



# Real World 0-days

0-days Detected In-the-Wild in 2021



Maddie Stone  
@maddiestone  
OffensiveCon 2022



Make Oday hard.

0-day exploits  
exploited  
in-the-wild.

0-day exploits  
detected &  
disclosed as  
in-the-wild.

57

0-days detected in-the-wild

## 0day "In the Wild"

Last updated: 2021-05-03

This spreadsheet is used to track cases of zero-day exploits that were detected "in the wild". This means the vulnerability was detected in real attacks against users as a zero-day vulnerability (i.e. not known to the public or the vendor at the time of detection). This data is collected from a range of public sources. We include relevant links to third-party analysis and attribution, but we do this only for your information; their inclusion does not mean we endorse or validate the content there.

An introduction to this spreadsheet is available on the Project Zero blog:

<https://googleprojectzero.blogspot.com/p/0day.html>

Some additional notes on how the data is processed:

- **Scope for inclusion:** there are some 0day exploits (such as CVE-2017-12824) in areas that aren't active research targets for Project Zero. Generally this list includes targets that Project Zero has previously investigated (i.e. there are bug reports in our issue tracker) or will investigate in the near future.
- **Security supported:** this list does not include exploits for software that is explicitly EOL at the time of discovery (such as the ExplodingCan exploit for IIS on Windows Server 2003, surfaced in 2017).
- **Post-disclosure:** this list does not include CVEs that were opportunistically exploited by attackers in the gap between public disclosure (or "full disclosure") and a patch becoming available to users (such as CVE-2015-0072, CVE-2018-8414 or CVE-2018-8440).
- **Reasonable inference:** this list includes exploits that were not discovered in an active breach, but were leaked or discovered in a form that suggests with high confidence that they were probably used "in the wild" at some point (e.g. Equation Group and Hacking Team leaks).
- **Date resolution:** we only set the date of discovery when the reporter specifies one. If a discovery is indicated as being made in "late April" or "early March", we record that as if no date was provided.
- **Attribution:** generally the "claimed attribution" column refers to the attack team that is reportedly using the exploit, but in some cases it can refer to the supplier of the exploit (c.f. HackingTeam, NSO Group, Exodus Intel) if no other information is available.
- **Time range:** data collection starts from the day we announced Project Zero -- July 15, 2014.

For additions, corrections, questions, or comments, please contact [0day-in-the-wild@google.com](mailto:0day-in-the-wild@google.com)

## Root Cause Analyses

Originally published by Maddie Stone on the [Google Project Zero blog](#) on 27 July 2020

Beginning in 2019, Project Zero began a program to systematically study 0-day exploits that are used in the wild. It's another way we're trying to make 0-day hard. We published our [tracking spreadsheet](#) for recording publicly known cases of detected 0-day exploits. Today we're beginning to share the root cause analyses we perform on these detected 0-day exploits. To better understand our approach and reasoning behind these analyses, please read [this blog post](#).

We will continue to publish new root cause analyses as they are completed, hopefully in a very timely manner. We hope other researchers who detect and/or analyze 0-day exploits will also publish this information to better inform actions and decision making in the security and tech communities. The template that we use is available [here](#). We welcome pull requests!

Our goal is that this information helps the security and technical communities. Please [reach out](#) with any feedback or suggestions.

CVE	Link
CVE-2019-11707: IonMonkey Type Confusion in Array.Pop	<a href="#">↗</a>
CVE-2019-1367: Internet Explorer JScript use-after-free	<a href="#">↗</a>
CVE-2019-13720: Chrome use-after-free in webaudio	<a href="#">↗</a>
CVE-2019-1458: Windows win32k uninitialized variable in task switching	<a href="#">↗</a>
CVE-2019-2215: Android use-after-free in Binder	<a href="#">↗</a>
CVE-2019-7286: iOS use-after-free in cfprefsd	<a href="#">↗</a>
CVE-2019-7287: iOS Buffer Overflow in ProvInfoLOKitUserClient	<a href="#">↗</a>

Caveat: These are my takes & thoughts. I'd love to hear yours.



The uptick in 0-days is due to progress in security.

The uptick in 0-days is due to progress in security.

The data points show that all the 0-days actually look a lot like what we've seen before.

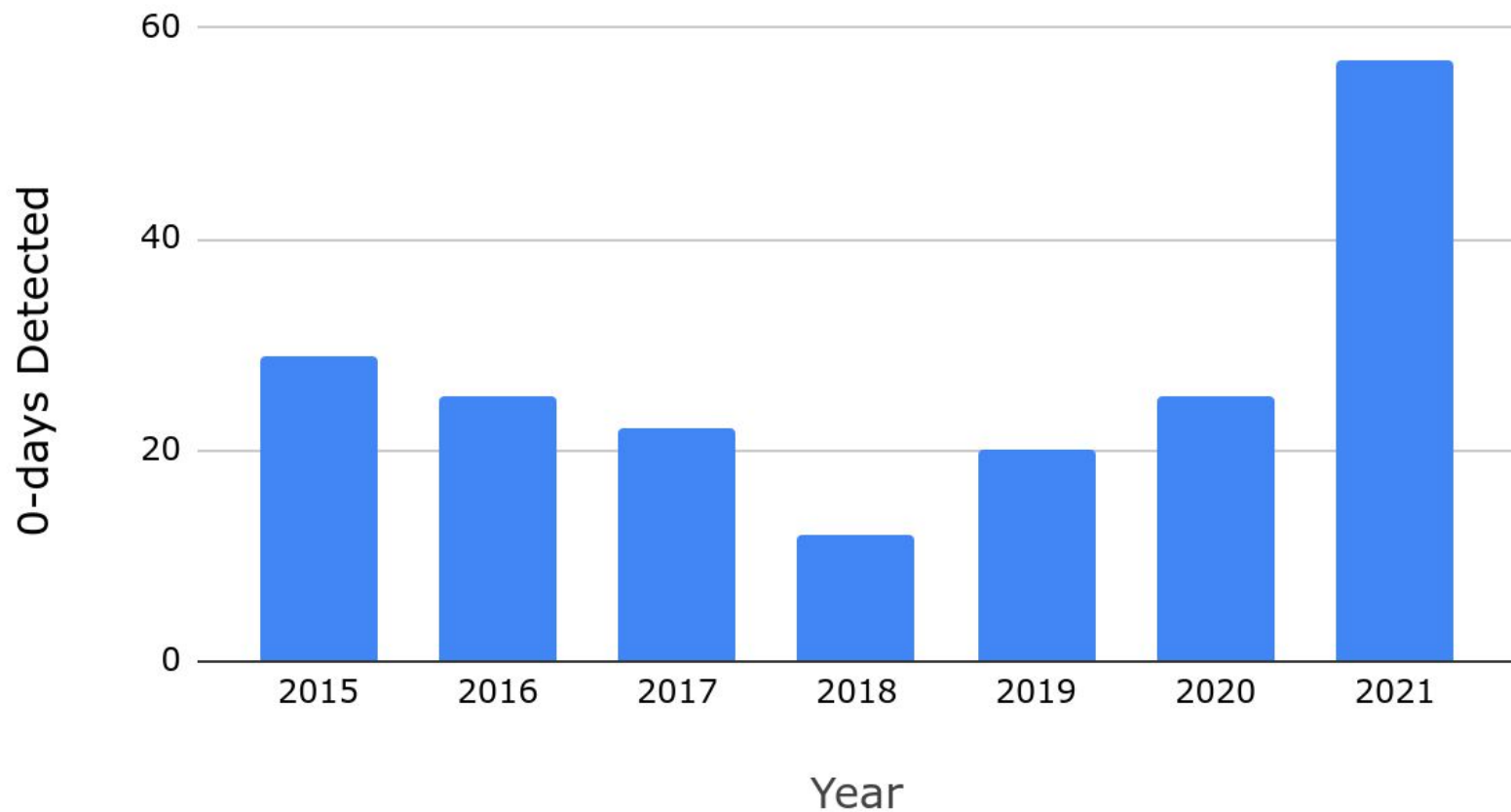
The uptick in 0-days is due to progress in security.

The data points show that all the 0-days actually look a lot like what we've seen before.

While we have more data than ever before, we're still missing a lot.

The uptick in 0-days is due to progress in security.

## In-the-Wild 0-days Detected vs. Year



# Project Zero

News and updates from the Project Zero team at Google

Wednesday, July 29, 2020

## Detection Deficit: A Year in Review of 0-days Used In-The-Wild in 2019

Posted by Maddie Stone, Project Zero

In May 2019, Project Zero released our [tracking spreadsheet](#) for 0-days used “in the wild” and we started a more focused effort on analyzing and learning from these exploits. This is another way Project Zero is trying to make zero-day hard. This blog post synthesizes many of our efforts and what we’ve seen over the last year. We provide a review of what we can learn from 0-day exploits detected as used in the wild in 2019. In conjunction with this blog post, we are also publishing another [blog post](#) today about our root cause analysis work that informed the conclusions in this Year in Review. We are also releasing [8 root cause analyses](#) that we have done for in-the-wild 0-days from 2019.

When I had the idea for this “Year in Review” blog post, I immediately started brainstorming the different ways we could slice the data and the different conclusions it may show. I thought that maybe there’d be interesting conclusions around why use-after-free is one of the most exploited bug classes or how a given exploitation method was used in Y% of 0-days or... but despite my attempts to find these interesting technical conclusions, over and over I kept coming back to the problem of the detection of 0-days. Through the variety of areas I explored, the data and analysis continued to highlight a single conclusion: **As a community, our ability to detect 0-days being used in the wild is severely lacking to the point that we can’t draw significant conclusions due to the lack of (and biases in) the data we have collected.**

# Project Zero

News and updates from the Project Zero team at Google

Wednesday, July 29, 2020

## Detection Deficit: A Year in Review of 0-days Used In-The-Wild in 2019

Posted by Maddie Stone, Project Zero

In May 2019, Project Zero released our [tracking spreadsheet](#) for 0-days used “in the wild” and we started a way Project Zero is trying we’ve seen over the last used in the wild in 2019. In about our root cause analysis [8 root cause analyses](#) that

As a community, our ability to detect 0-days being used in the wild is severely lacking to the point that we can’t draw significant conclusions due to the lack of (and biases in) the data we have collected.

When I had the idea for this “Year in Review” blog post, I immediately started brainstorming the different ways we could slice the data and the different conclusions it may show. I thought that maybe there’d be interesting conclusions around why use-after-free is one of the most exploited bug classes or how a given exploitation method was used in Y% of 0-days or... but despite my attempts to find these interesting technical conclusions, over and over I kept coming back to the problem of the detection of 0-days. Through the variety of areas I explored, the data and analysis continued to highlight a single conclusion: **As a community, our ability to detect 0-days being used in the wild is severely lacking to the point that we can’t draw significant conclusions due to the lack of (and biases in) the data we have collected.**



More detection  
More disclosure





More detection

More disclosure

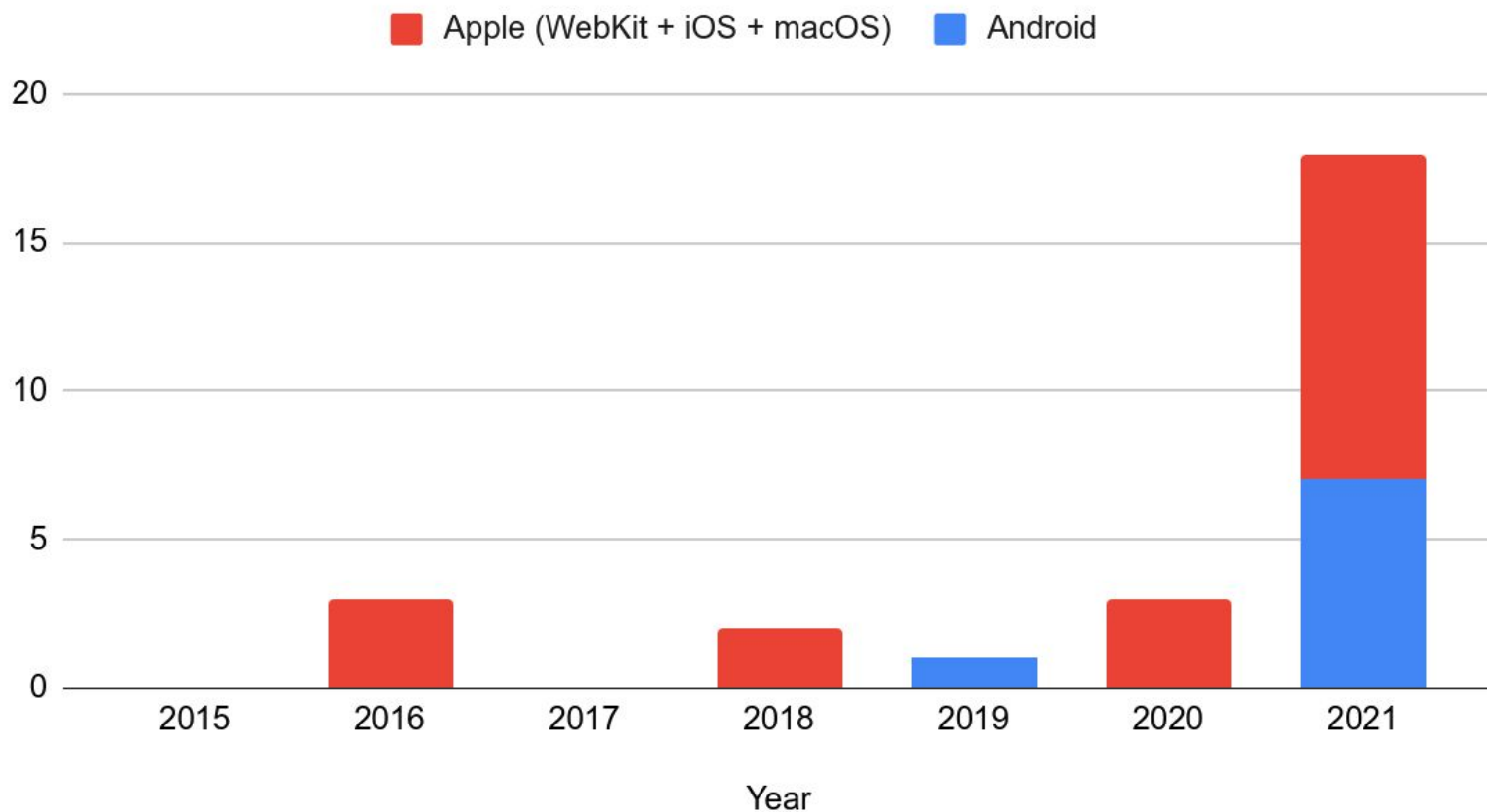
2019 Reporters	2020 Reporters	2021 Reporters
Alibaba Cloud Intelligence Security Team	Andy	cPanel Security Team
Coinbase Security	Codesafe Team of Legendsec at Qi'anxin Group	DBAppSecurity Co., Ltd
ESET	Francisco Alonso & Javier Marcos	DEVCORE Research Team
Google Project Zero	Google Project Zero	Dubex
Google Threat Analysis Group	Google Threat Analysis Group	Enki
Kaspersky Lab	Kaspersky Lab	EXPMON
Microsoft Security Response Center	National Security Agency	Github Security Lab
Resecurity, Inc.	Qihoo 360 ATA	Google Project Zero
Trend Micro	Trend Micro Research	Google Threat Analysis Group
	Trend Micro's Zero Day Initiative	Kaspersky Lab
		Mandiant
		Mattias Buelens
		Microsoft Browser Vulnerability Research
		Microsoft Security Response Center
		Microsoft Threat Intelligence Center
		Qihoo 360 ATA
		Sangfor
		The Citizen Lab
		Tianfu
		Volexity



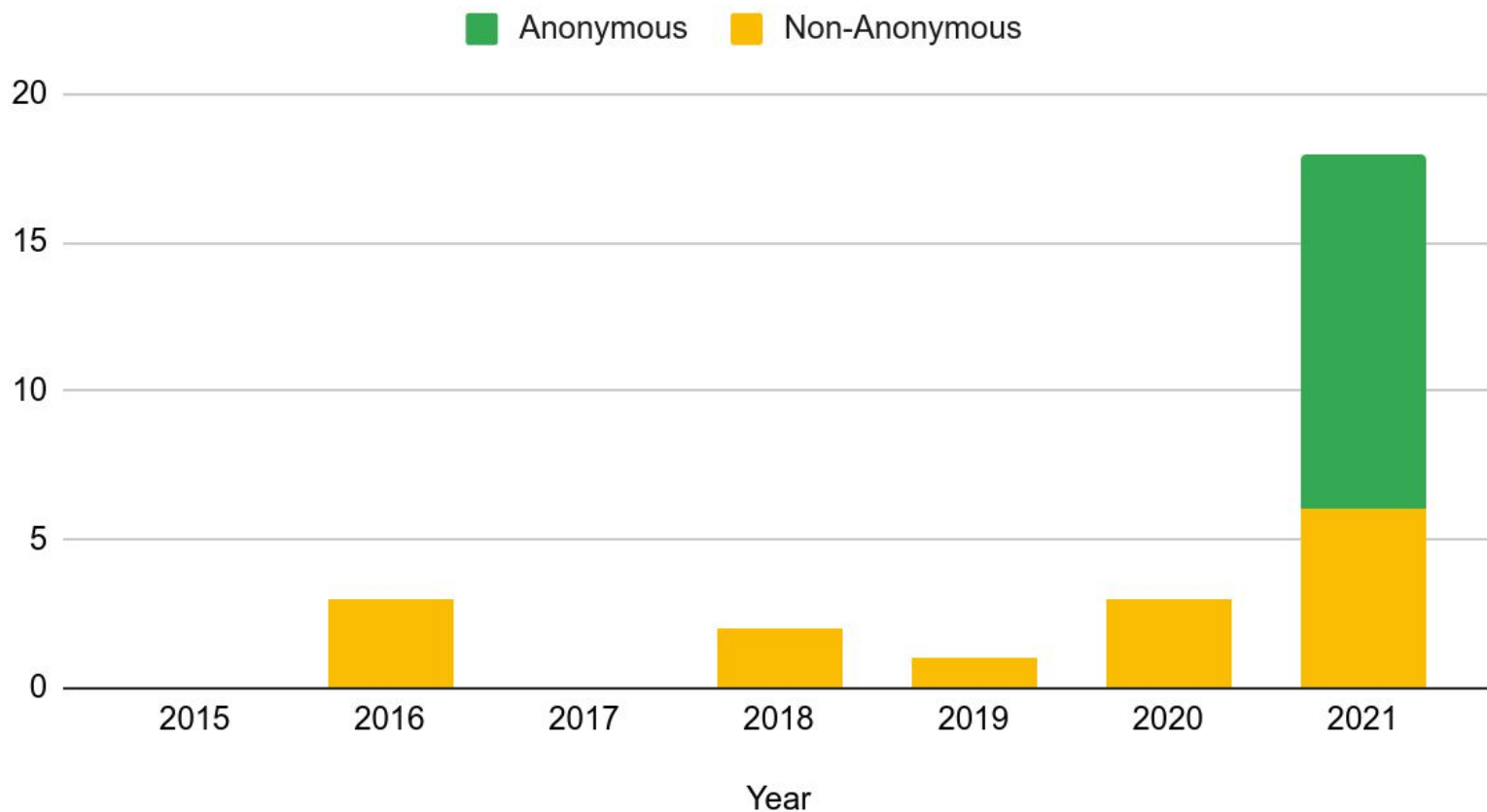
More detection

More disclosure

## Android and Apple (WebKit + iOS + macOS)



## Android and Apple (WebKit + iOS + macOS)



Nothing is all that new.\*

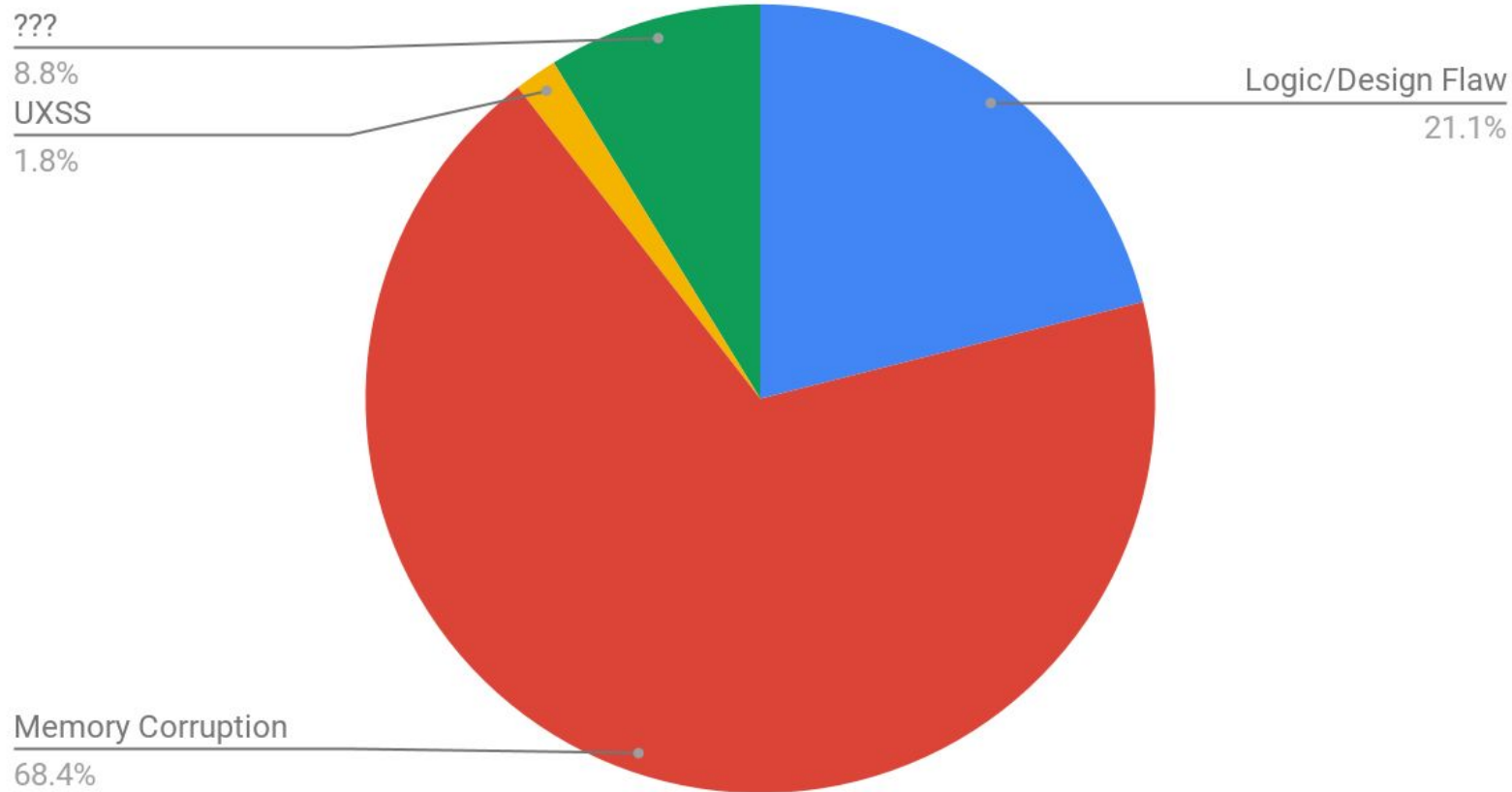
\* one exception

Memory Corruption is **still** the name  
of the game.

# 39 Memory Corruption Bugs



## Count of Type



17 use-after-free

6 out of bounds r/w

4 buffer overflow

4 integer overflow

# Browsers

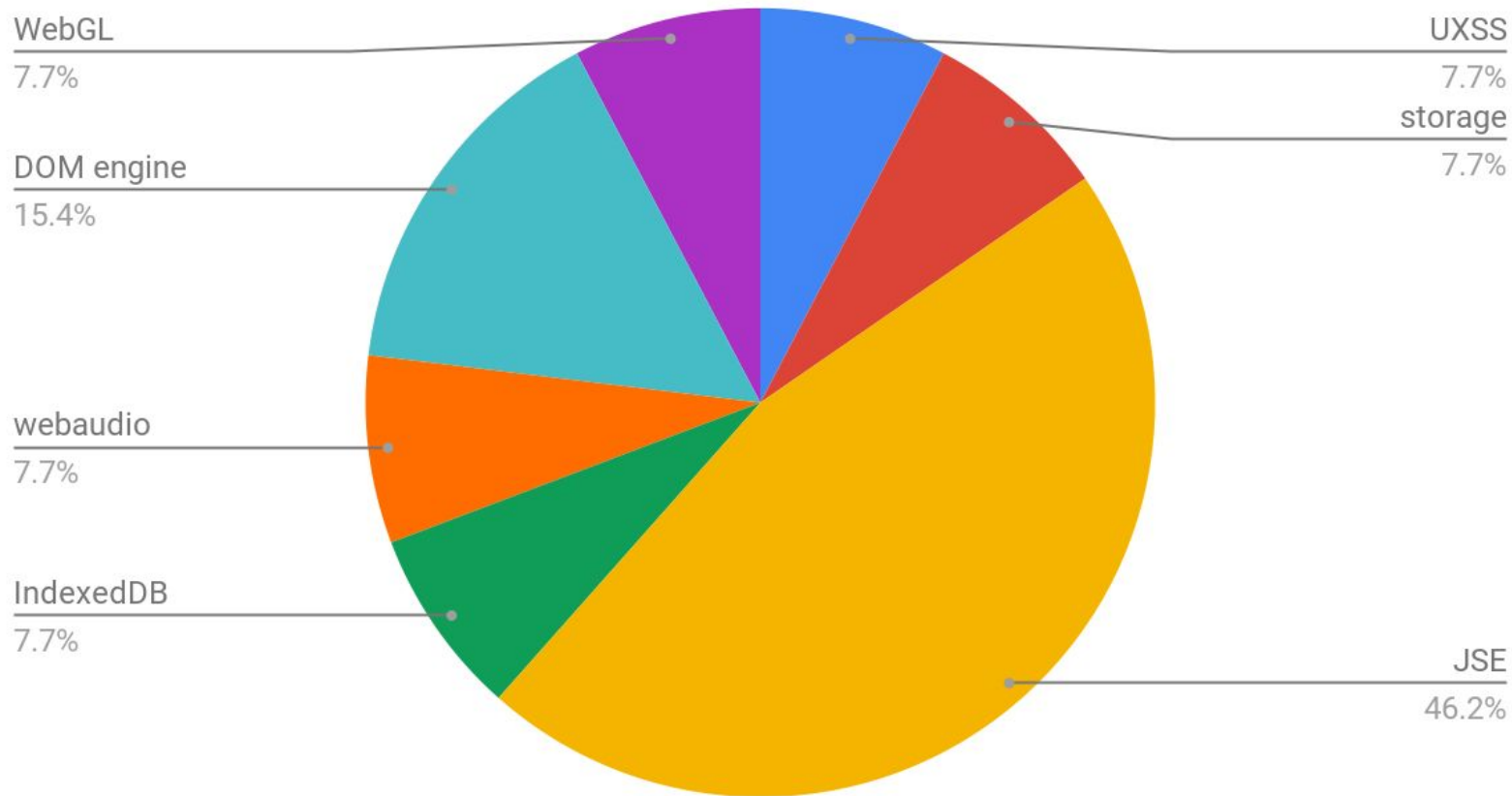
14 Chromium

7 WebKit (Safari)

4 Internet Explorer

0-days detected in-the-wild

# Component Targeted



# Chromium

6 Javascript Engine (v8)

2 DOM Engine (Blink)

1 WebGL

1 IndexedDB

1 webaudio

1 Portals

2 Other

WebKit



4 Javascript Engine (JSC)

1 IndexedDB

1 Storage

1 UXSS

# Use-after-free in WebKit

## CVE-2021-30858

# CVE-2021-30858

## WebKit

Available for: iPhone 6s and later, iPad Pro (all models), iPad Air 2 and later, iPad 5th generation and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: Processing maliciously crafted web content may lead to arbitrary code execution. Apple is aware of a report that this issue may have been actively exploited.

Description: A use after free issue was addressed with improved memory management.

CVE-2021-30858: an anonymous researcher

# CVE-2021-30858

## Changeset 281648 in webkit

**Timestamp:** Aug 26, 2021 1:21:27 PM (5 months ago)

**Author:** mmaxfield@apple.com

**Message:** REGRESSION(r256659): We try to remove fonts from the CSSFontFace which were never added  
➡ [https://bugs.webkit.org/show\\_bug.cgi?id=229535](https://bugs.webkit.org/show_bug.cgi?id=229535)  
<rdar://problem/78857440>

Reviewed by Darin Adler.

After r256659, asking for a failed CSSFontFace's families() returns nullopt. It's possible to add a failed font to a CSSFontFaceSet (of course). When we do that, we recognize the font is failed and don't update our internal data structures, because there's no need to - we can't do anything useful with a failed font.

If you \_then\_ try to remove the font from the CSSFontFace, we don't call families(), but instead just pull out the raw m\_families member, and look in our internal data structures for it, but we don't find it, because it was never added.

- css/CSSFontFaceSet.cpp:

(WebCore::CSSFontFaceSet::addToFacesLookupTable):

(WebCore::CSSFontFaceSet::removeFromFacesLookupTable):

**Location:** [trunk/Source/WebCore](#)

**Files:**  2 edited

 [ChangeLog](#) (1 diff)

 [css/CSSFontFaceSet.cpp](#) (2 diffs)

# The Vulnerability

---

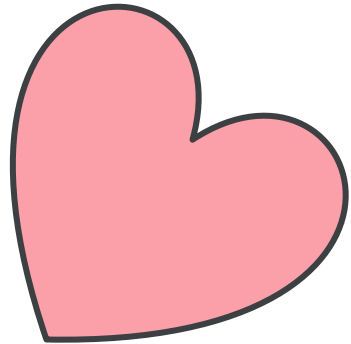
**Bug class:** Use-after-free

**Vulnerability details:** The vulnerability is a use-after-free due to an unchecked `end()` iterator. There was an assert statement:

`ASSERT(iterator != m_facesLookupTable.end());`, but `ASSERT` s don't do anything in release builds. Therefore, even if `iterator == m_facesLookupTable.end()` in the release build, nothing would happen and `iterator` would still be used.

<https://github.com/WebKit/WebKit/blob/74bd0da94fa1d31a115bc4ee0e3927d8b2ea571e/Source/WebCore/css/CSSFontFaceSet.cpp#L223>

```
void CSSFontFaceSet::removeFromFacesLookupTable(const CSSFontFace& face, const CSSValueList& familiesToSearchFor)
{
    for (auto& item : familiesToSearchFor) {
        String familyName = CSSFontFaceSet::familyNameFromPrimitive(downcast<CSSPrimitiveValue>(item.get()));
        if (familyName.isEmpty())
            continue;
    }
}
```



```
var fontFace1 = new FontFace("font1", "", {});  
var fontFaceSet = new FontFaceSet([fontFace1]);  
fontFace1.family = "font2";
```

“Greetings from Apple Product Security”

## Changeset 281384 in webkit

**Timestamp:** Aug 21, 2021 6:33:10 PM (6 months ago)

**Author:** [sihui\\_liu@apple.com](mailto:sihui_liu@apple.com)

**Message:** IndexedDB: crash when triggering IDBOpenRequest completion back on a worker thread

⇒ [https://bugs.webkit.org/show\\_bug.cgi?id=229375](https://bugs.webkit.org/show_bug.cgi?id=229375)

Source/WebCore:

Reviewed by Brady Eidson.

Client may dispatch custom events to an IDBRequest, and we should only change request state based on events created internally.

- Modules/indexeddb/IDBRequest.cpp:

(WebCore::IDBRequest::dispatchEvent):

Source/WTF:





Protect callee in CrossThreadTask if it inherits from ThreadSafeRefCounted<T>.

Reviewed by Brady Eidson.

- wtf/CrossThreadTask.h:

**Location:** [trunk/Source](#)

**Files:**  4 edited

-  [WTF/ChangeLog](#) (1 diff)
-  [WTF/wtf/CrossThreadTask.h](#) (2 diffs)
-  [WebCore/ChangeLog](#) (1 diff)
-  [WebCore/Modules/indexeddb/IDBRequest.cpp](#) (1 diff)



```
- template<typename T, typename std::enable_if<std::is_base_of<ThreadSafeRefCounted<T>, T>::value,  
int>::type = 0, typename... Parameters, typename... Arguments>
```

```
+ template<typename T, typename std::enable_if<std::is_base_of<ThreadSafeRefCountedBase, T>::value,  
int>::type = 0, typename... Parameters, typename... Arguments>
```

```
    CrossThreadTask createCrossThreadTask(T& callee, void (T::*method)(Parameters...), const Arguments&...  
arguments)  
{  
    return CrossThreadTask([callee = makeRefPtr(&callee), method, arguments =  
std::make_tuple(crossThreadCopy(arguments)...)]() mutable {  
        callMemberFunctionForCrossThreadTask(callee.get(), method, WTFMove(arguments));  
    });  
}
```

```
- template<typename T, typename std::enable_if<!std::is_base_of<ThreadSafeRefCounted<T>, T>::value,  
int>::type = 0, typename... Parameters, typename... Arguments>
```

```
+ template<typename T, typename std::enable_if<!std::is_base_of<ThreadSafeRefCountedBase, T>::value,  
int>::type = 0, typename... Parameters, typename... Arguments>
```

```
    CrossThreadTask createCrossThreadTask(T& callee, void (T::*method)(Parameters...), const Arguments&...  
arguments)
```

```
class IDBOpenDBRequest final : public IDBRequest {
```

```
class IDBOpenDBRequest final : public IDBRequest {
```

```
class IDBRequest :
```

```
    public EventTargetWithInlineData, public IDBActiveDOMObject,  
    public ThreadSafeRefCounted<IDBRequest> {
```

```
class IDBOpenDBRequest final : public IDBRequest {
```

```
class IDBRequest :
```

```
    public EventTargetWithInlineData, public IDBActiveDOMObject,  
    public ThreadSafeRefCounted<IDBRequest> {
```

```
- template<typename T,
```

```
    typename std::enable_if<std::is_base_of<ThreadSafeRefCounted<T>, T>
```

```
::value, int>::type = 0, typename... Parameters, typename... Arguments>
```

```
+ template<typename T, typename std::enable_if<std::is_base_of<ThreadSafeRefCountedBase, T>::value,
```

```
    int>::type = 0, typename... Parameters, typename... Arguments>
```

```
    CrossThreadTask createCrossThreadTask(T& callee, void (T::*method)(Parameters...), const Arguments&...
arguments)
    {
        return CrossThreadTask([callee = makeRefPtr(&callee), method, arguments =
std::make_tuple(crossThreadCopy(arguments)...)]() mutable {
            callMemberFunctionForCrossThreadTask(callee.get(), method, WTFMove(arguments));
        });
    }
```

```
- template<typename T, typename std::enable_if<!std::is_base_of<ThreadSafeRefCounted<T>, T>::value,
```

```
    int>::type = 0, typename... Parameters, typename... Arguments>
```

```
+ template<typename T, typename std::enable_if<!std::is_base_of<ThreadSafeRefCountedBase, T>::value,
```

```
    int>::type = 0, typename... Parameters, typename... Arguments>
```

```
    CrossThreadTask createCrossThreadTask(T& callee, void (T::*method)(Parameters...), const Arguments&...
arguments)
```

```
- template<typename T,
```

```
    typename std::enable_if<std::is_base_of<ThreadSafeRefCounted<T>, T> ::value,
```

```
    int>::type = 0, typename... Parameters, typename... Arguments>
```

```
+ template<typename T, typename std::enable_if<std::is_base_of<ThreadSafeRefCountedBase, T>::value, int>::type
```

```
    = 0, typename... Parameters, typename... Arguments>
```

```
    CrossThreadTask createCrossThreadTask(T& callee, void (T::*method)(Parameters...), const Arguments&...
arguments)
    {
        return CrossThreadTask([callee = makeRefPtr(&callee), method, arguments =
std::make_tuple(crossThreadCopy(arguments)...)]() mutable {
            callMemberFunctionForCrossThreadTask(callee.get(), method, WTFMove(arguments)); });
    }
```

```
- template<typename T, typename std::enable_if<!std::is_base_of<ThreadSafeRefCounted<T>, T>::value, int>::type
```

```
    = 0, typename... Parameters, typename... Arguments>
```

```
+ template<typename T, typename std::enable_if<!std::is_base_of<ThreadSafeRefCountedBase, T>::value, int>::type
```

```
    = 0, typename... Parameters, typename... Arguments>
```

```
    CrossThreadTask createCrossThreadTask(T& callee, void (T::*method)(Parameters...), const Arguments&...
arguments)
```

`is_base_of<ThreadSafeRefCounted<IDBRequest>, IDBOpenDBRequest>`

- `template<typename T,`

`typename std::enable_if<std::is_base_of<ThreadSafeRefCounted<T>, T> ::value,`

`int>::type = 0, typename... Parameters, typename... Arguments>`

+ `template<typename T, typename std::enable_if<std::is_base_of<ThreadSafeRefCountedBase, T>::value, int>::type`

`= 0, typename... Parameters, typename... Arguments>`

```
    CrossThreadTask createCrossThreadTask(T& callee, void (T::*method)(Parameters...), const Arguments&...
arguments)
    {
        return CrossThreadTask([callee = makeRefPtr(&callee), method, arguments =
std::make_tuple(crossThreadCopy(arguments)...)]() mutable {
            callMemberFunctionForCrossThreadTask(callee.get(), method, WTFMove(arguments)); });
    }
```

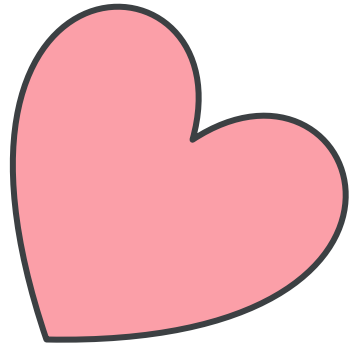
index.html:

```
<script>w = new Worker('idbworker.js');</script>
```

idbworker.js:

```
function gc() {  
    for (var i = 0; i < 1000; i++) { a = new Uint8Array(1024*1024); }  
}
```

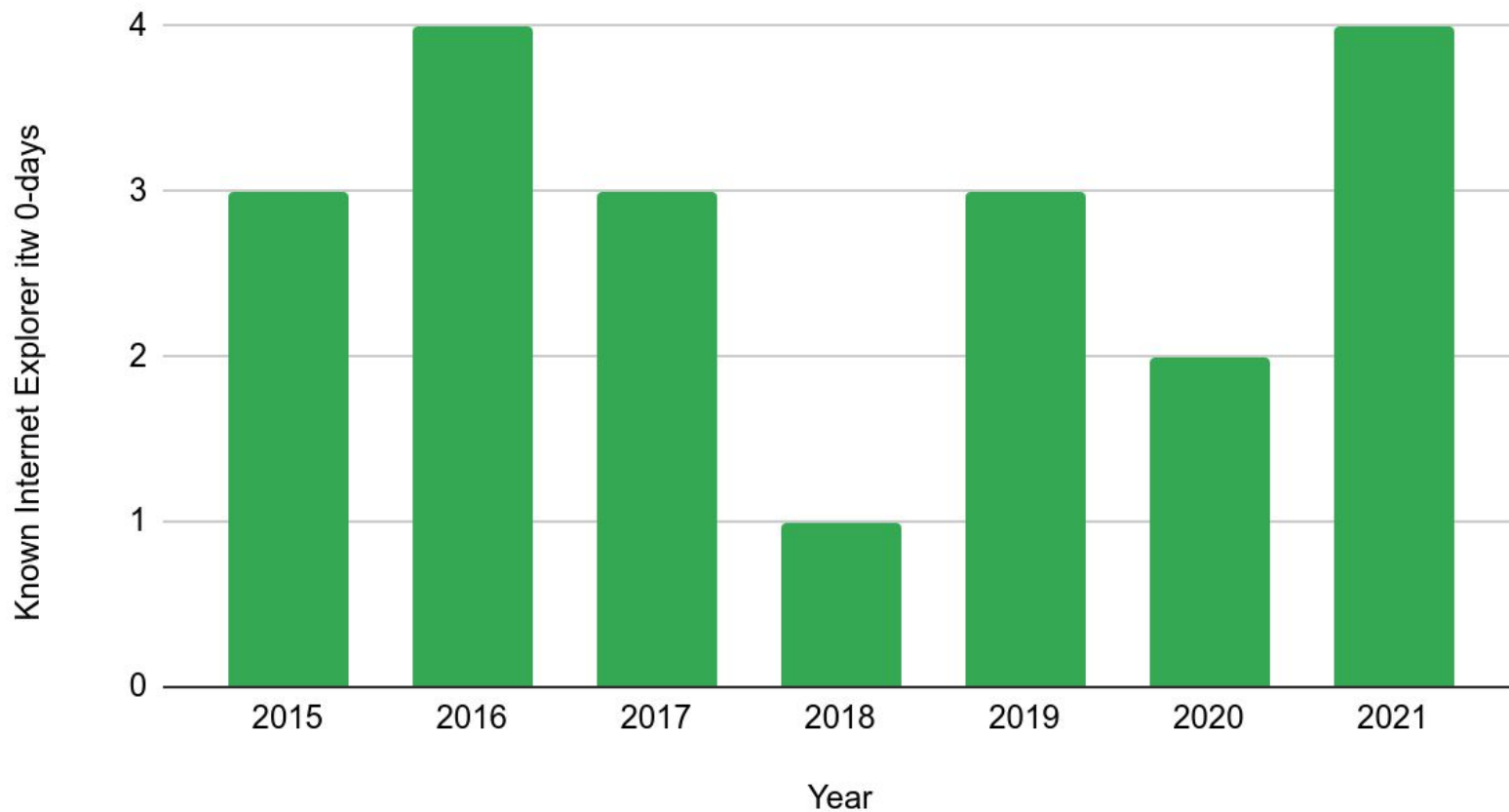
```
let ev = new Event('mine');  
let req = http://indexedDB.open\('db'\);  
req.dispatchEvent(ev);  
req = 0;  
ev = 0;  
gc();
```





# Internet Explorer

## Known Internet Explorer itw 0-days



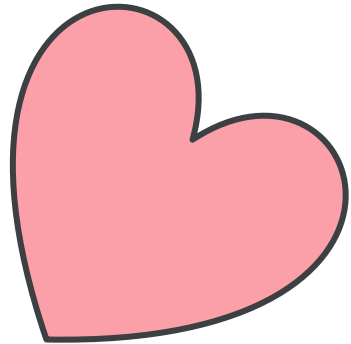
Known Internet Explorer itw 0-days						
	MSHTML	JScript	VBScript	JScript9	Info Leak	Total
2021	3			1		4
2020		1		1		2
2019		2			1	3
2018		1				1
2017			1		2	3
2016		1	1		2	4
2015	1			2		3

Known Internet Explorer itw 0-days						
	MSHTML	JScript	VBScript	JScript9	Info Leak	Total
2021	3			1		4
2020		1		1		2
2019		2			1	3
2018		1				1
2017			1		2	3
2016		1	1		2	4
2015	1			2		3

# CVE-2021-33742: Out of Bounds Write in MSHTML

- Malicious Office docs that loaded web content in Internet Explorer
- Spawned an Internet Explorer process via VBA macros to navigate to a web page
- Web page fingerprinted then delivered exploit
- Out-of-bounds write in MSHTML
  - Size of the string of the inner html element is truncated (size&0x1FFFFFFF) in the CTreePos structure while the non-truncated size is still in the text data object. Memory at is allocated based on the size in the CTreePos structure, the truncated size. The memcpy is based on the non-truncated size.

CVE-2021-33742



```
<script>
```

```
var b = document.createElement("html");  
b.innerHTML = Array(40370176).toString();  
b.innerHTML = "";  
</script>
```

Over the last year, you may have noticed our movement away from Internet Explorer ("IE") support, such as an announcement of the [end of IE support by Microsoft 365 online services](#). Today, we are at the next stage of that journey: we are announcing that the future of Internet Explorer on Windows 10 is in Microsoft Edge. Not only is Microsoft Edge a faster, more secure and more modern browsing experience than Internet Explorer, but it is also able to address a key concern: compatibility for older, legacy websites and applications. Microsoft Edge has Internet Explorer mode ("IE mode") built in, so you can access those legacy Internet Explorer-based websites and applications straight from Microsoft Edge. With Microsoft Edge capable of assuming this responsibility and more, **the Internet Explorer 11 desktop application will be retired and go out of support on June 15, 2022, for certain versions of Windows 10.**

*Note: This retirement does not affect in-market Windows 10 LTSC or Server Internet Explorer 11 desktop applications. It also does not affect the MSHTML (Trident) engine. For a full list of what is in scope for this announcement, and for other technical questions, please [see our FAQ](#).*

Over the last year, you may have noticed our movement away from Internet Explorer ("IE") support, such as an announcement of the [end of IE support by Microsoft 365 online services](#). Today, we are at the next stage of that journey: we are announcing that the future of Internet Explorer on Windows 10 is in Microsoft Edge. Not only is Microsoft Edge a faster, more secure and more modern browsing experience than Internet Explorer, but it is also able to address browser compatibility for older Internet Explorer-based applications. Microsoft Edge has taken on the responsibility of ensuring that Internet Explorer-based websites and applications continue to work. We are retiring Internet Explorer on Windows 10 out of support on June 15, 2022.

This retirement does not affect [...] It also does not affect the **MSHTML (Trident) engine**.

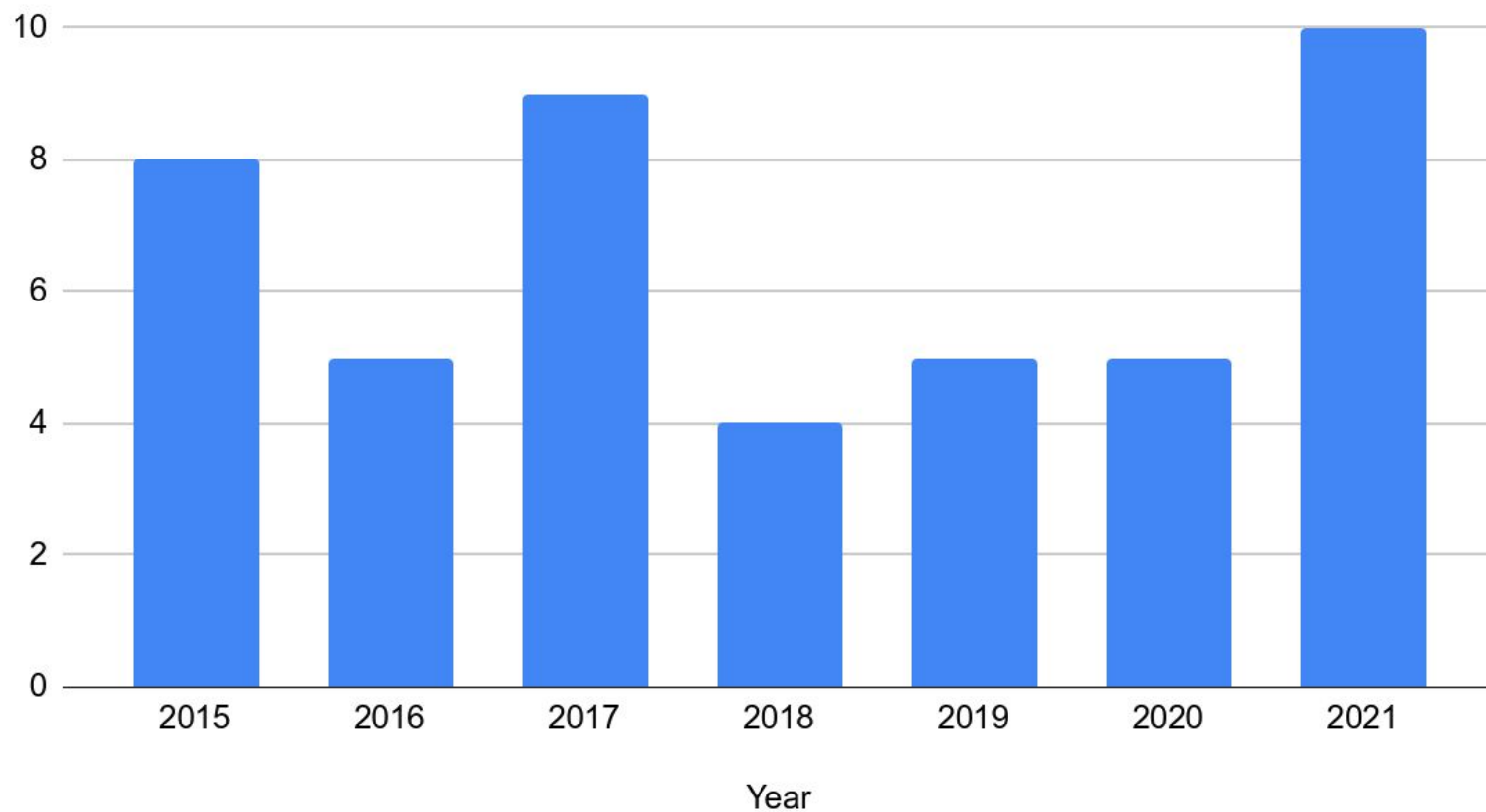
*Note: This retirement does not affect in-market Windows 10 LTSC or Server Internet Explorer 11 desktop applications. It also does not affect the MSHTML (Trident) engine. For a full list of what is in scope for this announcement, and for other technical questions, please [see our FAQ](#).*





# Windows

## Detected Windows itw 0-days





# 10 Windows

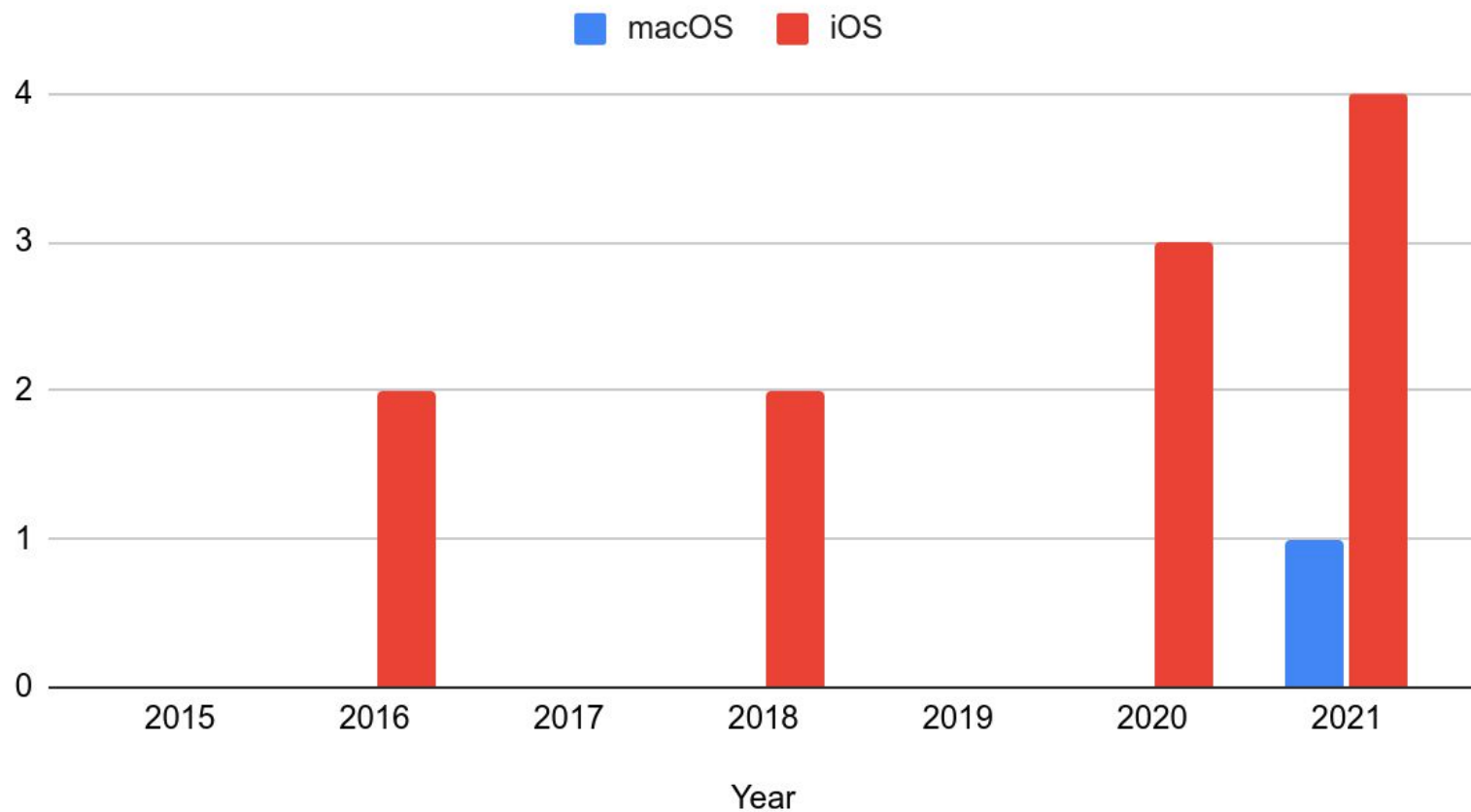
0-days detected in-the-wild

- 2 Enhanced crypto provider
- 2 NTOS kernel
- 2 win32k
- 1 Windows update medic
- 1 SuperFetch
- 1 dwmcore.dll
- 1 ntfs.sys

In 2019, 75% of  
Windows 0-days  
targeted win32k.

# iOS & macOS

## Detected macOS & iOS itw 0-days



# 4 iOS & 1 macOS

0-days detected in-the-wild

- 2 IOMobileFrameBuffer
- 2 XNU (1 iOS & 1 macOS)
- 1 CoreGraphics



# 4 iOS & 1 macOS

0-days detected in-the-wild

- 2 IOMobileFrameBuffer
- 2 XNU (1 iOS & 1 macOS)
- 1 CoreGraphics

# 4 iOS & 1 macOS

0-days detected in-the-wild

- 2 IOMobileFrameBuffer
- 2 XNU (1 iOS & 1 macOS)
- 1 CoreGraphics

**Drum roll please for the best bug  
of the year....**



# The Zero Click iMessage Bug

## CVE-2021-30860

```

Guint numSyms; // (1)
numSyms = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            numSyms += ((JBIG2SymbolDict *)seg)->getSize(); // (2)
        } else if (seg->getType() == jbig2SegCodeTable) {
            codeTables->append(seg);
        }
    } else {
        error(errSyntaxError, getPos(),
            "Invalid segment reference in JBIG2 text region");
        delete codeTables;
        return;
    }
}
...
// get the symbol bitmaps
syms = (JBIG2Bitmap **)gmallocn(numSyms, sizeof(JBIG2Bitmap *)); // (3)
kk = 0;
for (i = 0; i < nRefSegs; ++i) {
    if ((seg = findSegment(refSegs[i]))) {
        if (seg->getType() == jbig2SegSymbolDict) {
            symbolDict = (JBIG2SymbolDict *)seg;
            for (k = 0; k < symbolDict->getSize(); ++k) {
                syms[kk++] = symbolDict->getBitmap(k); // (4)
            }
        }
    }
}
}

```

“My other  
compression format  
is turing-complete!”

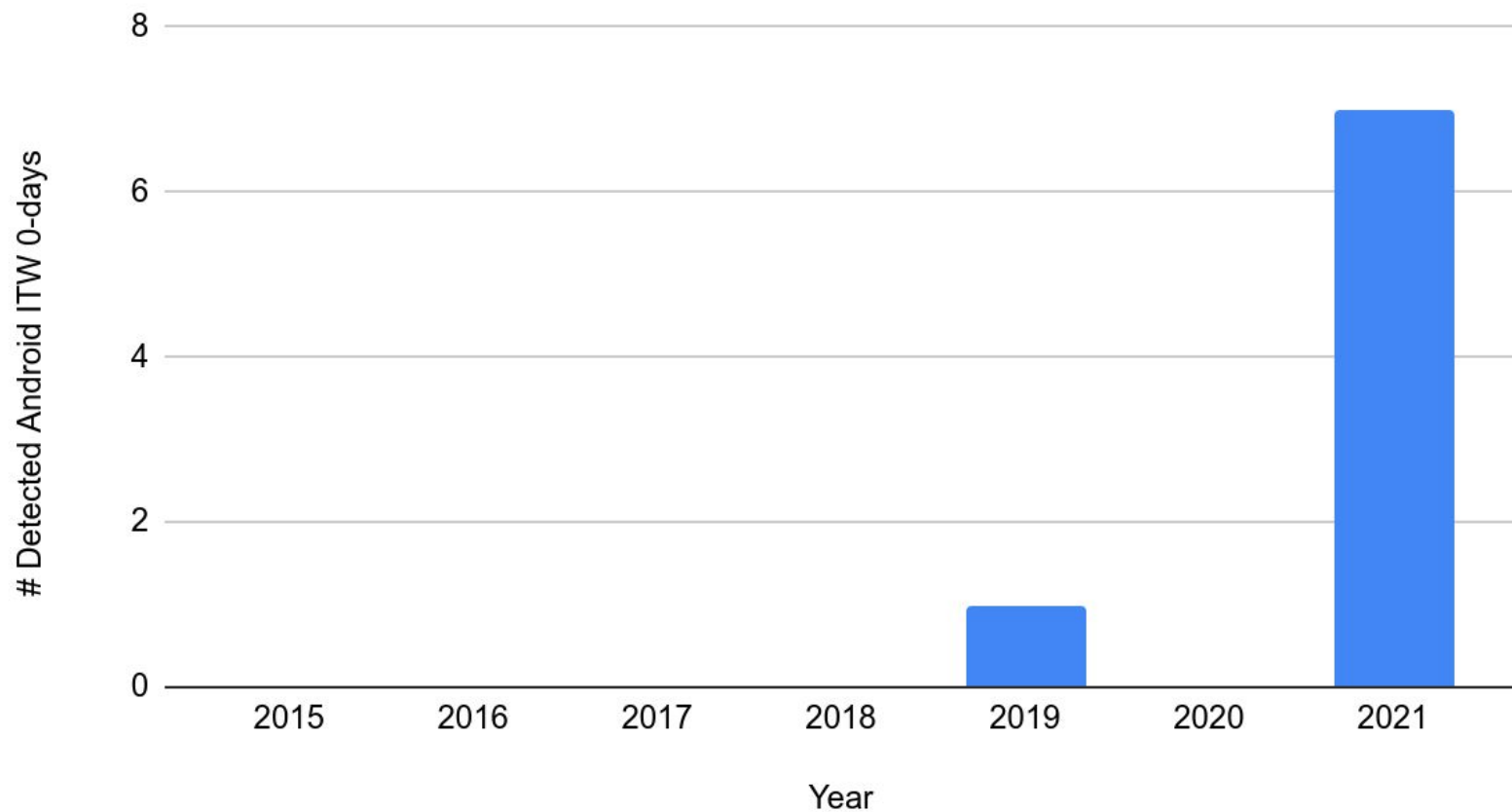
[Ian Beer's blog post](#)



# Android



## Detected Android ITW 0-days



# 7 Android

0-days detected in-the-wild

- 3 Qualcomm Adreno GPU
- 2 ARM Mali GPU
- 2 upstream kernel

# 7 Android

0-days detected in-the-wild

- 3 Qualcomm Adreno GPU
- 2 ARM Mali GPU
- 2 upstream kernel

# 7 Android

0-days detected in-the-wild

- 3 Qualcomm Adreno GPU
- 2 ARM Mali GPU
- 2 upstream kernel

# CVE-2021-1048

```
author      Al Viro <viro@zeniv.linux.org.uk> 2020-09-02 11:30:48 -0400
committer   Al Viro <viro@zeniv.linux.org.uk> 2020-09-02 11:30:48 -0400
commit      77f4689de17c0887775bb77896f4cc11a39b1848 (patch)
tree        48e71e89ec43f9869327fc81f1b8c83ffb60c72d
parent      52c479697c9b73f628140dcdfcd39ea302d05482 (diff)
download    linux-77f4689de17c.tar.gz
```

## fix regression in "epoll: Keep

epoll\_loop\_check\_proc() can r  
we can't grab a reference on  
reverse path check anyway.

Tested-by: Marc Zyngier <maz@kernel.org>

Fixes: a9ed4a6560b8 ("epoll: Keep a reference on files added to the check list")

Signed-off-by: Al Viro <viro@zeniv.linux.org.uk>

## Diffstat

```
-rw-r--r-- fs/eventpoll.c 6
```

1 files changed, 3 insertions, 3 deletions

```
diff --git a/fs/eventpoll.c b/fs/eventpoll.c
```

```
index e0decff22ae27..8107e06d7f6f5 100644
```

```
--- a/fs/eventpoll.c
```

```
+++ b/fs/eventpoll.c
```

```
@@ -1995,9 +1995,9 @@ static int ep_loop_check_proc(void *priv, void *cookie, int call_nests)
        * during ep_insert().
        */
```

2020-09-02 11:30:48 -0400

## 2021-11-06 security patch level vulnerability details

In the sections below, we provide details on the vulnerabilities included in the 2021-11-06 security patch. Vulnerabilities are grouped under the following categories: CVE ID, associated references, type of vulnerability, and severity. We link the public change that addresses the vulnerability to a single bug, additional references, and the upstream patch.

Not included in Android Security Bulletin until November 2021. 14 months after upstream patch.

### Kernel components

The vulnerability in this section could lead to a local escalation of privilege due to a use after free.

CVE	References	Type	Severity	Component
CVE-2021-1048	A-204573007 <a href="#">Upstream kernel</a>	EoP	High	Kernel

```
author      Al Viro <viro@zeniv.linux.org.uk> 2020-09-02 11:30:48 -0400
committer   Al Viro <viro@zeniv.linux.org.uk> 2020-09-02 11:30:48 -0400
commit      77f4689de17c0887775bb77896f4cc11a39bf848 (patch)
tree        48e71e89ec43f9869327fc81f1b8c83ffb60c72d
parent      52c479697c9b73f628140dcd39ea302d05482 (diff)
download    linux-77f4689de17c.tar.gz
```

### fix regression in "epoll: Keep a reference on files added to the check list"

epoll\_loop\_check\_proc() can run into a file already committed to destruction;  
we can't grab a reference on those and don't need to add them to the set for  
reverse path check anyway.

Tested-by: Marc Zyngier <maz@kernel.org>

Fixes: a9ed4a6560b8 ("epoll: Keep a reference on files added to the check list")

Signed-off-by: Al Viro <viro@zeniv.linux.org.uk>

Fixes: a9ed4a6560b8 ("epoll: Keep a reference on  
files added to the check list")

```
--- a/fs/eventpoll.c
+++ b/fs/eventpoll.c
@@ -1995,9 +1995,9 @@ static int ep_loop_check_proc(void *priv, void *cookie, int call_nests)
     * during ep_insert().
     */
     if (list_empty(&epi->ffd.file->f_tfile_llink)) {
-        get_file(epi->ffd.file);
-        list_add(&epi->ffd.file->f_tfile_llink,
-                &tfile_check_list);
+        if (get_file_rcu(epi->ffd.file))
+            list_add(&epi->ffd.file->f_tfile_llink,
+                    &tfile_check_list);
     }
 }
```

“The more you know, the more you realize you don’t know.” – Aristotle



# Where are the

Messaging app

Other phone components

Linux

Cloud

Specific Android OEM

0-days?

Detection, Disclosure, or Both?

2019 Reporters	2020 Reporters	2021 Reporters
Alibaba Cloud Intelligence Security Team	Andy	cPanel Security Team
Coinbase Security	Codesafe Team of Legendsec at Qi'anxin Group	DBAppSecurity Co., Ltd
ESET	Francisco Alonso & Javier Marcos	DEVCORE Research Team
Google Project Zero	Google Project Zero	Dubex
Google Threat Analysis Group	Google Threat Analysis Group	Enki
Kaspersky Lab	Kaspersky Lab	EXPMON
Microsoft Security Response Center	National Security Agency	Github Security Lab
Resecurity, Inc.	Qihoo 360 ATA	Google Project Zero
Trend Micro	Trend Micro Research	Google Threat Analysis Group
	Trend Micro's Zero Day Initiative	Kaspersky Lab
		Mandiant
		Mattias Buelens
		Microsoft Browser Vulnerability Research
		Microsoft Security Response Center
		Microsoft Threat Intelligence Center
		Qihoo 360 ATA
		Sangfor
		The Citizen Lab
		Tianfu
		Volexity

Do these bug patterns look like previous bugs because that's what we know how to detect?

Where are the spl0itz?

5 out of 57 0-days  
have the exploit  
sample publicly  
available.

Big uptick in in-the-wild 0-days = 😄



Big uptick in in-the-wild 0-days = 😄

Bug patterns & types are similar to  
previous years = 😐



Big uptick in in-the-wild 0-days = 😄

Bug patterns & types are similar to previous years = 😐

Still room for improvement on detection & disclosure = 🙄🙏

What do you think?

# Hopes & Dreams

- Root cause analysis, patch analysis, and variant analysis performed on every in-the-wild 0-day.
- Exploit samples are shared widely.
- Vendors transparently annotate exploited status.
- More, more, more detection.



# THANK YOU!

@maddiestone

# RCE in Windows Defender CVE-2021-1647

# CVE-2021-1647

- Heap buffer overflow when mpengine.dll is processing the section table to unpack an ASProtect packed executable.
- Each entry in the section table has two values:
  - Virtual address of section & section size

# CAsprotectDLLAndVersion::RetrieveVersionInfoAndCreateObjects

Example section table: [ (0,0), (0,0), (0x2000,0), (0x2000,0x3000) ]

```
if ( next_sect_addr > sect_addr )
{
    sect_addr = next_sect_addr;
    sect_sz = (next_sect_sz + 0xFFF) & 0xFFFFF000;
}
// if next_sect_addr <= sect_addr we continue on to next entry in the table

[...]  
    new_sect_alloc = operator new[](sect_sz + sect_addr);  
[...]
```

# CAsprotectDLLAndVersion::RetrieveVersionInfoAndCreateObjects

Example section table: [ (0,0), (0,0), (0x2000,0), (0x2000,0x3000) ]

```
if ( next_sect_addr > sect_addr )  
{  
    sect_addr = next_sect_addr;  
    sect_sz = (next_sect_sz + 0xFFF) & 0xFFFFF000;  
}
```

// if next\_sect\_addr <= sect\_addr we continue on to next entry in the table

[...]

```
new_sect_alloc = operator new[](sect_sz + sect_addr);
```

[...]



# CAsprotectDLLAndVersion::RetrieveVersionInfoAndCreateObjects

Example section table: [ (0,0), (0,0), (0x2000,0), (0x2000,0x3000) ]

```
if ( next_sect_addr > sect_addr )
{
    sect_addr = next_sect_addr;
    sect_sz = (next_sect_sz + 0xFFF) & 0xFFFFF000;
}
// if next_sect_addr <= sect_addr we continue on to next entry in the table
```

[...]

```
new_sect_alloc = operator new[](sect_sz + sect_addr);
```

[...]

# CAsprotectDLLAndVersion::RetrieveVersionInfoAndCreateObjects

Example section table: [ (0,0), (0,0), (0x2000,0), (0x2000,0x3000) ]

```
if ( next_sect_addr > sect_addr )
{
    sect_addr = next_sect_addr;
    sect_sz = (next_sect_sz + 0xFFF) & 0xFFFFF000;
}
// if next_sect_addr <= sect_addr we continue on to next entry in the table
[...]
```

new\_sect\_alloc = operator new[](sect\_sz + sect\_addr);

```
[...]
```