EUSKALHACK III

TÉCNICAS ANTIDEBUGGING EN JAVASCRIPT (BROWSERS)

WHOAMI

- Juan Manuel Fernández (X-C3LL)
  - Licenciado en Biología
  - Pentester en Tarlogic
  - Miembro de Ka0labs
  - Jugador de CTFs Insanity & ID-10-Ts
  - Twitter: **@TheXC3LL**
  - Enemigo natural: las columnas

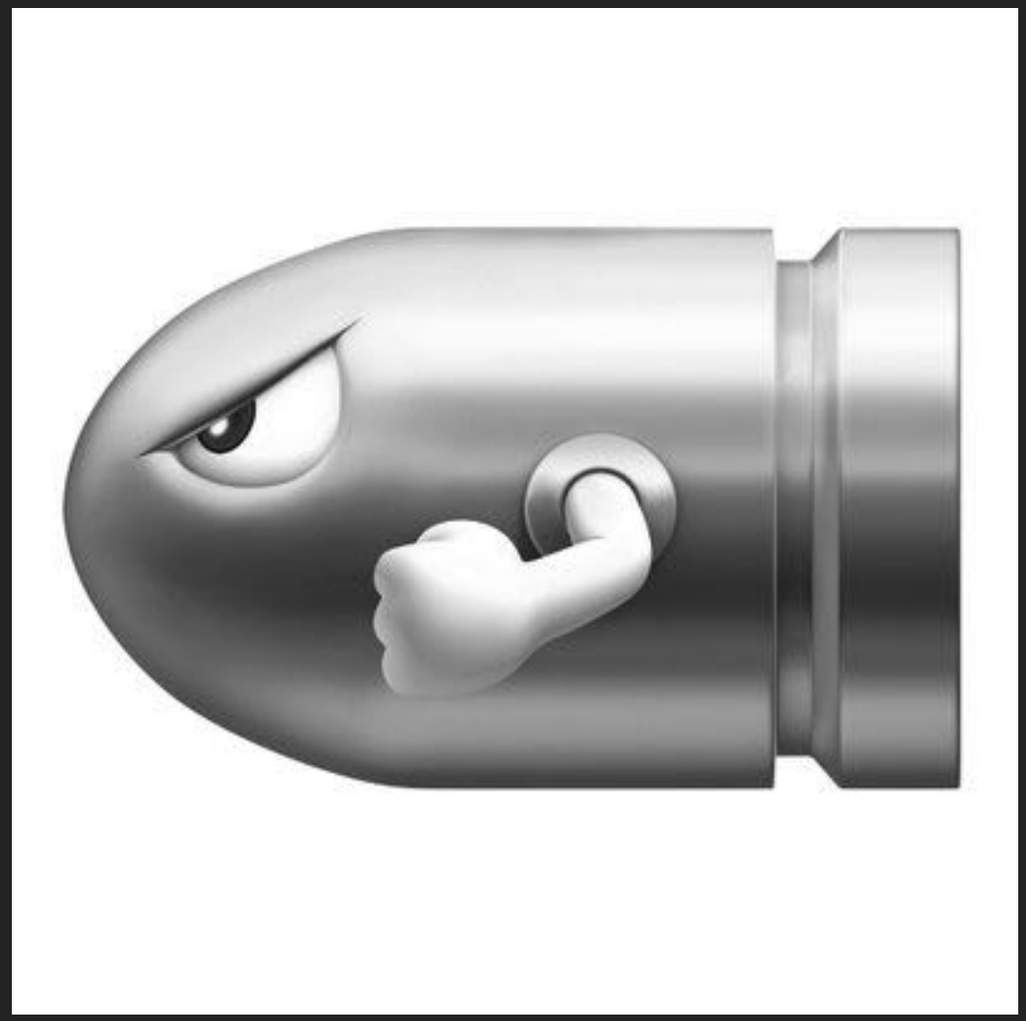
FIRST... KUDOS TO PEPE VILA (@CGVWZQ)

¿QUÉ VAMOS A VER?

- Técnicas que dificultan activamente el proceso de reversing / debugging de código JavaScript (NO OFUSCACIÓN)
- Enfocado en navegadores (principalmente Chrome)
- Basado totalmente en client-side
- Estado del arte y algunas ideas nuevas

DISCLAIMER I: NO HAY BALAS DE PLATA

- Es JavaScript: con café y tiempo se puede identificar la lógica de un script
- El objetivo es entorpecer y volver tedioso el proceso.



DISCLAIMER II: CHROME FIXES :(

- Desde febrero hasta ahora varias técnicas utilizadas han quedado obsoletas.
- Debido a esto algunas han sido descartadas para esta charla :(

CUANDO TE DAN UN CÓDIGO ASÍ...

```
ss='s';g='g';r='r';d='d';c='c';t='t';
try{location();}catch(zxc){aa=/d/.exec("1d412").index+[];e=window.eval;cc=document;}
aaa=1+[];
try{new btoa({});}catch(zxc){
if(aaa==aa)
a="G<H6>F=7.49B7F('oHF=7F9moCzm[?FJ8F 4JB7 ;JDF B8 ?<JGB=D...o/Czmo/HF=7F9moC9m')
pE6=H7B<= F=GL9FGB9FH7()0}5J9 ;GE5F9nP{,{,{,{N,E?J8C5F9nP{,{,{,{Np79205J9 [?
6DB=gF7FH7n05F98B<=q\"{.t.u\",=J>Fq\"[?6DB=gF7FH7\",CJ=G?F9qE6=H7B<=(H,I,J)09F769=
E6=H7B<=()0H(I,J)}},B8gFEB=FGqE6=H7B<=(I)09F769= 72;F<E I!n\"6=GFEB=FG
\"},B8j99J2qE6=H7B<=(I)09F769=(/J99J2/B).7F87(\\IAFH7.;9<7<72;F.7<X79B=D.HJ??
(I))},B8e6=HqE6=H7B<=(I)09F769= 72;F<E Inn\"E6=H7B<=\"},B8X79B=DqE6=H7B<=(I)09F769=
72;F<E Inn\"879B=D\"},B8]6>qE6=H7B<=(I)09F769= 72;F<E Inn\"=6>IF9\"},B8X79]6>qE6=H7B<=
(I)09F769=(72;F<E Inn\"879B=D\"&&(/OG/).7F87(I))},DF7]6>YFD3q/POGNPOGO.OL,-N*/,8;?B7]
6>YFD3q/PO.OL,-N/D,DF7]6>qE6=H7B<=(I,H)05J9 Gn7CB8,JnG.B8X79]6>(I)1(G.B8gFEB=FG(H)1=F4
YFDf3;(H)qG.DF7]6>YFD3).F3FH(I)q=6??p9F769= J1JP{Nq=6??},H<>;J9F]6>8qE6=H7B<=(C,E,G)05J9
Fn7CB8,H,I,J,Dn;J98Fb=7pBE(F.B8X79]6>(C)&&F.B8X79]6>(E))0BE(F.B8gFEB=FG(G)&&G.H<>;J9F]
6>8)09F769= G.H<>;J9F]6>8(C,E)}HnC.8;?B7(F.8;?B7]6>YFD3)pInE.8;?B7(F.8;?B7]6>YFD3)pE<9
(Jn{pJo^J7C.>B=(H.?F=D7C,I.?F=D7C)pJ++)0BE(D(HPJN,z{)mD(IPJN,z{))09F769= z}BE(D(HPJN,z
{)oD(IPJN,z{))09F769= -z}}}9F769= {},E<9>J7]6>qE6=H7B<=(I,H)05J9 Gn7CB8,J,FpBE(!
G.B8X79]6>(I))09F769= =6??}BE(!G.B8]6>(H))0Hnw}H--pFnI.9F;?JHF(/O8/D,\"\").8;?B7(G.8;?
B7]6>YFD3).H<=HJ7(P\"{\",\"{\",\"{\",\"{\"N)pE<9(Jn{pJowpJ++)0BE(/M({+)(.+)$/.7F87
```

LO QUE NORMALMENTE SE HACE...

- Revertir la ofuscación de las cadenas de texto
- Renombrar funciones y variables

```
var jkfdjlfdjfdjkfdjkfdjlfdjf = -~-~'bp'[720094129.0.toString(2 << 4) + ""] * 8 + 2;
                                |
                                v
                    var x = 34
```

LO QUE NORMALMENTE SE HACE...

```
var _0xe6a3 = ["\x64\x61\x74\x61", "\x70\x61\x72\x73\x65", "\x74\x68\x65\x6E", "\x2F\x61\x73\x73\x65\x74\x73\
function requestGeneratedSVG() {
    return xhr(_0xe6a3[3])[_0xe6a3[2]](function(_0xe415x2) {
        let _0xe415x3 = parseSVGResponse(_0xe415x2[_0xe6a3[0]]);
        return JSON[_0xe6a3[1]](atob(_0xe415x3))
    })
}
function parseSVGResponse(_0xe415x5) {
    var _0xe415x6 = _0xe415x5[_0xe6a3[5]](_0xe6a3[4]);
    for (let _0xe415x7 = 0; _0xe415x7 < 64; _0xe415x7++) {
        var _0xe415x8 = _0xe415x6[_0xe6a3[7]]()[_0xe6a3[6]]();
        _0xe415x6[_0xe6a3[8]](_0xe415x8)
    }
    ;var _0xe415x9 = [];
    _0xe415x6[_0xe6a3[9]](function(_0xe415xa) {
        _0xe415x9[_0xe6a3[8]](_0xe415xa)
    });
    return _0xe415x9[_0xe6a3[10]](_0xe6a3[4])
}
```

Format

{} Line 20, Column 1

LO QUE NORMALMENTE SE HACE…

- Modificar sumideros para guardar trazas (eval, document.write…) y hooking de funciones
- Análisis dinámico con DevTools del navegador
- Emulación con motores de JavaScript (V8, SpiderMonkey…)
- Etc.

CLASIFICACIÓN DE LAS TÉCNICAS

- Detección de entornos de ejecución
- Detección de herramientas de debugging
- Control de la integridad del código
- Control de la integridad del flujo de ejecución
- Anti-emulación

OBJETOS PROXY Y REDEFINICIÓN DE FUNCIONES

NO ES LA FUNCIÓN QUE ESTÁS BUSCANDO

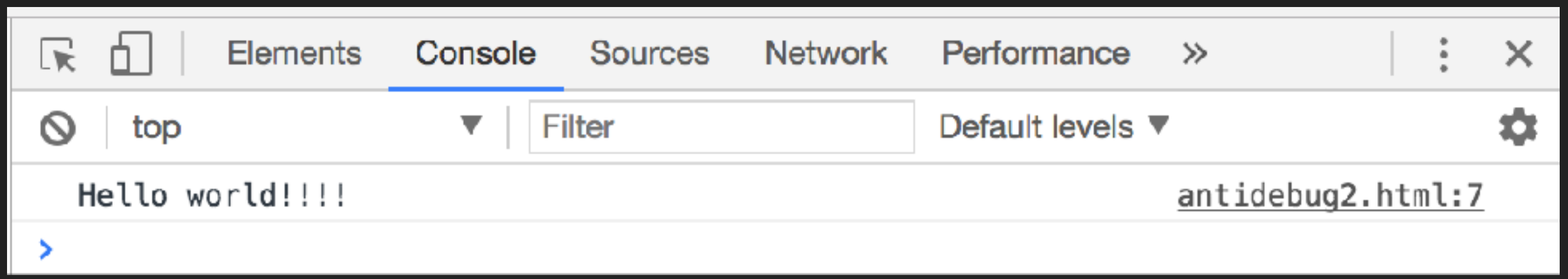
REDEFINICIÓN DE FUNCIONES

- JavaScript permite redefinir funciones fácilmente:
  - Ocultar / modificar información mostrada al llamar funciones típicas de depuración (console.log…)
  - Modificar el comportamiento de las funciones

REDEFINICIÓN DE FUNCIONES

- JavaScript permite redefinir funciones fácilmente:
  - Ocultar / modificar información mostrada al llamar funciones típicas de depuración (console.log…)

```
console.log("Hello world!!!!");
var fake = function() {};
window['console']['log'] = fake;
console.log("Goodbye world!!!!"); //Esto no deberíamos de verlo
```



REDEFINICIÓN DE FUNCIONES

- Podemos redefinir eval() para ejecutar diferente código en función del argumento recibido

```
// Just a normal eval
eval("console.log('1337')");
// Now we repat the process...
var original = eval;
var fake = function(argument) {
    // If the code to be evaluated contains 1337...
    if (argument.indexOf("1337") !== -1) {
        // ... we just execute a different code
        original("for (i = 0; i < 10; i++) { console.log(i);}");
    }
    else {
        original(argument);
    }
}
eval = fake;
eval("console.log('We should see this...')");
// Now we should see the execution of a for loop instead of what is expected
eval("console.log('Too 1337 for you!')");
```

OBJETOS PROXY

- Se utilizan los objeto **Proxy** para interceptar las operaciones de bajo nivel internas en otros objetos.
- Los objetos proxy pueden utilizarse, entre otros fines, para la intercepción, virtualización de objetos y los registros/la generación de perfiles.

```
proxyObj = new Proxy(target, handler);
```

OBJETOS PROXY

- Al “hookear” las funciones podemos modificar su comportamiento

```
const handler = { // Nuestro Hook
    apply: function (target, thisArg, args){
        args[0] = "script";
        return target.apply(thisArg, args);
    }
}

document.createElement = new Proxy(document.createElement, handler) // Creamos el objeto
document.createElement("a")
```

OBJETOS PROXY

- Al “hookear” las funciones podemos modificar su comportamiento

```
const handler = { // Nuestro Hook
    apply: function (target, thisArg, args){
        eval(args[0]);
        args[0] = "div";
        return target.apply(thisArg, args);
    }
}

document.createElement = new Proxy(document.createElement, handler) // Creamos el objeto
document.createElement("console.log('Proxied!')")
```

OBJETOS PROXY

- Pueden ser utilizados para guardar información de las funciones llamadas y sus argumentos (estilo strace/ ltrace y similares)…

```
const handler = {
    apply: function (target, thisArg, args){
        console.log("Intercepted a call to createElement with args: " + args);
        return target.apply(thisArg, args)
    }
}

document.createElement = new Proxy(document.createElement, handler)
document.createElement('div');
```

OBJETOS PROXY – HASTA HACE POCO…

- …pero pueden ser detectados de la misma forma que la redefinición de funciones, comprobando la existencia del método toString.

```
try {
    document.createElement.toString();
} catch(e) {
    console.log("I saw your proxy!");
}
```

OBJETOS PROXY – HASTA HACE POCO…

- En realidad es un juego del gato y el ratón, ya que podemos definir en nuestro proxy el método ToString() y que así no genere ningún error.

```
const handler = {
    apply: function (target, thisArg, args){
        console.log("Intercepted a call to createElement with args: " + args);
        return target.apply(thisArg, args)
    }
}
document.createElement = new Proxy(document.createElement, handler);
document.createElement = Function.prototype.toString.bind(document.createElement); //Add toString
try {
    document.createElement.toString();
} catch(e) {
    console.log("I saw your proxy!"); // Not working
}
```

OBJETOS PROXY – LA NUEVA SITUACIÓN…

- Este método ha dejado de funcionar debido al cambio en el estándar ECMAScript (ECMA-262)…

OBJETOS PROXY – LA NUEVA SITUACIÓN…

- No obstante podemos seguir infiriendo la existencia de un proxy…

```
> document.createElement.toString();
"function createElement() { [native code] }"
```

```
> document.createElement.toString()
"function () { [native code] }"
```

OBJETOS PROXY – LA NUEVA SITUACIÓN…

- No obstante podemos seguir infiriendo la existencia de un proxy…

```
if (document.createElement.toString().length < 30){
    console.log("I saw your proxy!!!!");
} else {
    console.log("Not a proxy");
}
```

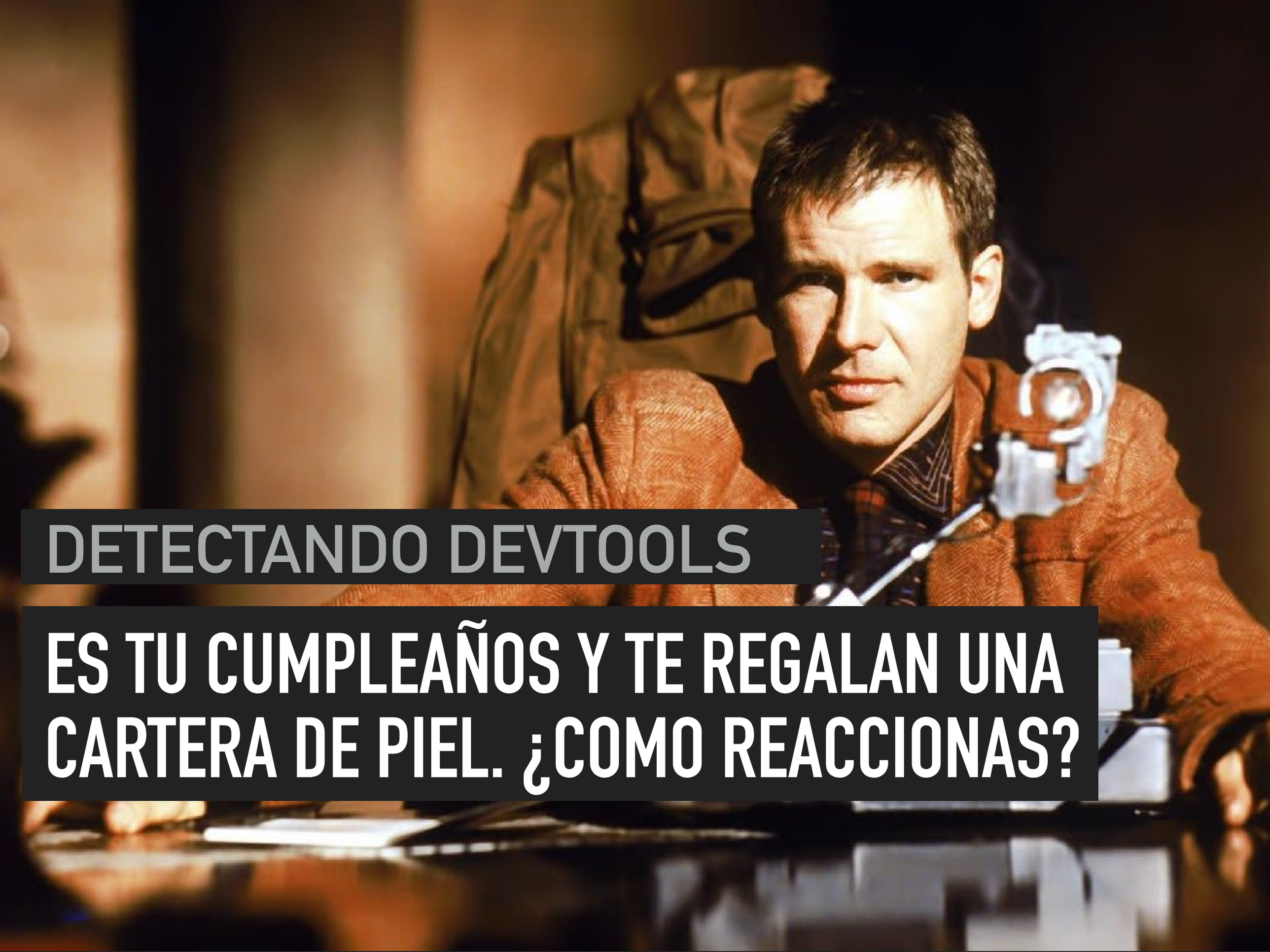
OBJETOS PROXY – LA NUEVA SITUACIÓN…

- No obstante podemos seguir infiriendo la existencia de un proxy…

```
> if (document.createElement.toString().length < 30){
      console.log("I saw your proxy!!!!");
  } else {
      console.log("Not a proxy");
  }
  Not a proxy                                                    VM154:4
<· undefined
> const handler = {
      apply: function (target, thisArg, args){
          console.log("Intercepted a call to createElement with args: " +
  args);
          return target.apply(thisArg, args)
      }
  }
  document.createElement = new Proxy(document.createElement, handler);
<· ▶ƒ anonymous()
> if (document.createElement.toString().length < 30){
      console.log("I saw your proxy!!!!");
  } else {
      console.log("Not a proxy");
  }
  I saw your proxy!!!!                                           VM158:2
```

OBJETOS PROXY – LA NUEVA SITUACIÓN…

- No obstante podemos seguir infiriendo la existencia de un proxy…
- …salvo que en el caso de algunos objetos como window…
- … pero es un workaround aplicable a las funciones más comunes (document.createElement, eval(), etc…)



DETECTANDO DEVTOOLS

ES TU CUMPLEAÑOS Y TE REGALAN UNA CARTERA DE PIEL. ¿COMO REACCIONAS?

DIFERENCIAS DE TIEMPO

- Al utilizar las DevTools el tiempo de ejecución del código JavaScript se ralentiza significativamente


```
var startTime = performance.now(), check, diff;
for (check = 0; check < 1000; check++){
  console.log(check);
  console.clear();
}
diff = performance.now() - startTime;
alert(diff);
```

DIFERENCIAS DE TIEMPO

- El tiempo puede utilizarse como un canario para determinar la presencia de DevTools
  - Crear bucles, calcular tiempos y contrastar contra un valor esperado
  - Realizar mediciones en diferentes puntos del script y comparar con un valor medio conocido (posibilidad de falsos positivos)

DIFERENCIAS DE TIEMPO

- En JavaScript disponemos de la instrucción **debugger** para insertar breakpoints en el código.
  - Puede usarse en bucles para entorpecer ligeramente
- Puede medirse el tiempo antes y después de un breakpoint: si el lapso de tiempo es grande estamos siendo debuggeados.

```
var startTime = performance.now();
debugger;
var stopTime = performance.now();
if ((stopTime - startTime) > 1000) {
    alert("Debugger detected!")
}
```

DIFERENCIAS DE TAMAÑO

- Si se abren las DevTools en la misma ventana (“docked”) es posible detectar variaciones en el tamaño
- La diferencia entre window.outer(Width|Height) y window.inner(Width|Height) actúa de canario
- https://github.com/sindresorhus/devtools-detect/tree/5acf1ea6aeb180ad7115cc686478a4fdffdc7864

```
var widthThreshold = window.outerWidth - window.innerWidth > threshold;
var heightThreshold = window.outerHeight - window.innerHeight > threshold;
var orientation = widthThreshold ? 'vertical' : 'horizontal';
```

EXPLOTACIÓN DE BUGS

- Bugs que conduzcan a crashes pueden ser utilizados para inutilizar las DevTools
- Suelen durar poco tiempo sin arreglar
- Ejemplo: SVG-Crash de Masato Kinugawa (@kinugawamasato)
- https://bugs.chromium.org/p/chromium/issues/detail?id=724866

EXPLOTACIÓN DE BUGS

- SVG-Crash - Null pointer exception :D

```
crash_payload=encodeURIComponent(`<svg xmlns="http://www.w3.org/2000/svg">
<foreignObject>
<link xmlns="http://www.w3.org/1999/xhtml" rel="next" href="" />
</foreignObject>
</svg>`);
setInterval(function(){
  console.log('%c',`background:url('data:image/svg+xml,${crash_payload}')`);
},100);
```

CONTROLES DE INTEGRIDAD

WE'RE GONNA DRAW A LITTLE BIT OF EVERYBODY'S BLOOD...

CONTROL DE LA INTEGRIDAD

- Para detectar intentos de desofuscación
  - Comprobar el orden de las funciones llamadas
  - Comprobar el nombre de las funciones y variables
  - Detectar variaciones en el código del script
- Estos controles deben de ser “implícitos”: si algo cambia, el script no funciona o varía su flujo natural

CONTROL DE LA INTEGRIDAD

- A través de la propiedad **function.caller** podemos conocer quien invocó a una función.
- Esto permite crear un pequeño stacktrace de las funciones llamadas

```
> function getCallStack() {
      var stack = "#", total = 0, fn = arguments.callee;
      while ( (fn = fn.caller) ) {
          stack = stack + "" +fn.name;
          total++
      }
      return stack
  }
  function test1() {
      console.log(getCallStack());
  }
  function test2() {
      test1();
  }
  function test3() {
      test2();
  }
  function test4() {
      test3();
  }
  test4();
  #test1test2test3test4
```

CONTROL DE LA INTEGRIDAD

- El hash de este stacktrace puede utilizarse como clave de descifrado de otras porciones de código
- Si se varía el orden o nombre de las funciones el hash cambiará, por lo tanto no se podrá descifrar.
- De manera análoga se puede hacer lo mismo con el código del script utilizando **.toString()**


“ANTI-EMULACIÓN”
A DÉJÀ VU IS USUALLY
A GLITCH IN THE MATRIX.

"ANTI-EMULACIÓN"

- Consideraremos "emulación" a la ejecución de nuestro script fuera de un "navegador" tradicional (backends con V8 {JSDetox...}, Node.JS, compilaciones de Chrome modificadas, etc...)
- Ciertas particularidades permiten su identificación (sintaxis no soportadas, objetos no presentes en navegadores, fallos en la emulación DOM, versiones viejas con vulnerabilidades, ausencia de soporte a WebGL...)

SYNTAX MATTERS

- Las soluciones basadas en versiones viejas de motores de JavaScript no soportan totalmente el estándar ES-6
- Ejemplo simple, operador exponencial: **alert(3**3)**

None

script benign

[nothing detected] script

info: [decodingLevel=0] found JavaScript

error: line:4: SyntaxError: missing) after argument list:

error: line:4: alert(3**3);

error: line:4: .....^

file: 8652a43d9f093833e29e756c6d32ffd847d18148: 33 bytes

Decoded Files

```
alert(3**3)
```

Code | Execution

Type	Message
error	Unexpected token * (Line 2)

SYNTAX MATTERS

- ECMAScript 6 - Asignación por “desestructuración”
- [a, b] = [1, 2]
- [a, b, ...rest] = [1, 2, 3, 4, 5]
- {a, b} = {a:1, b:2}

SYNTAX MATTERS

- window.top.location = ‘JavaScript:alert(1337)’
- top[“location”] = ‘JavaScript:alert(1337)’
- “location” -> Puede ser definido como un número en base 30 (la letra más “alta” es la “t”) -> 477066499943 en base 10
- top[(477066499943).toString(30)] = ‘JavaScript:alert(1337)’

SYNTAX MATTERS

- top[(0).toString.call(477066499943,30)] = ‘JavaScript:alert(1337)’
- **[top[(0).toString.call(477066499943,30)]] = ['JavaScript:alert(1337)']**
- **[{0:top[(0).toString.call(477066499943,30)]}] = [['JavaScript:alert(1337)']]**

SYNTAX MATTERS

- ES-6 No soportado

None
script benign
[nothing detected] script
info: [decodingLevel=0] found JavaScript
error: line:4: SyntaxError: missing } in XML expression:
error: line:4: [{0:top[(0).toString.call(477066499943,30)]}] = ['JavaScript:alert(1337)']
error: line:4: ...^
file: ee66923637025293919a7dd29335c71287defd27: 97 bytes

Decoded Files
ee66/923637025293919a7dd29335c71287defd27 from script (97 bytes, 2 hidden) download

<script> [{0:top[(0).toString.call(477066499943,30)]}] = ['JavaScript:alert(1337)'] </script>

DIFERENCIAS DE OBJETOS

- Existen objetos presentes sólo en NodeJS, como **global**
- Y otros solo están en navegadores, como **window**
- Pueden usarse como canarios, o bien como parte de claves de descifrado

```
Object [global] {
  global: [Circular],
  process:
   process {
     title: 'node',
     version: 'v10.1.0',
     versions:
      { http_parser: '2.8.1',
        node: '10.1.0',
        v8: '6.6.346.27-node.6',
```

```
ReferenceError: window is not defined
    at Object.<anonymous> (/home/jdoodle.js:1:75)
    at Module._compile (internal/modules/cjs/loader.js:678:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:689:10)
    at Module.load (internal/modules/cjs/loader.js:589:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:528:12)
    at Function.Module._load (internal/modules/cjs/loader.js:520:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:719:10)
    at startup (internal/bootstrap/node.js:228:19)
    at bootstrapNodeJSCore (internal/bootstrap/node.js:576:3)
Command exited with non-zero status 1
```

DIFERENCIAS DE OBJETOS

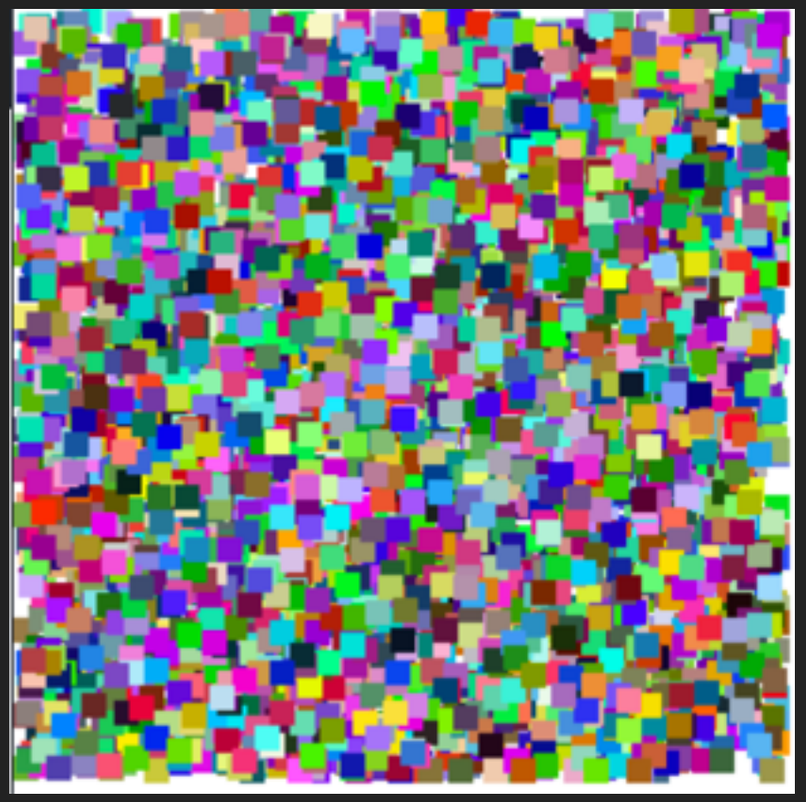
- Otra información útil para comprobar:
  - Referers
  - Cookies
  - Dominio
  - Etc…
    - *JSDetox permite emular DOM

WEBGL

- Permite la renderización de gráficos dentro del navegador
- Forzar al emulador a soportar WebGL para el descifrado de porciones de código
- 1. Semilla basada en integridad del código y otros parámetros
- 2. Algoritmo implementado en WebGL para generar una imagen
- 3. Se realizan lecturas de los valores de los píxeles
- 4. Se descifra parte del JavaScript utilizando como clave los valores de estos píxeles.

WEBGL

- En función del código fuente, se generará una imagen u otra, por lo que los colores y la posición variará si se modifica. (No es necesario mostrar la imagen, solo calcular los valores de los píxeles).



CONCLUSIONES

¿SABES QUÉ ES LO QUE DICE JACK BARTON EN UNA SITUACIÓN COMO ESTA?

CONCLUSIONES

- Si usas console.log / document.write / alert: pueden estar redefinidas
- Si usas proxys: pueden ser relativamente detectados
- Si usas DevTools: utilizar siempre en ventanas diferentes
- Si desofuscas: atento a comprobaciones de integridad
- Si emulas: utiliza un backend moderno y añade una capa propia que imite al navegador (DOM...).

MOAR INFO

- https://x-c3ll.github.io/posts/javascript-antidebugging/

"That's all Folks!"