



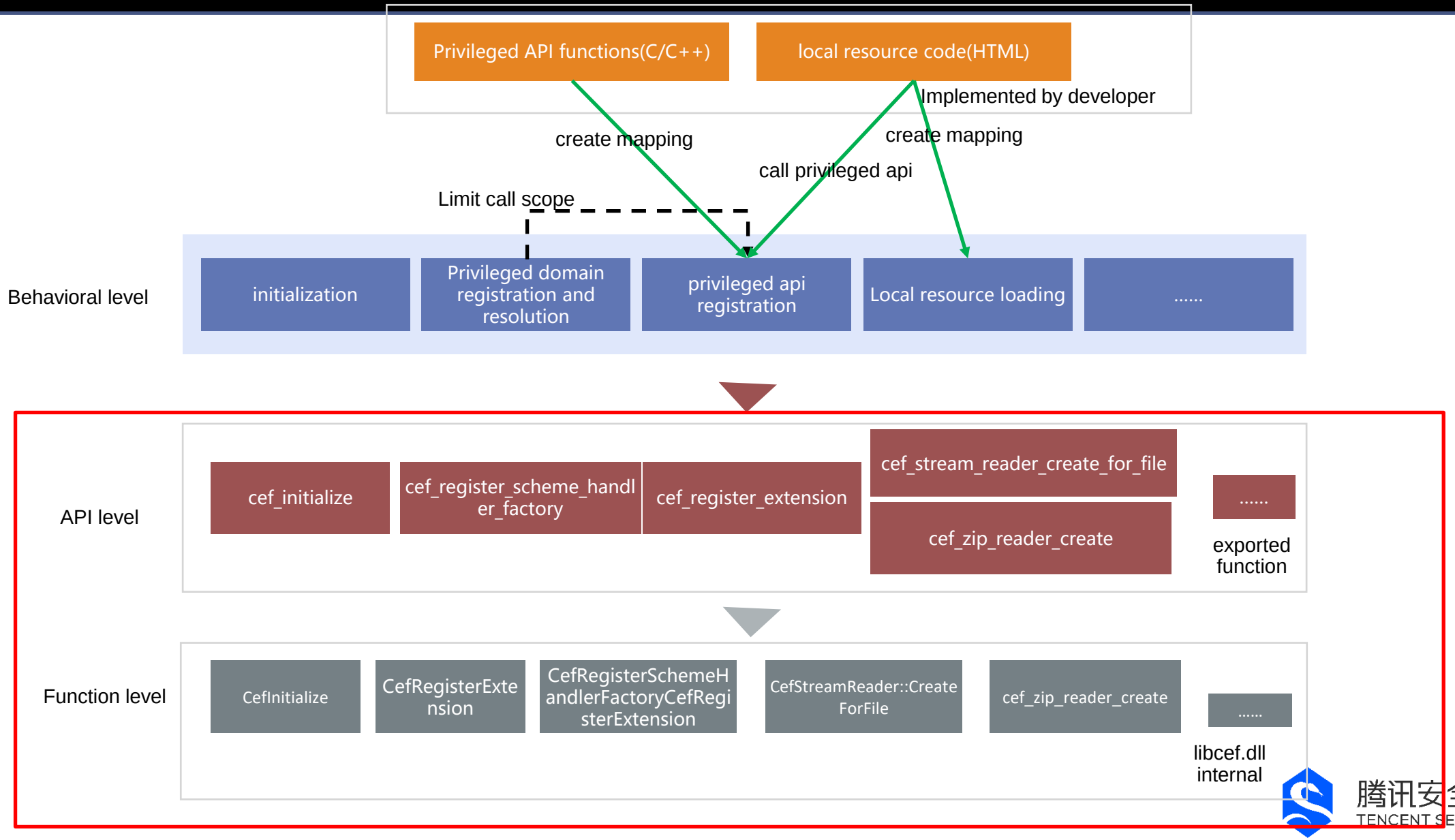



```
45  export const startDebugging =
46    (app: AppInfo): ThunkAction<any, State, any, any> =>
47    async (dispatch, getState) => {
48      const nodePort = await getPort()
49      const windowPort = await getPort()
50
51      const sp = spawn(
52        app.exePath,
53        [`--inspect=${nodePort}`, `--remote-debugging-port=${windowPort}`],
54        {
55          cwd: process.platform === 'win32' ? path.dirname(app.exePath) : '/',
56        }
57      )
58
59      const id = v4()
60      dispatch(addSession(id, app.id, nodePort, windowPort))
61
62      sp.on('error', (err) => {
63        dialog.showErrorBox(`Error: ${app.name}`, err.message)
64      })
65
66      sp.on('close', (code) => {
67        // console.log(`child process exited with code ${code}`)
68        dispatch(removeSession(id))
69        // TODO: Remove temp app
70      })
71
```

[7] https://github.com/pd4d10/debugtron

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

CEF Architecture



腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

How to locate and debug

CEF

- Privileged API is implemented by C/C++, local resource code is implemented by HTML.
- Exists in the form of a dynamic link library.
- The location of the local resource file is uncertain, and the file may be encrypted.
- Flexible usage of CEF and possible modifications to the library.



Any common debugging method?



Hook the API in libcef,
get and modify what we want

How to locate and debug

Locate

- Development mode:
  cef_stream_reader_create_for_file + cef_zip_reader_create + zip_reader_open_file

```
NO_SANITIZE("cfi-icall")
struct _cef_stream_reader_t* cef_stream_reader_create_for_file(
    const cef_string_t* fileName) {
  return g_libcef_pointers.cef_stream_reader_create_for_file(fileName);
}
```

```
NO_SANITIZE("cfi-icall")
struct _cef_zip_reader_t* cef_zip_reader_create(
    struct _cef_stream_reader_t* stream) {
  return g_libcef_pointers.cef_zip_reader_create(stream);
}
```

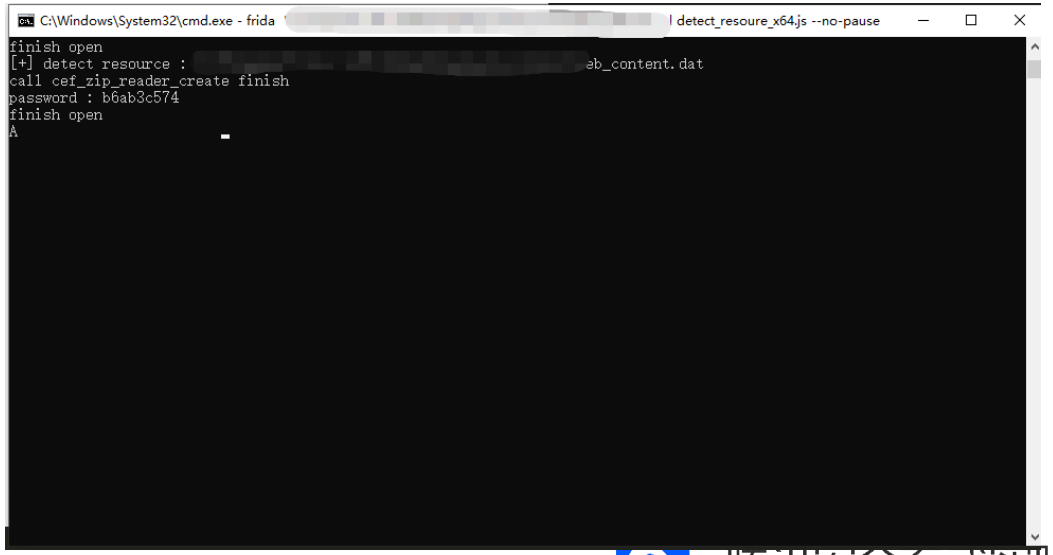
```
int CEF_CALLBACK zip_reader_open_file(struct _cef_zip_reader_t* self,
                                      const cef_string_t* password) {
  shutdown_checker::AssertNotShutdown();

  // AUTO-GENERATED CONTENT - DELETE THIS COMMENT BEFORE MODIFYING

  DCHECK(self);
  if (!self) {
    return 0;
  }
  // Unverified params: password

  // Execute
  bool _retval = CefZipReaderCppToC::Get(self)->OpenFile(CefString(password));

  // Return type: bool
  return _retval;
}
```



腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

How to locate and debug

Debug

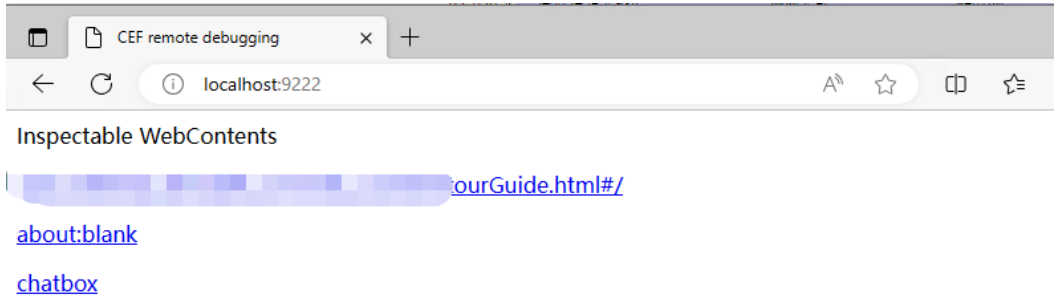
- Development mode:

cef_initialize with static configuration parameters

```
NO_SANITIZE("cfi-icall")
int cef_initialize(const cef_main_args_t* args,
                   const struct _cef_settings_t* settings,
                   struct _cef_app_t* application,
                   void* windows_sandbox_info) {
  return g_libcef_pointers.cef_initialize(args, settings, application,
                                          windows_sandbox_info);
}
```

```
160  typedef struct _cef_settings_t {
161    ///
162    /// Size of this structure.
163    ///
164    size_t size;
165
166    ///
167    /// Set to true (1) to disable the sandbox for sub-processes. See
168    /// cef_sandbox_win.h for requirements to enable the sandbox on Windows. Also
169    /// configurable using the "no-sandbox" command-line switch.
170    ///
171    int no_sandbox;
```

```
374    ///
375    /// Set to a value between 1024 and 65535 to enable remote debugging on the
376    /// specified port. Also configurable using the "remote-debugging-port"
377    /// command-line switch. Remote debugging can be accessed by loading the
378    /// chrome://inspect page in Google Chrome. Port numbers 9222 and 9229 are
379    /// discoverable by default. Other port numbers may need to be configured via
380    /// "Discover network targets" on the Devices tab.
381    ///
382    int remote_debugging_port;
383
```



We can also build more powerful debug tools based on development mode

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 1: Whitelist of allowed loads

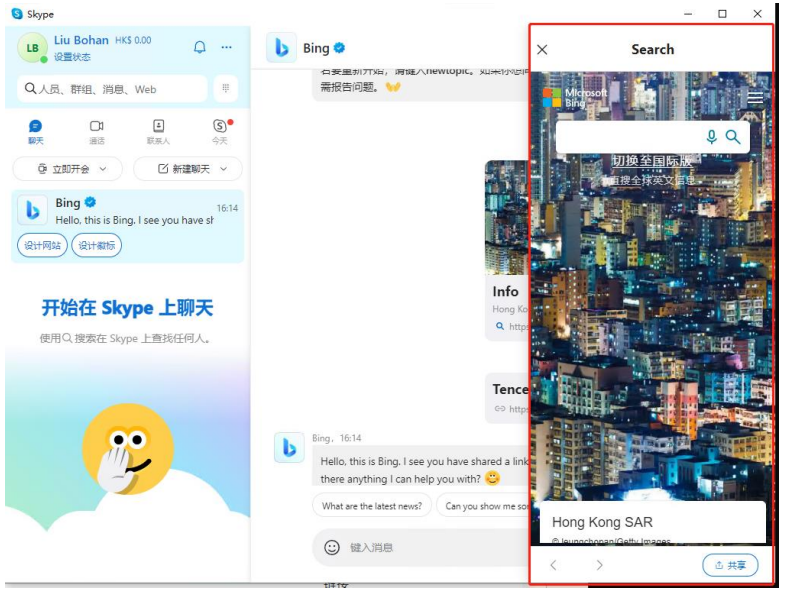
Development logic:

User clicks on URL

→ Check if URL’s domain is in whitelist?

→ if yes, load it in the software's internal browser component

→ if not, Call shellExecute("open",url,…) to open the URL with default browser

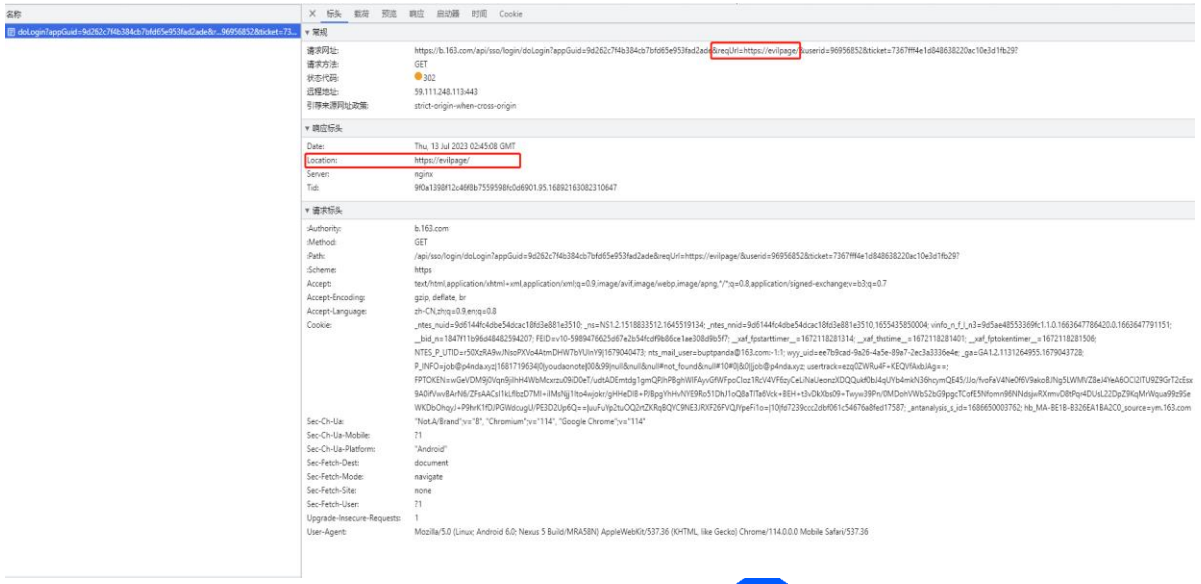


Scenario 1: Whitelist of allowed loads

How to bypass?

- Bypass check via Page redirection
  - Hyperlink navigation
  - 30X redirection
  - JavaScript navigation



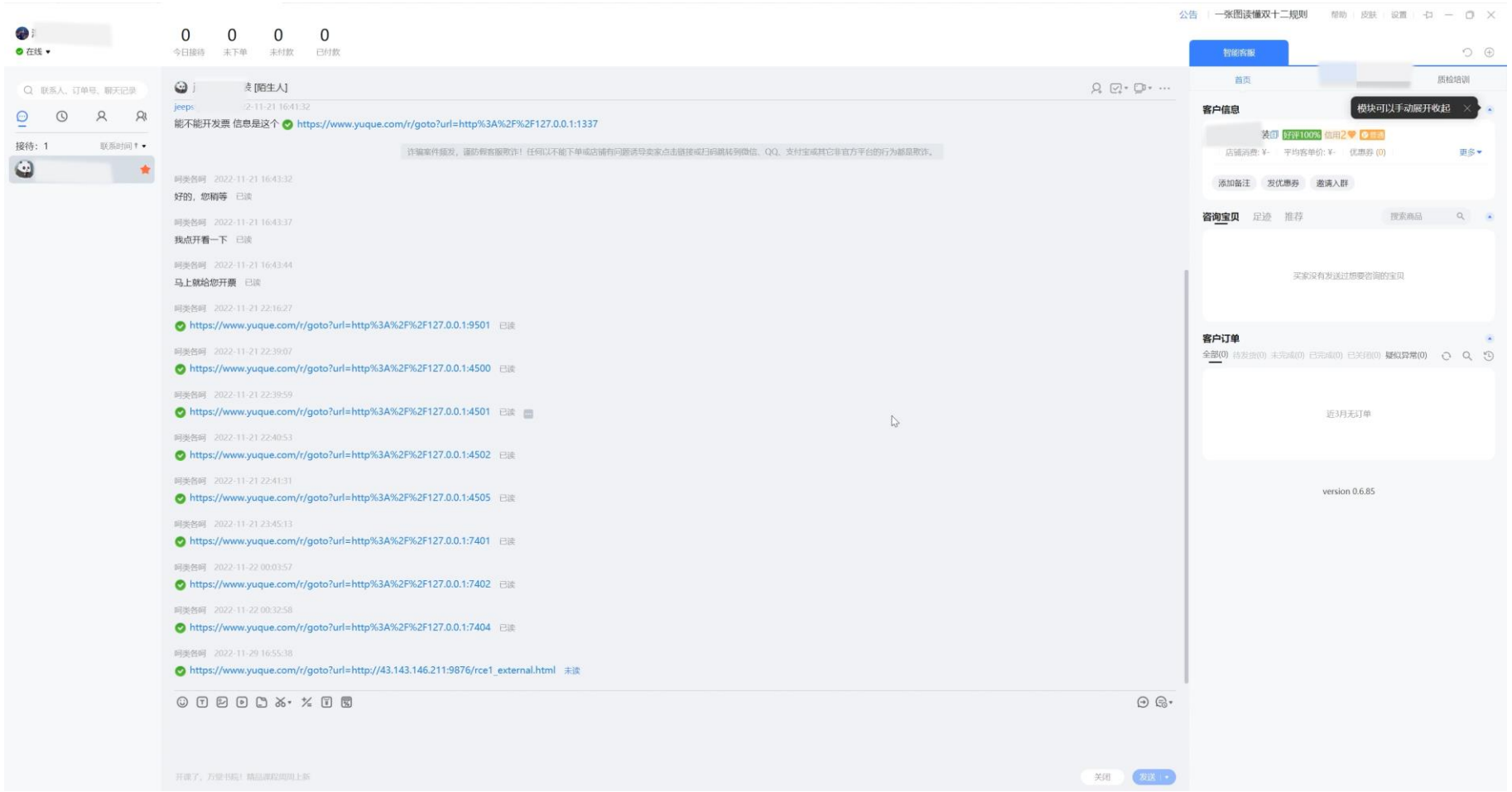


腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 1: Whitelist of allowed loads

How to bypass?

- Bypass check via Page redirection （demo）



腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 1: Whitelist of allowed loads

How to bypass?

- Incomplete whitelist checking
  - Incorrect regular expression
  - Incomplete Triple Check

```
int FilterUrl()
{
  JMUrlUtils::g_strRegJdUrl = QString::fromAscii_helper(
                              "(^[\\S]*(jd|jdpay|jcloud|jdcloud|jdwl|360buyimg|kuaidi100|300hu)\\.com[\\S]*$)|(^[\\S]*("
                              "(3|jdl|wjx)\\.cn|jd\\.hk)[\\S]*$)",
                              114);
  return sub_100018B6(sub_100278E0);
}
```

GP

"(^[\S]*(jd|jdpay|jcloud|jdcloud|jdwl|360buyimg|kuaidi100|300hu)\.com[\S]*$)|(^[\S]*((3|jdl|wjx)\.cn|jd\.hk)[\S]*$)" What strings can this regular expression match?

The given regular expression:

`(^[\S]*(jd|jdpay|jcloud|jdcloud|jdwl|360buyimg|kuaidi100|300hu)\.com[\S]*$)|(^[\S]*((3|jdl|wjx)\.cn|jd\.hk)[\S]*$)`

This regular expression can match strings that fall into the following patterns:

1. Any string that contains one of the following substrings, followed by the ".com" domain extension:
   - jd
   - jdpay
   - jcloud
   - jdcloud
   - jdwl
   - 360buyimg
   - kuaidi100
   - 300hu

   For example: "examplejd.com", "abcjdpay.com", "xyz360buyimg.com"
2. Any string that contains one of the following substrings, followed by either ".cn" or "jd.hk":
   - 3
   - jdl
   - wjx

   For example: "example3.cn", "abcjd.hk", "xyzjdl.cn"

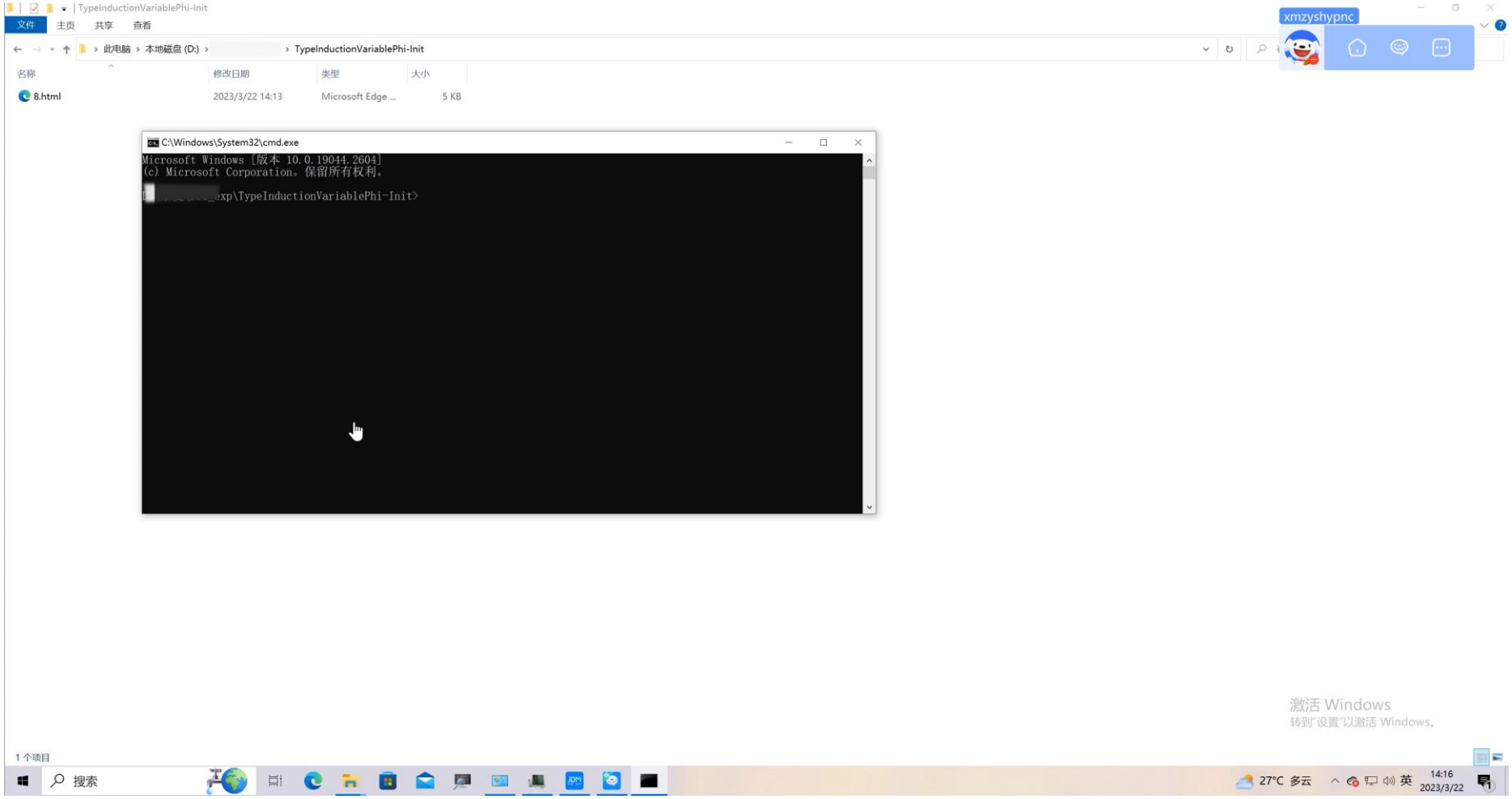
Please note that the regular expression matches patterns in the given strings but does not guarantee that the matched strings are valid URLs or exist in reality.

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 1: Whitelist of allowed loads

How to bypass?

- Incomplete whitelist checking (demo)

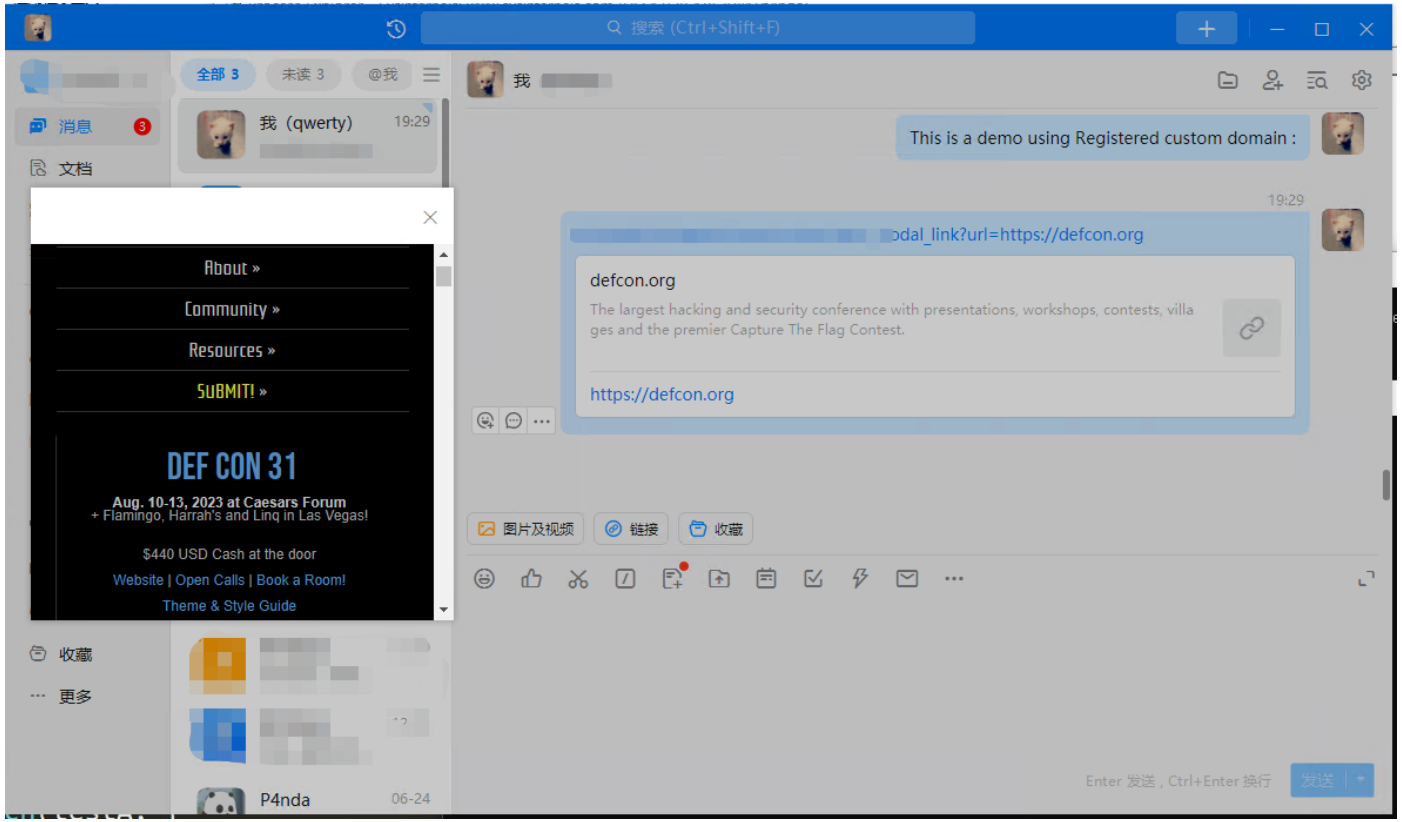


腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 1: Whitelist of allowed loads

How to bypass?

- Special URL handler
  - Scheme
  - Registered custom domain

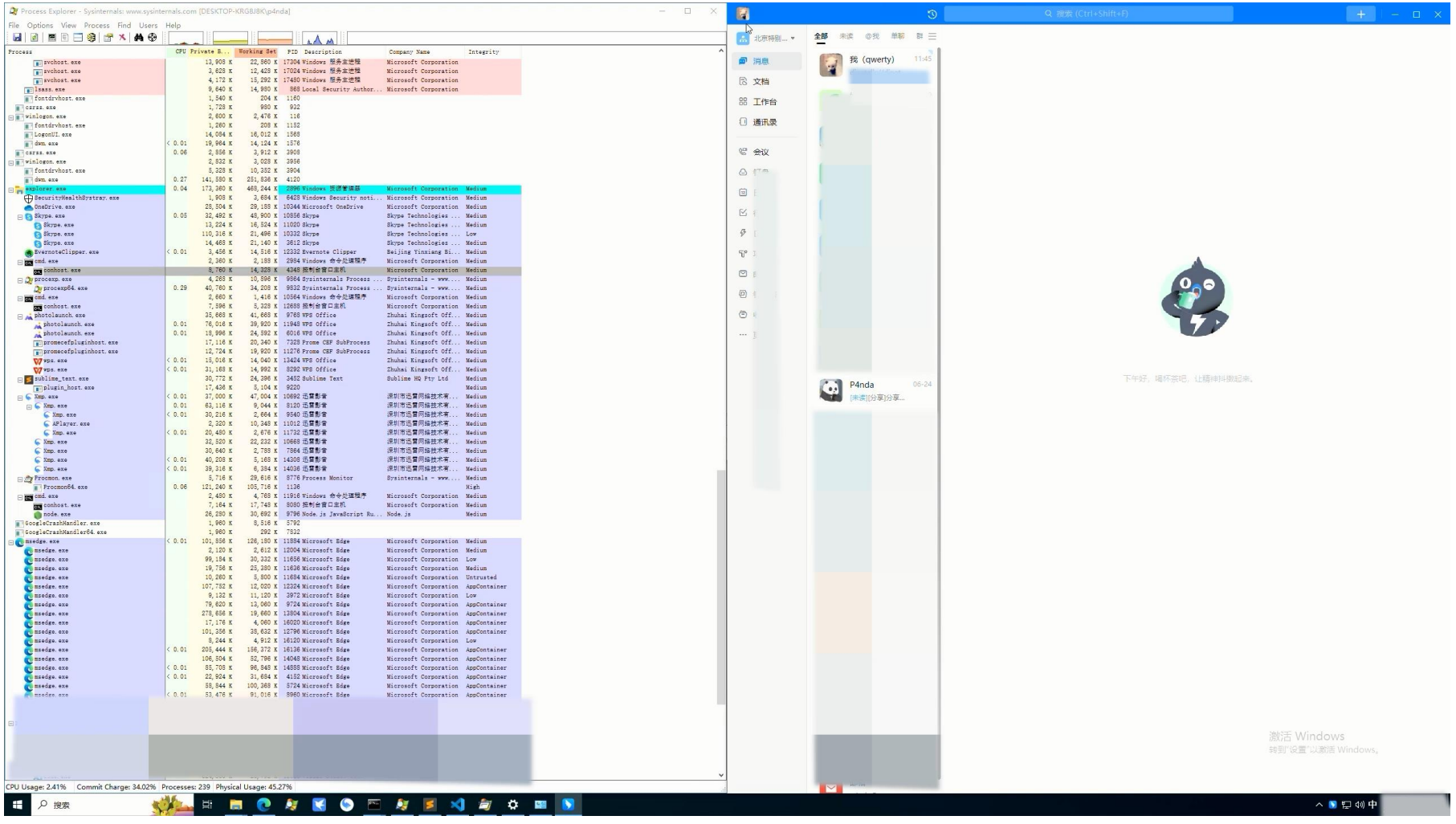


腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 1: Whitelist of allowed loads

How to bypass?

- Special URL handler (demo)



腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 2: Page code injection

What's the difference of XSS between Web Security and chromium-based software?

- XSS type
- Vulnerability discovery method
- Input source

Scenario 2: Page code injection

Local Resource XSS

- Decrypt Local resource
- Find file without CSP
- Find controlled input
- Trigger XSS

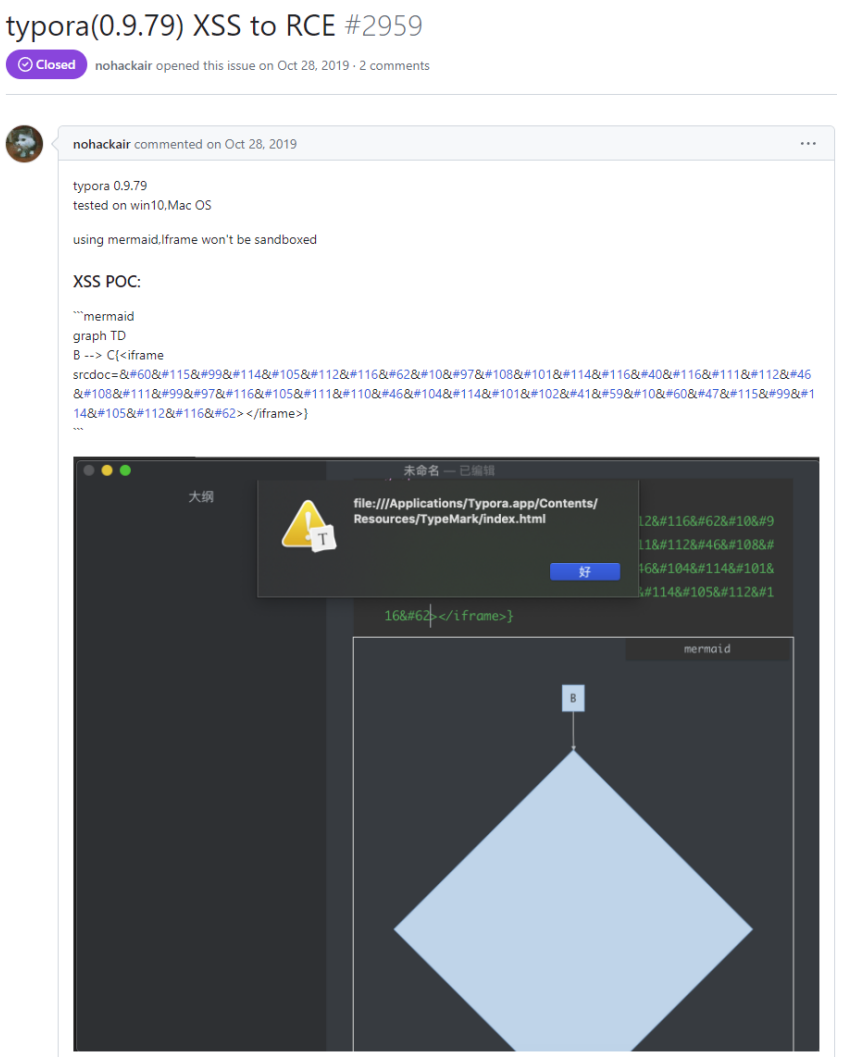
```
var ErrorPageObj = {
  reloadIfOnline: function() {
    window.addEventListener("online",function(e){errorpageobj.refreshPage();});
  },
  // ...
  getUrlParam: function(name) {
    const reg = new RegExp(`&?${name}=([^&#]*)`);
    const r = window.location.href.match(reg);     1
    if (r != null) return window.decodeURI(r[1]);
    return '';
  },
  setDarkColor: function() {
    var theme = this.getUrlParam("theme");
    // ...
  },
  setErrorInfo: function() {
      var encodeJsonString = this.getUrlParam("errorInfo");
      var jsonString = decodeURIComponent(encodeJsonString);
      var errorInfo = JSON.parse(jsonString);
      document.title = errorInfo["pageTitle"] || "";
      window.document.getElementsByClassName('msg')[0].innerHTML = errorInfo["msg"] || "";
  }
}
                                                    2
```

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 2: Page code injection

Third-Party Library XSS (case study)

- Mermaid XSS affecting Typora
- Used to be exploited by privileged API
- Using a Chromium Nday (CVE-2017-5070)



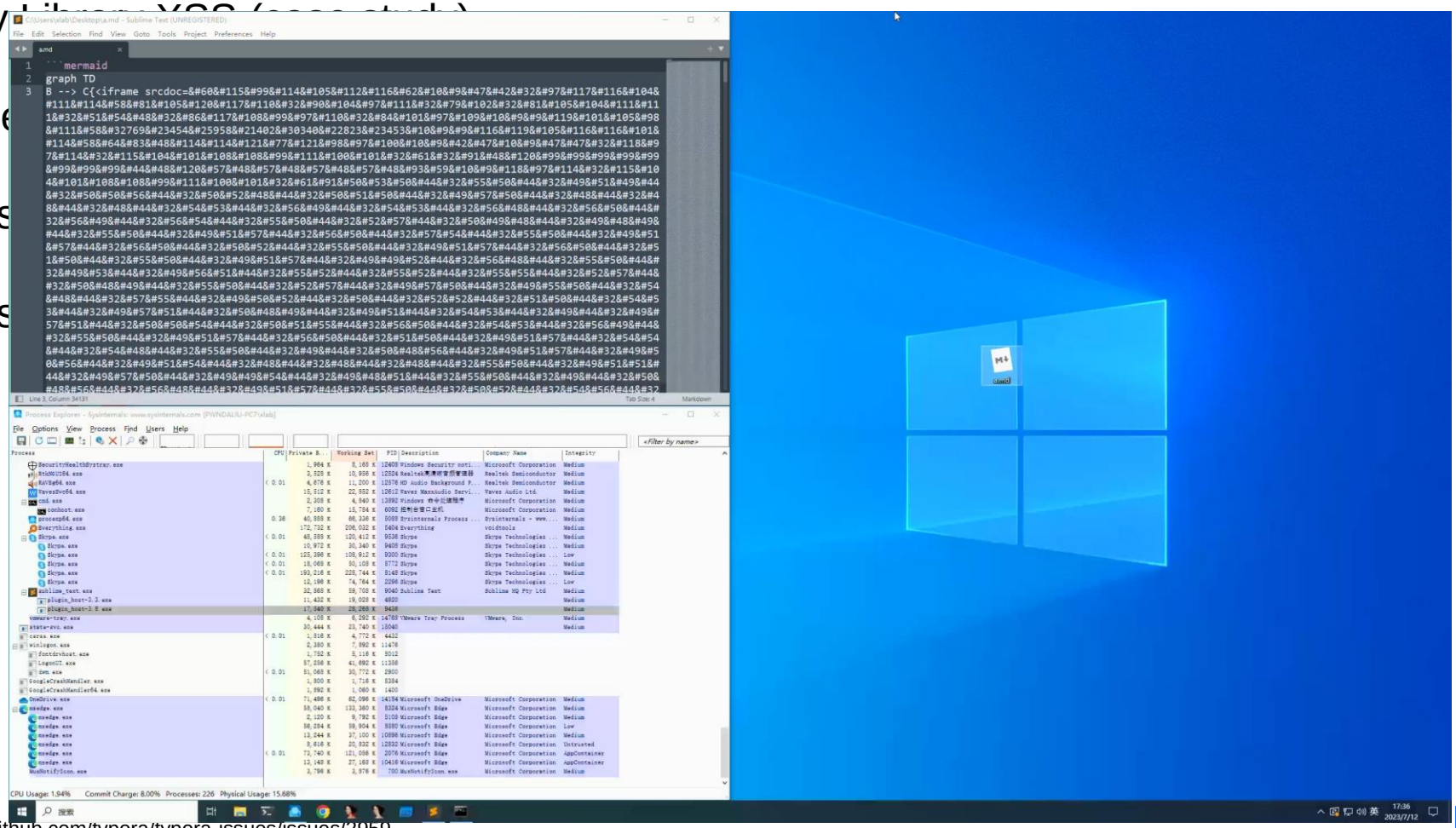
[9] https://github.com/typora/typora-issues/issues/2959
[10] https://bugs.chromium.org/p/chromium/issues/detail?id=722756

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 2: Page code injection

Third-Party Library XSS (mermaid)

- Me
- Us
- Us


mermaid
graph TD
B --> C{<iframe srcdoc=<script>
	/* author:Qixun Zhao Of Qiho 360 Vulcan Team
	*/
	author: shellcoders.io
	*/
});
	var script = [0xccccccc,0x90909090];
	var shell = [252, 72, 131, 228, 240, 232, 192, 0, 81, 65, 80, 82, 81, 86, 72, 139, 82, 96, 72, 139, 82, 24, 72, 139, 82, 72, 15, 183, 74, 77, 49, 201, 70, 49, 192, 1, 60, 97, 124, 2, 44, 32, 65, 193, 226, 201, 13, 65, 81, 93, 226, 82, 37, 82, 65, 81, 72, 193, 82, 208, 136, 80, 0, 82, 192, 1, 60, 97, 124, 2, 44, 32, 65, &#49

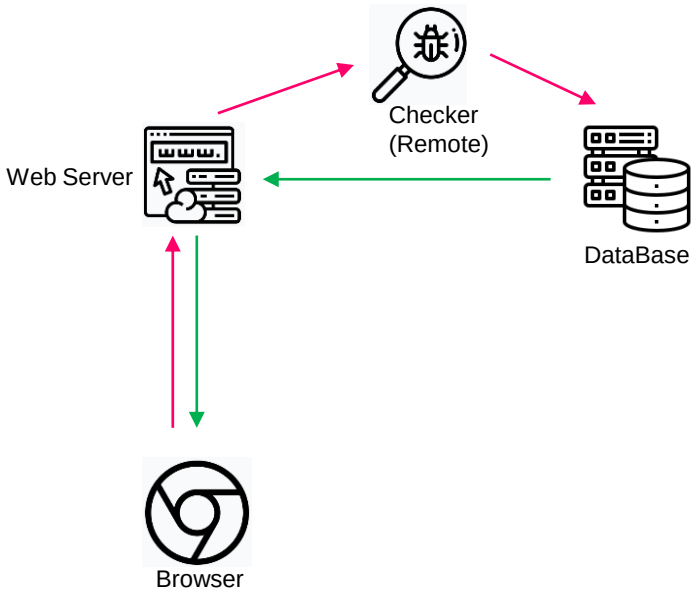
Scenario 2: Page code injection

Iframe in chromium-based software

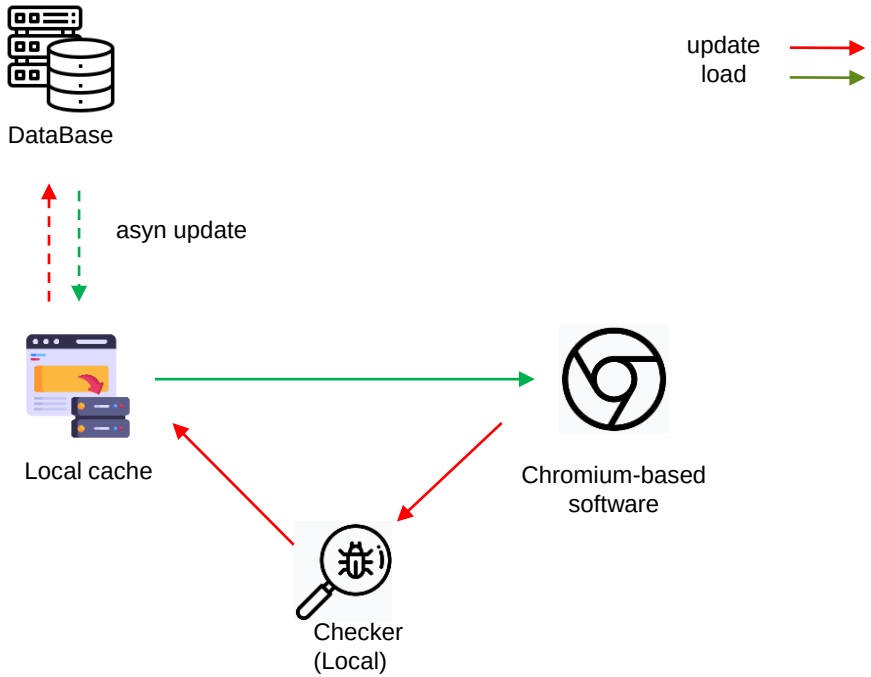
- convenient feature in note/doc software
- No privileged API in iframe
- More dangerous than Web security
- The same logic as loading URL

Scenario 3: Cache data injection

How chromium-based software load and update data?



Data transfer in WEB



Data transfer in some
Chromium-based software

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

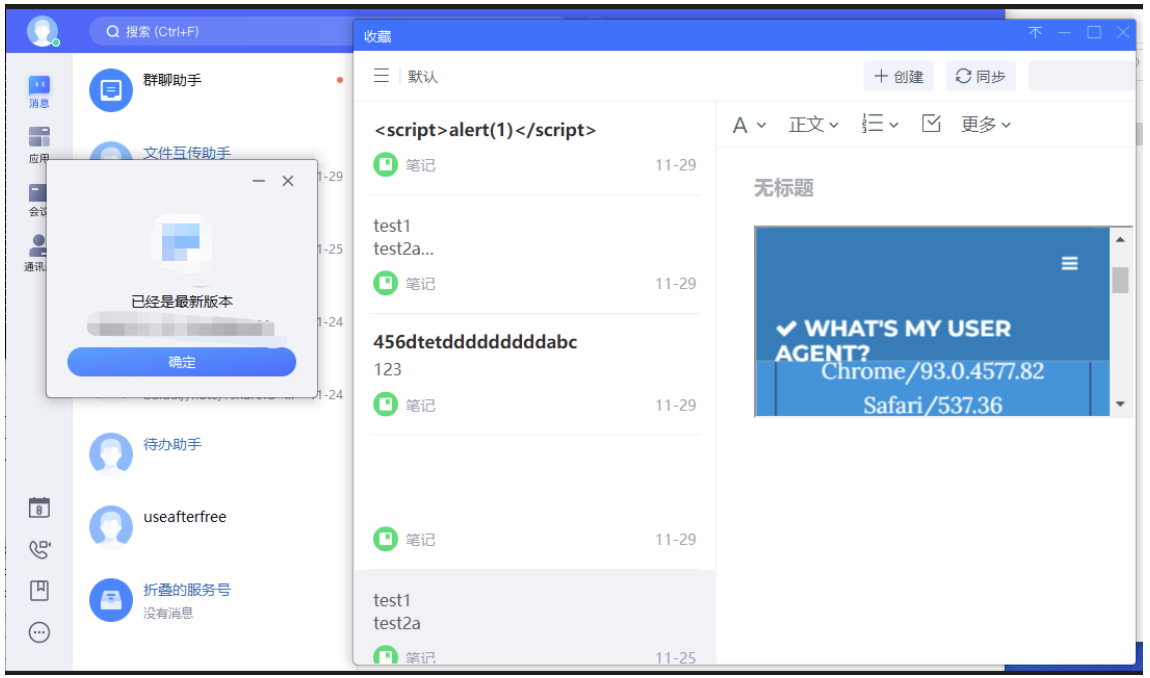
Scenario 3: Cache data injection

Cache data in chromium-base software

- How to find the cache file?

- What are the types of cache file?
  - Database
  - Json
  - custom format
  - ……





腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

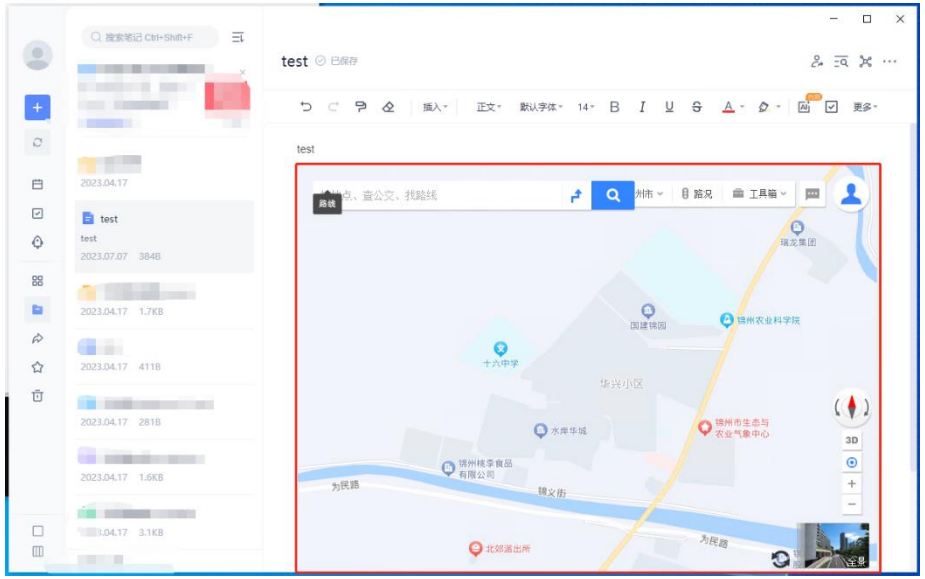
Scenario 3: Cache data injection

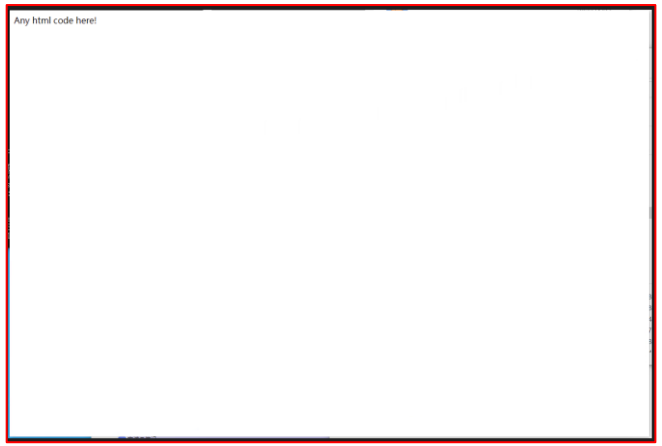
An interesting case

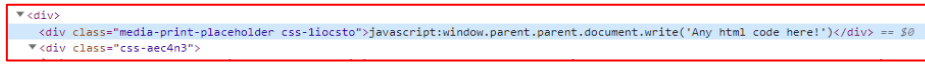
- From Local cache modify to server store XSS



javascript:window.parent.parent.document.write('Any html code here!')







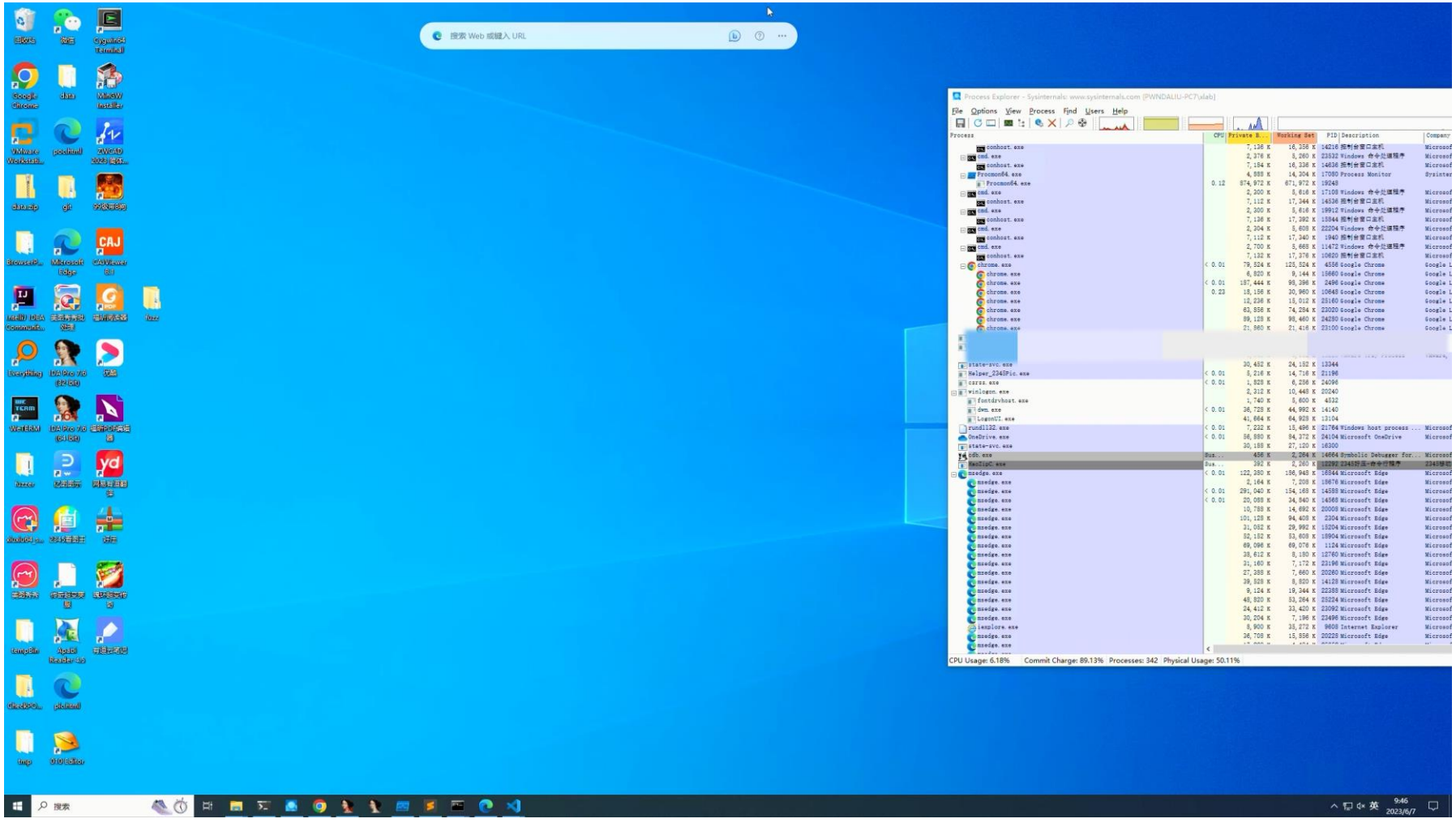


腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 3: Cache data injection

An interesting case

- From Local cache modify to RCE



腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

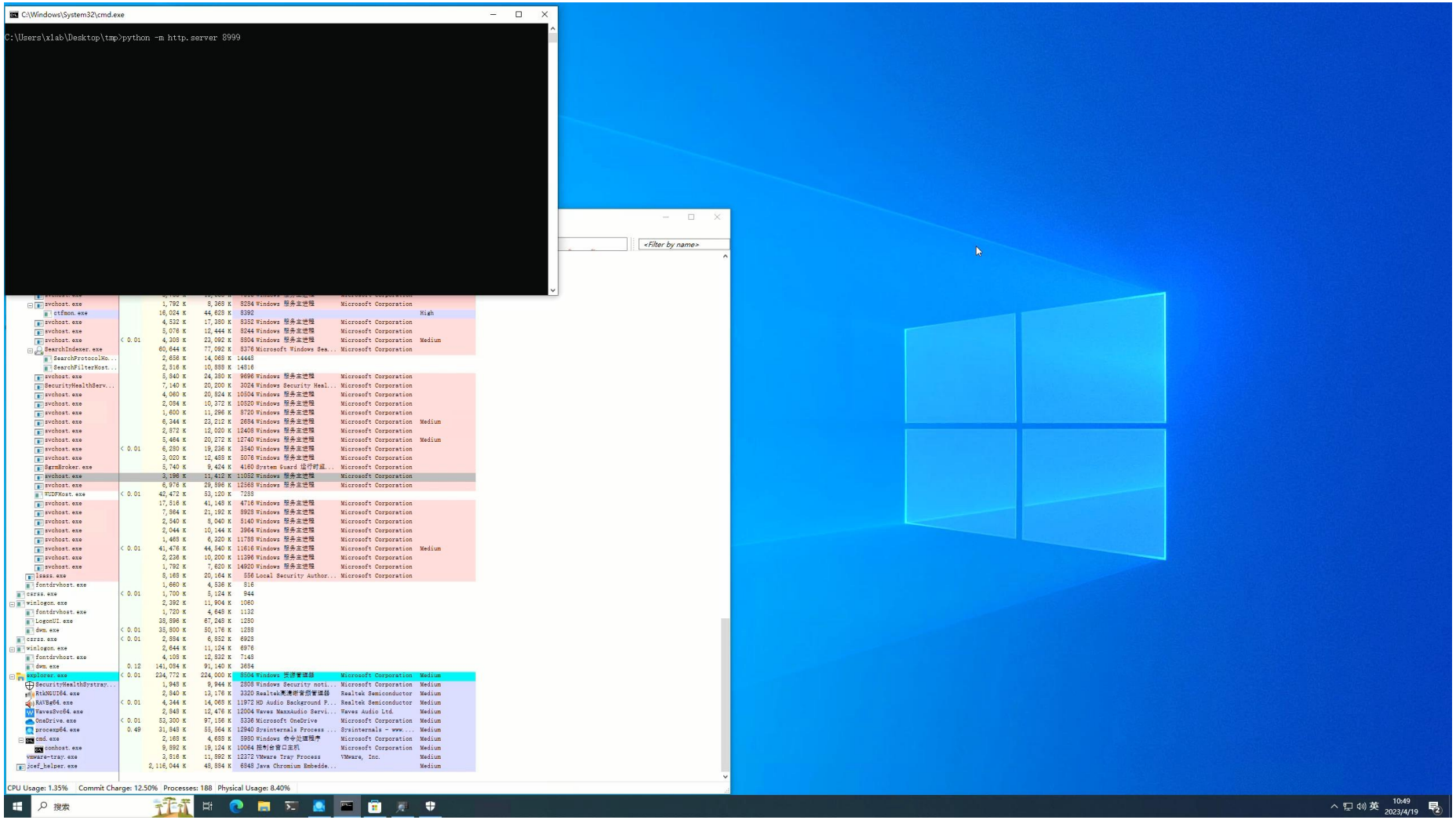
Scenario 4: plugin store and other open platform

Plugin store and other open platform

- Plugins give developers more functionality
- Plugins usually need to be officially reviewed before put on the store
- Has the convenience of a watering hole attack
- Hackers **_may_** use browser components to bypass plugin rereviews

Scenario 4: plugin store and other open platform

Plugin store and other open platform (demo)



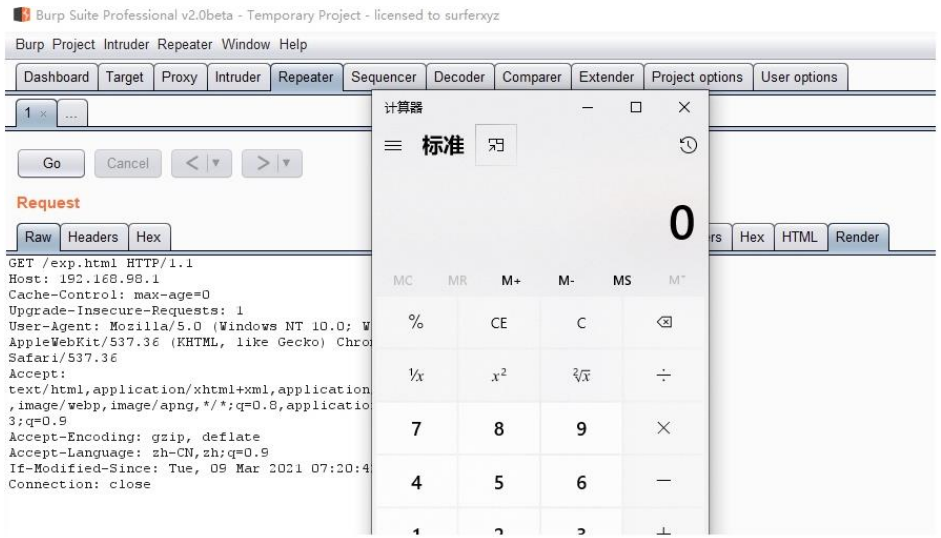
腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Scenario 5: other software feature

other software feature

Some functions may have an arbitrary page loading process, such as pre-rendering of crawlers/notes.

- Burp Suite RCE in 2021
- ……



[11] https://blog.noah.360.net/burp-suite-rce/

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

A general V8 UAF exploit method in 32 bit

As the RCE window period is short, we need to write exp ASAP.

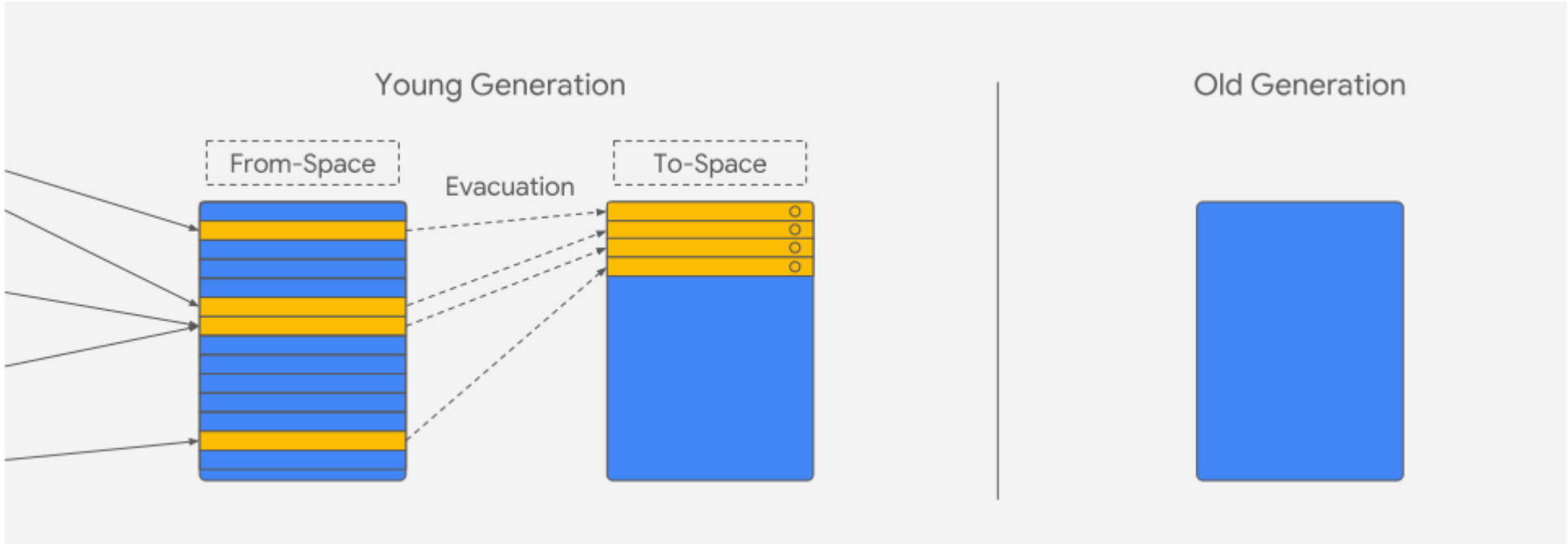
Typer bug => foo the Turbofan to make inconsistent scenes with JIT function, resulting Type Confusion

What about Use-after-Free? 🤔

A general V8 UAF exploit method in 32 bit

CVE-2022-1310(issue-1307610)

Overview about V8 Minor GC : track references from old-space-object to new-space-object by write-bariier



A general V8 UAF exploit method in 32 bit

CVE-2022-1310(issue-1307610)

Root cause : lack of write-barrier in RegExpUtils::SetLastIndex

```
MaybeHandle<Object> RegExpUtils::SetLastIndex(Isolate* isolate,
                                              Handle<JSReceiver> recv,
                                              uint64_t value) {
  Handle<Object> value_as_object =
      isolate->factory()->NewNumberFromInt64(value); /*** A ***/
  if (HasInitialRegExpMap(isolate, *recv)) { /*** B ***/
    JSRegExp::cast(*recv).set_last_index(*value_as_object,
    SKIP_WRITE_BARRIER); /*** C ***/
    return recv;
  } else {
    return Object::SetProperty(
        isolate, recv, isolate->factory()->lastIndex_string(),
        value_as_object,
        StoreOrigin::kMaybeKeyed, Just(kThrowOnError));
  }
}
```

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

A general V8 UAF exploit method in 32 bit

V8 ONE_BYTE_INTERNALIZED_STRING_TYPE
String layout : map | hash | length | content

```
gdb-peda$ telescope 0x5bc922bd-1
0000| 0x5bc922bc --> 0x370c21c9 -->
0004| 0x5bc922c0 --> 0xc40d3bc2
0008| 0x5bc922c4 --> 0x6
0012| 0x5bc922c8 ("cook1e")
0016| 0x5bc922cc --> 0x6531 ('1e')
```

```
DebugPrint: 0x5bc922bd: [String] in OldSpace: #cook1e
0x370c21c9: [Map] in ReadOnlySpace
 - type: ONE_BYTE_INTERNALIZED_STRING_TYPE
 - instance size: variable
 - elements kind: HOLEY_ELEMENTS
 - unused property fields: 0
 - enum length: invalid
 - stable_map
 - back pointer: 0x370c232d <undefined>
 - prototype_validity cell: 0
 - instance descriptors (own) #0: 0x370c2139 <Other heap object
(STRONG_DESCRIPTOR_ARRAY_TYPE)>
 - prototype: 0x370c21ad <null>
 - constructor: 0x370c21ad <null>
 - dependent code: 0x370c2131 <Other heap object
(WEAK_FIXED_ARRAY_TYPE)>
 - construction counter: 0
```

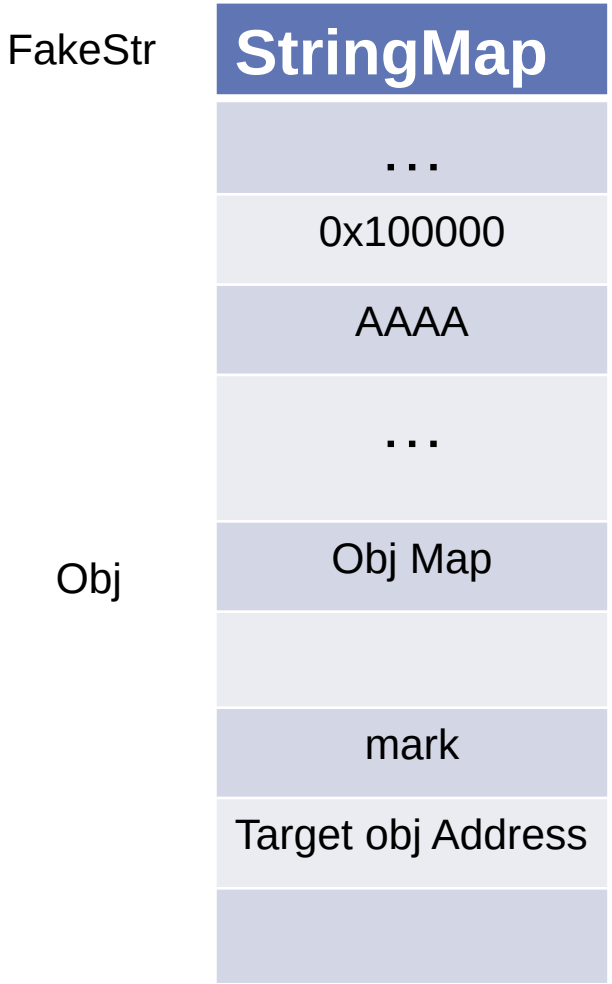
A general V8 UAF exploit method in 32 bit

Overview : fake one_byte_internalized_string

AddressOf:

1. Obj : {mark:0x13371337,prop:target_obj}

2. Fake string : find target object address by mark.😎



腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

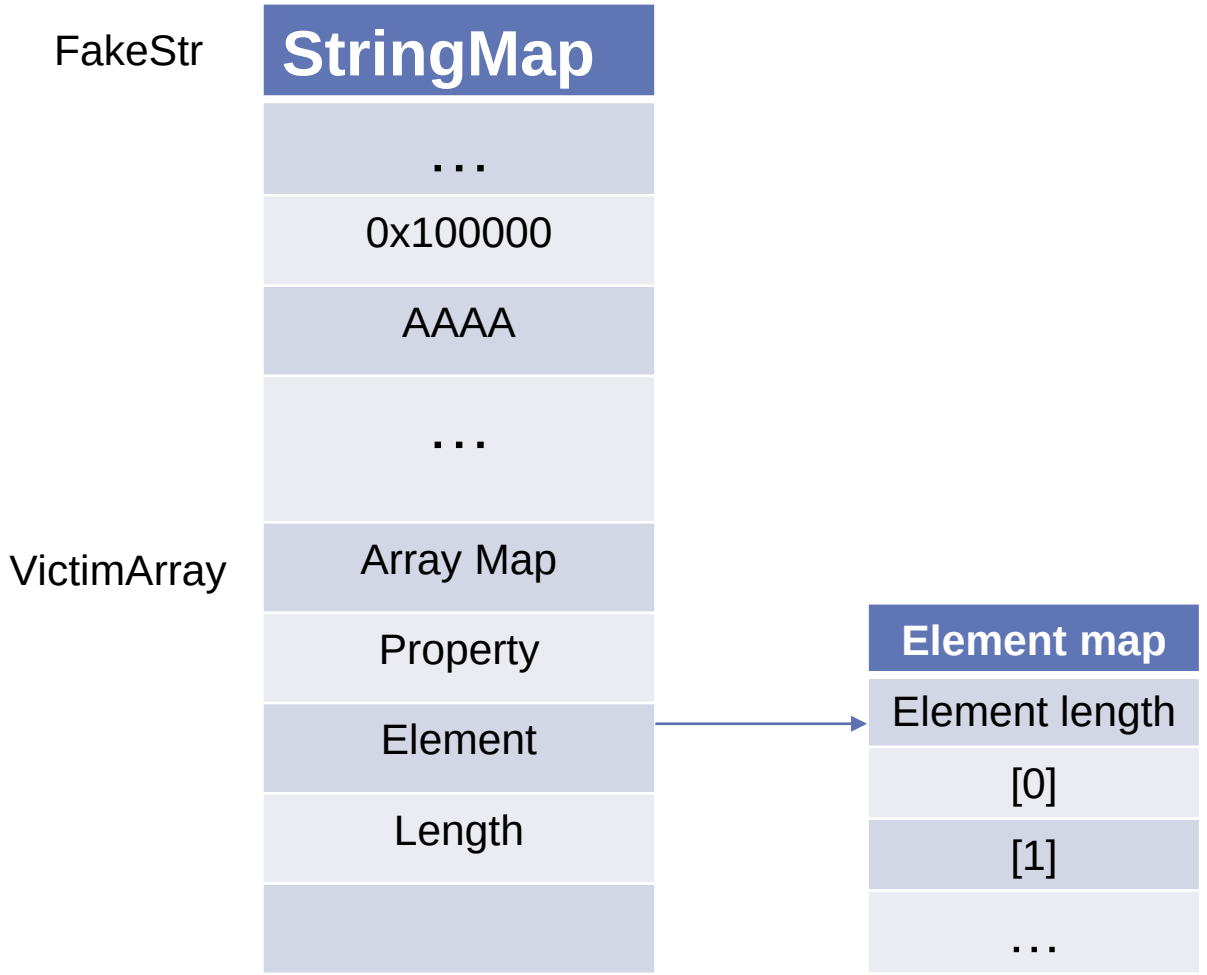
A general V8 UAF exploit method in 32 bit

Overview : fake one_byte_internalized_string

ArbitraryRead/ArbitraryWrite:

1. Hijack VictimArray->Element to target_addr-0x10

2. Read/write VictimArray[0] will directly read/write target_addr



腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

A general V8 UAF exploit method in 32 bit

But how to fake string?
V8 object use map to identify the type, so the key is to fake string map.

HeapSpray:

1. Stable address : Spray Heap with ArrayBuffer and we can get a stable address with debugger

2. Fake string map in this memory : fake string map's prototype、constructor and the map of map.

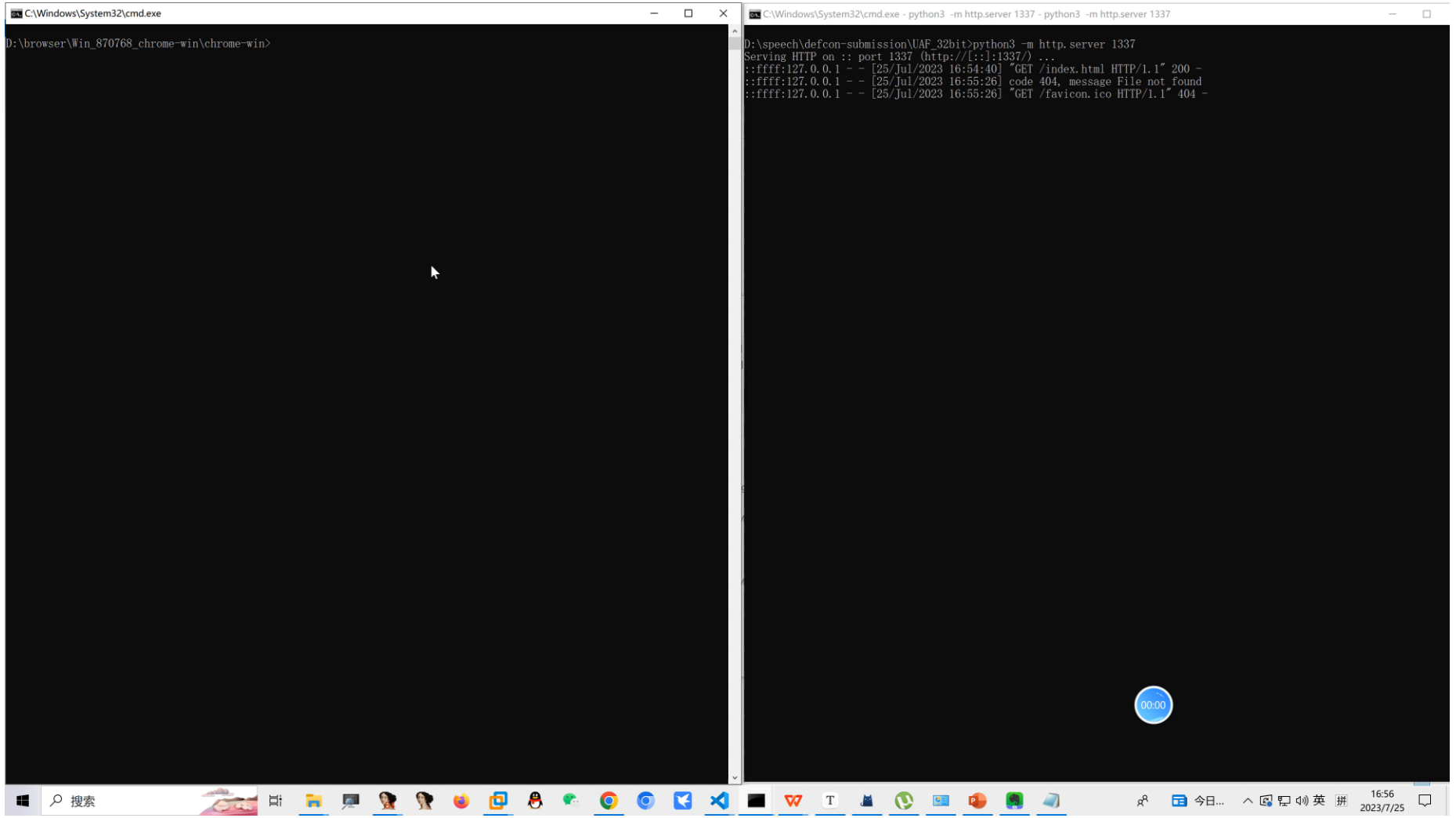
3. Spray FakeStr to occupy the RegExp.lastIndex with JsArray

```
0:026> dds 0x33c824cd-1 L8
33c824cc  20cf4001
33c824d0  20cf4001
33c824d4  00030000
33c824d8  20cf4001
33c824dc  20cf4001
33c824e0  20cf4001
33c824e4  00030000
33c824e8  20cf4001
```

```
0x33c824cd <String[196608]: #...<truncated>>>
```

A general V8 UAF exploit method in 32 bit

RCE on 91.0.4472.0(demo)



腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Summary

- A little bug or feature can lead to RCE.
- Lack of attention to the attack path of browser vulnerabilities
- The attack surface is relatively rudimentary
- Hackers also can bypass mitigation measures directly using powerful Chromium bugs.



3. Achieve RCE Without Sandbox Escape Bug

The sandbox in chromium

- Mandatory access controlled environment
- Isolated Process when HTML rendering and JavaScript execution
- Limited resource access & Limited IPC/kernel interaction access

OS	Sandbox implementation
Windows	A restricted token& The Windows job object & The Windows desktop object& Integrity levels&
Linux	Seccomp-BPF & User namespaces
Android	SELinux



How can we achieve RCE in sandbox-enabled software without a chromium sandbox-escape bug?

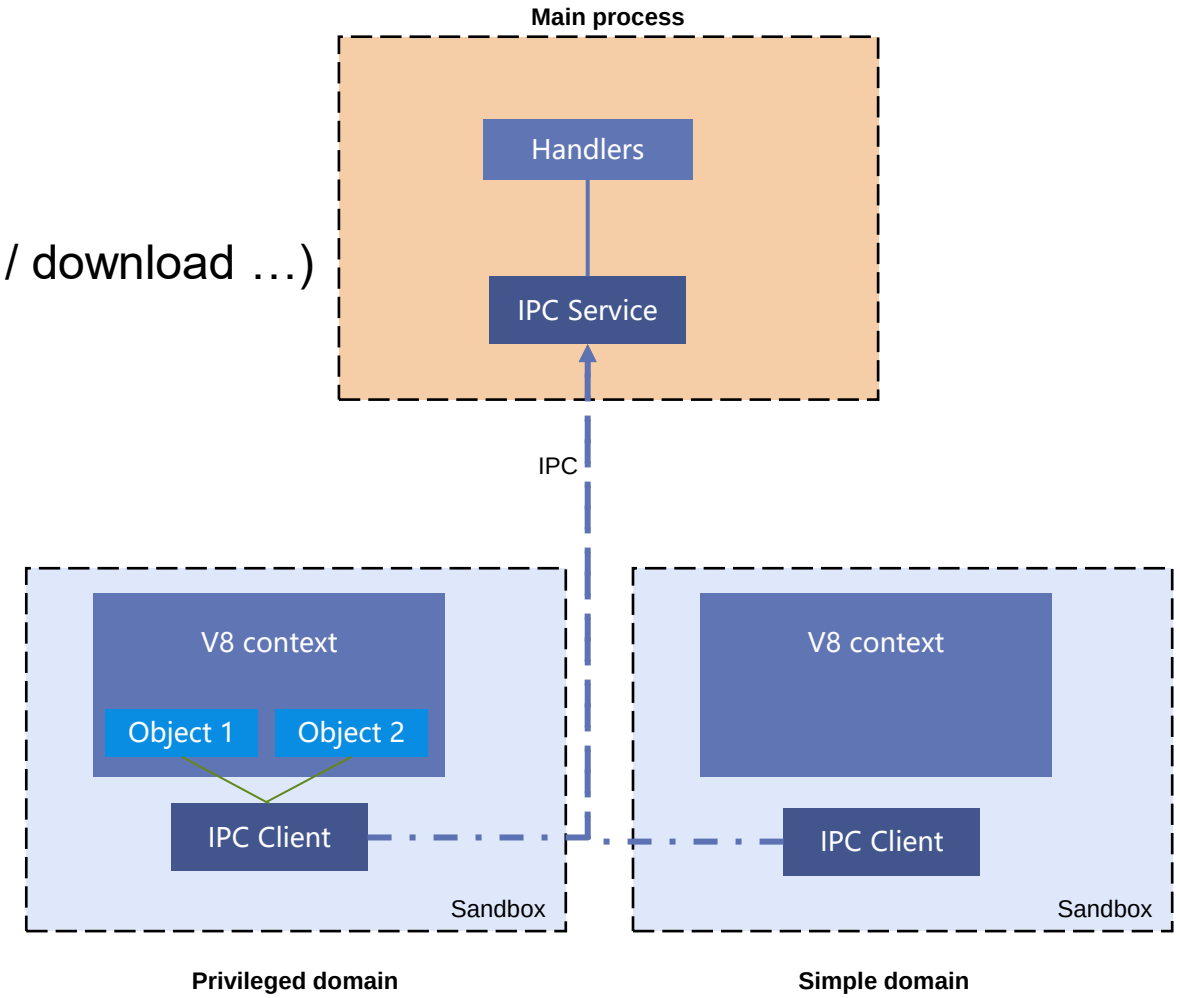
[12] https://blackhat.com/asia-19/briefings/schedule/index.html

What's new in chromium-based software?

Privileged API

- client-side behaviors in specific scenarios(upload / download …)
- For renderer, it is a series of JavaScript objects
- For Browser, it is a series of handler code

Type	Privileged API implementation
Electron	preload.js
CEF	CefV8Handler
Android	JavaScriptInterface

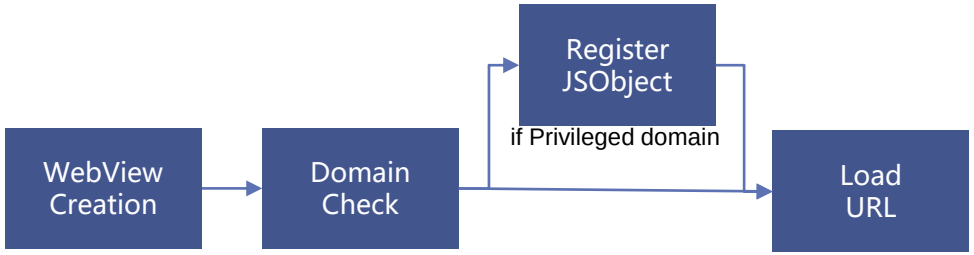


腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

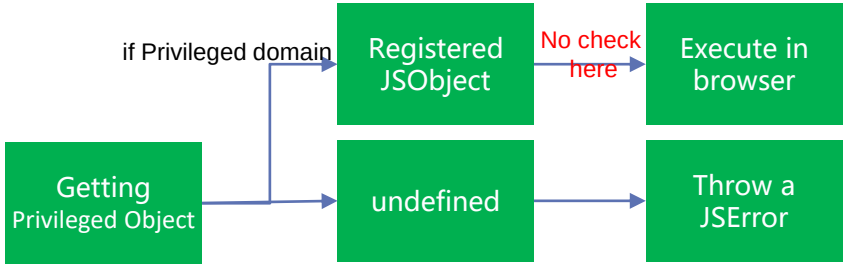
What's new in chromium-based software?

The development logic

1.



Initialization



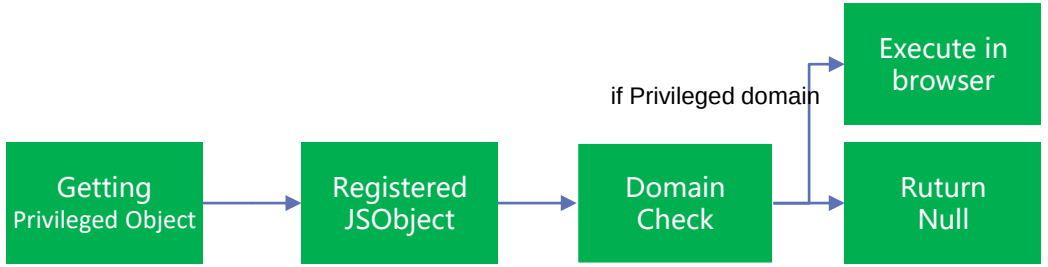
Runtime

What if a non-privileged WebView get a privileged context?

2.



Initialization



Runtime

How to properly determine if IPC access is from a privileged domain?

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Sandbox Escape In Electron

Sandbox Escape with the lack of context isolation

context isolation : separate JS context between page's scripts and preload scripts， introduced in electron 1.4.15 and enabled by default since Electron 12

Preload.js : has access to:

(1) Some Node.js native classes like **require**

(2) Some Electron modules like **ipcRenderer**

Attacker can use Prototype Pollution to make any ipc call.

[13] : https://speakerdeck.com/masatokinugawa/electron-abusing-the-lack-of-context-isolation-curecon-en?slide=2
[14] : https://i.blackhat.com/asia-19/Thu-March-28/bh-asia-Carettoni-Preloading-Insecurity-In-Your-Electron.pdf

Sandbox Escape In Electron

CVE-2022-29247

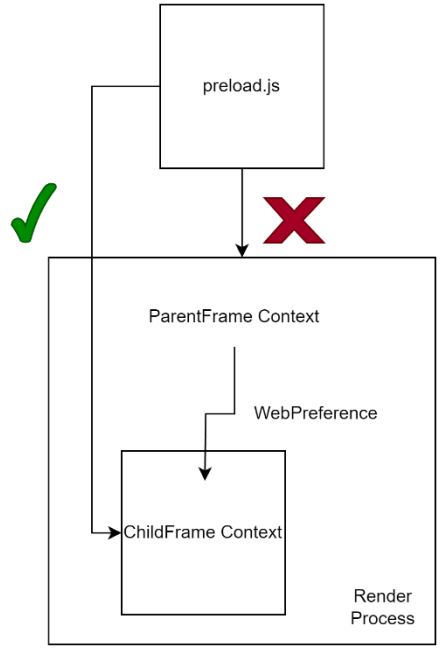
Overview : Disabling context isolation, node_integration_in_sub_frames using an unauthorised frame
In a comprised render process, attackers overwrite the context's prefs.node_integration_in_sub_frames to load preload.js in Child Frame

```
diff --git a/third_party/blink/common/web_preferences/web_preferenc
index 1e51bb68df930c44e165b32c8c03b99986792209..9535330e3e7a3c4621e
--- a/third_party/blink/common/web_preferences/web_preferences.cc
+++ b/third_party/blink/common/web_preferences/web_preferences.cc
@@ -143,6 +143,29 @@ WebPreferences::WebPreferences()
       navigate_on_drag_drop(true),
       v8_cache_options(blink::mojom::V8CacheOptions::kDefault),
       record_whole_document(false),
+
+      // Begin Electron-specific WebPreferences.
+      disable_electron_site_instance_overrides(),
+      background_color(base::EmptyString()),
+      opener_id(0),
+      context_isolation(false),
+      enable_remote_module(false),
+      world_safe_execute_javascript(false),
+      guest_instance_id(0),
+      hidden_page(false),
+      offscreen(false),
+      preload(base::FilePath::StringType()),
+      native_window_open(false),
+      node_integration(false),
+      node_integration_in_worker(false),
+      node_leakage_in_renderers(false),
+      node_integration_in_sub_frames(false),
+      enable_spellcheck(false),
+      enable_plugins(false),
+      enable_websql(false),
+      webview_tag(false),
+      // End Electron-specific WebPreferences.
+
```

```
void ElectronSandboxedRendererClient::DidCreateScriptContext(
    v8::Handle<v8::Context> context,
    content::RenderFrame* render_frame) {
  // Only allow preload for the main frame or
  // For devtools we still want to run the preload_bundle script
  // Or when nodeSupport is explicitly enabled in sub frames
  bool is_main_frame = render_frame->IsMainFrame();
  bool is_devtools =
      IsDevTools(render_frame) || IsDevToolsExtension(render_frame);

  bool allow_node_in_sub_frames =
      render_frame->GetBlinkPreferences().node_integration_in_sub_frames;

  bool should_load_preload =
      (is_main_frame || is_devtools || allow_node_in_sub_frames) &&
      !IsWebViewFrame(context, render_frame);
  if (!should_load_preload)
    return;
```



[15] https://hackerone.com/s1r1u5

Sandbox Escape In Electron

CVE-2022-29247

Overview : Disabling context isolation, node_integration_in_sub_frames using an unauthorised frame
In a comprised render, it overwrite the context's prefs.context_isolation and disable CISO in new renderer process.
Then we can send unauthorized IPC call with prototype pollution.

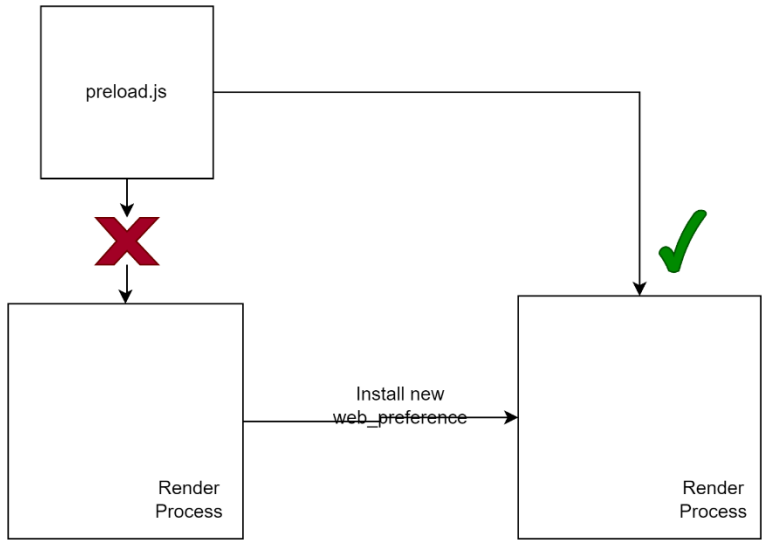
```
void ElectronRenderFrameObserver::DidInstallConditionalFeatures(
    v8::Handle<v8::Context> context,
    int world_id) {
  auto* isolate = context->GetIsolate();
  v8::MicrotasksScope microtasks_scope(
      isolate, v8::MicrotasksScope::kDoNotRunMicrotasks);

  if (ShouldNotifyClient(world_id))
    renderer_client_->DidCreateScriptContext(context, render_frame_);

  auto prefs = render_frame_->GetBlinkPreferences();
  [0] bool use_context_isolation = prefs.context_isolation;
  // This logic matches the EXPLAINED logic in electron_renderer_client.cc
  // to avoid explaining it twice go check that implementation in
  // DidCreateScriptContext();
  bool is_main_world = IsMainWorld(world_id);
  bool is_main_frame = render_frame_->IsMainFrame();
  bool allow_node_in_sub_frames = prefs.node_integration_in_sub_frames;

  bool should_create_isolated_context =
      use_context_isolation && is_main_world &&
      (is_main_frame || allow_node_in_sub_frames);

  if (should_create_isolated_context) {
    [1] CreateIsolatedWorldContext();
    if (!renderer_client_->IsWebViewFrame(context, render_frame_))
      renderer_client_->SetupMainWorldOverrides(context, render_frame_);
  }
}
```

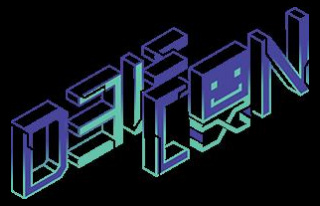


[15] https://hackerone.com/s1r1u5

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Others





4. Security Recommendations

Security Recommendations

How to defend against “RCE window period”

- For programs that use Chromium components, it is recommended to promptly incorporate the latest version of the Chromium library.
- For software with high coupling, it may not be possible to update to a major version of Chromium. In this case, it is recommended to use a stable development version and introduce security patches with software updates.
- Enable the Chromium sandbox.
- Disable unused software functionalities or enable internal integrated security defense mechanisms.

Security Recommendations

Security Development Suggestions

- Add backend validation and frontend rendering validation.
- Follow the principle of least privilege, strictly control the list of privileged domains, and limit privileged API functionalities.
- Enable universal protection mechanisms, such as Content Security Policy (CSP).
- Be vigilant about input from unexpected sources.



5. Conclusion & Takeaways

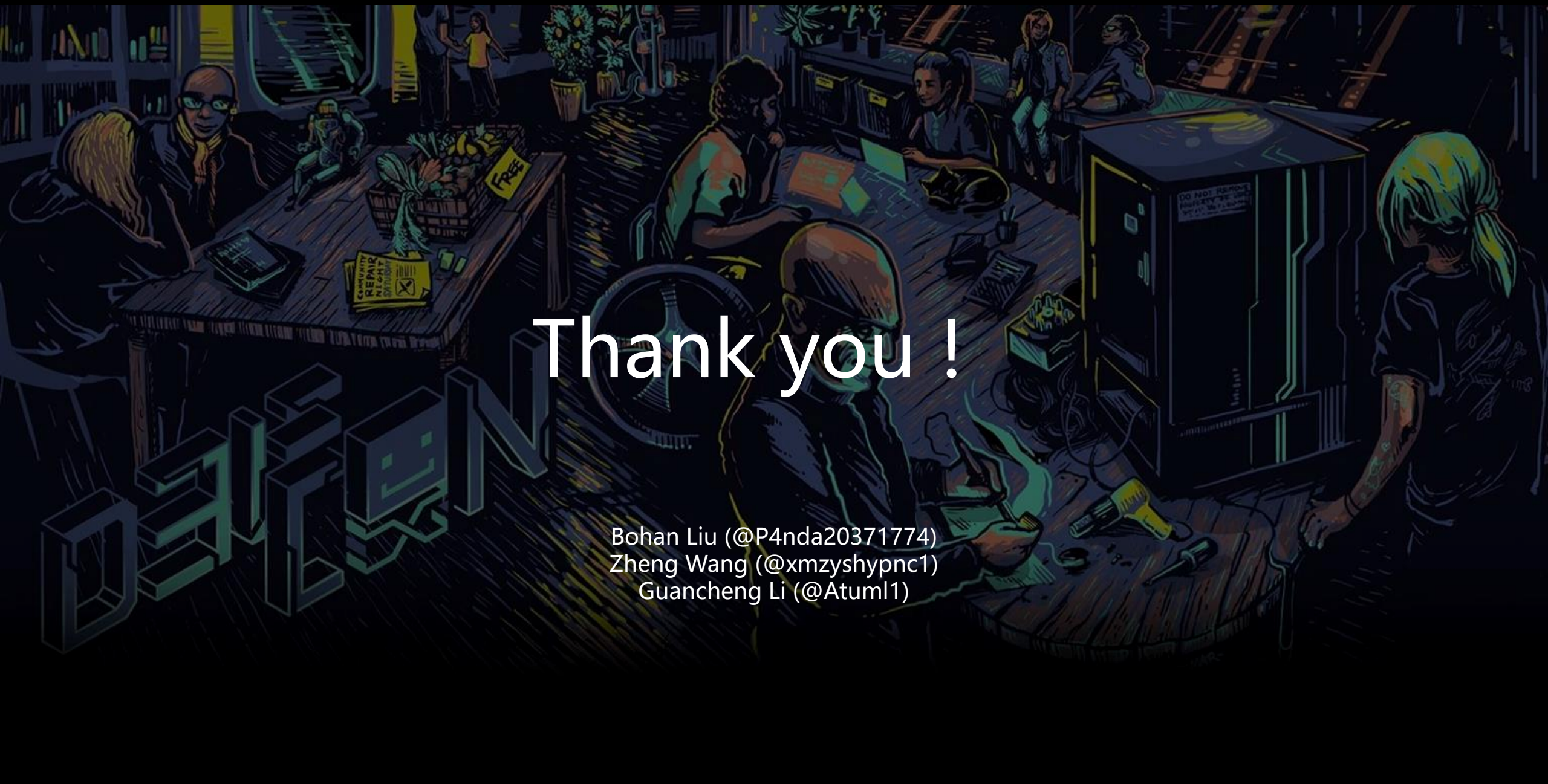
Conclusion & Takeaways

In this topic:

- Verify that a large number of chromium-based software have RCE window period.
- Summarize some software attack surfaces that cooperate with the RCE window period, and give real-world examples.
- Share a general V8 UAF exploit method in 32 bit.
- Provide some security recommendations to help strengthen the security of chromium-based software.

Acknowledgement

- Yang Yu (@tombkeeper)
- Kai Song (@ExpSky)
- Wei Liu
- Guannan Wang (@Keenan7310)
- ZeKai Wu (@hellowuzekai)
- Junyu Zhou(@md5_salt)
- Haibin Shi (@Aryb1n)
- Yongke Wang (@Rudykewang)
- Huiming Liu (@liuhm09)



Thank you !

Bohan Liu (@P4nda20371774)
Zheng Wang (@xmzyshypnc1)
Guancheng Li (@Atuml1)

References

[1] https://www.cvedetails.com/product/15031/Google-Chrome.html?vendor_id=1224
[2] https://googleprojectzero.blogspot.com/p/0day.html
[3] https://bughunters.google.com/about/rules/5745167867576320/chrome-vulnerability-reward-program-rules
[4] https://blog.electrovolt.io/posts/discord-rce/
[5] https://github.com/r4j0x00/exploits/commit/7ba55e5ab034d05877498e83f144e187d3ddb160
[6] https://i.blackhat.com/USA-22/Thursd...sktop-Apps.pdf
[7] https://github.com/pd4d10/debugtron
[8] https://github.com/electron/asar
[9] https://github.com/typora/typora-issues/issues/2959
[10] https://bugs.chromium.org/p/chromium/issues/detail?id=722756
[11] https://blog.noah.360.net/burp-suite-rce/
[12] https://blackhat.com/asia-19/briefings/schedule/index.html
[13] https://speakerdeck.com/masatokinugawa/electron-abusing-the-lack-of-context-isolation-curecon-en?slide=2
[14] https://i.blackhat.com/asia-19/Thu-March-28/bh-asia-Carettoni-Preloading-Insecurity-In-Your-Electron.pdf
[15] https://hackerone.com/s1r1u5
[16] https://crbug.com/1307610