HackIM 2014 Write-Up

Author: Nick Clark

Trivia 1

This esoteric programming language uses AWSUM THX and O NOES as error handling.

A simple google search leads us to LOLCODE as the answer.

Trivia 2

What software is used to hack into ENCOM's computer system?

Googling ENCOM leads us to the fictional company within the movie Tron. Going to Tron's Wiki (http://en.wikipedia.org/wiki/Tron) we see that the main actor, "Jeff Bridges also portrays Clu (Codified Likeness Utility), a hacking program intended to find evidence of Dillinger's theft in the mainframe." CLU is the answer to this challenge.

Trivia 3

Outcome of Stealth Project to make coffee.

Searching for Stealth Project "Coffee" leads us to this page: http://www.sis.pitt.edu/~icucart/fshi/java_introduction.html . In the conclusion of the story we see that the project ultimately became Java. Java is the answer.

Trivia 4

Successor of the DEAD persistent object format

Googling DEAD persistent object format leads us to the Wikipedia article on the Java class file: http://en.wikipedia.org/wiki/Java_class_file . Under Magic Number we see the story behind DEAD and that it was replaced by RMI. RMI is the answer.

Trivia 5

Oheebhtuf O6700 "havavgvnyvmrq" zrzbel (48-ovg)

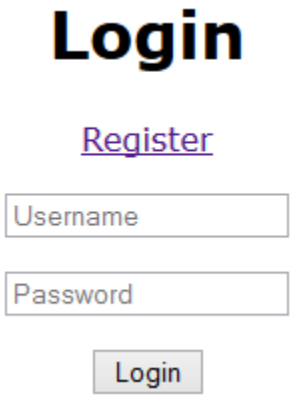
My first guess was to google the whole thing, but that led to no results. I then googled havavgvnyvmrq which lead to the Caeser Cipher word for uninitialized (http://easyciphers.com/uninitialized). This whole phrase has been ciphered. To decipher it I used http://web.forret.com/tools/rot13.asp and just pasted in the cipher text. Which results in:

BURROUGHS B6700 "UNINITIALIZED" MEMORY (48-BIT)

Googling this phrase gives us the answer: 0xBADBADBADBAD.

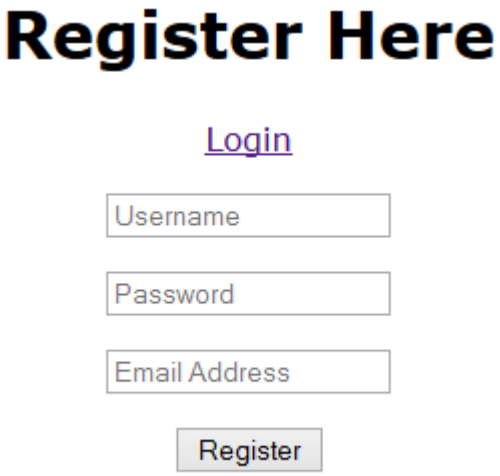
Web 1

index.php



Upon visiting this first page we see a username and password field, a login button, and a link to a register page. Viewing the source yields no hints and pressing login tells us to Please fill both username and password.

Let's see what's on the registration page.



This registration page is very similar in style to the login page. Creating an account and login in with it tells us that we must be an admin to login. So let's try and make ourselves admin.

Viewing the page source of the registration page we see that there is a large amount of new lines followed by commented code:

<!--<user>isAdmin=No</user> -->

This code looks awfully familiar to XML. Now I know that perhaps this website is using some sort of XML based Database which tells me that I should be trying XML injection.

OWASP has a very nice XML injection tutorial on their wiki: https://www.owasp.org/index.php/Testing_for_XML_Injection_%28OWASP-DV-008%29

It is known that the commented code will not validate for XML, however I guessed that <isAdmin> is its own tag within the <user> tag. Knowing this we can try to inject <isAdmin>Yes</isAdmin> into the XML DB. The tutorial on the wiki has various ways to do so. In order to understand the format of the tags I had to look at the source code and see that the fields are named username, password, and email.

In the email input was where I decided to inject my tag. To do the injection I followed the guide and put in a normal username and password. For the email input I put in:

s4tan@hell.com</email><isAdmin>Yes</isAdmin><email>s4tan@hell.com

This code successfully created an account. Let's try logging in…

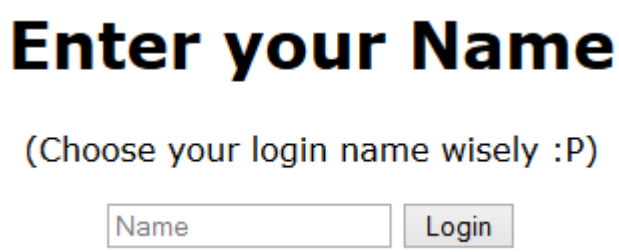
Upon logging in we are greeted with the admin.php page (requires being logged in as an admin to access without being redirected to index.php).

Welcome Admin!!
Here is your Flag: 138a3eb60f0c4787abb60f487cfaa39f

Web 2

login.php

On this webpage we are greeted with the following on login.php:

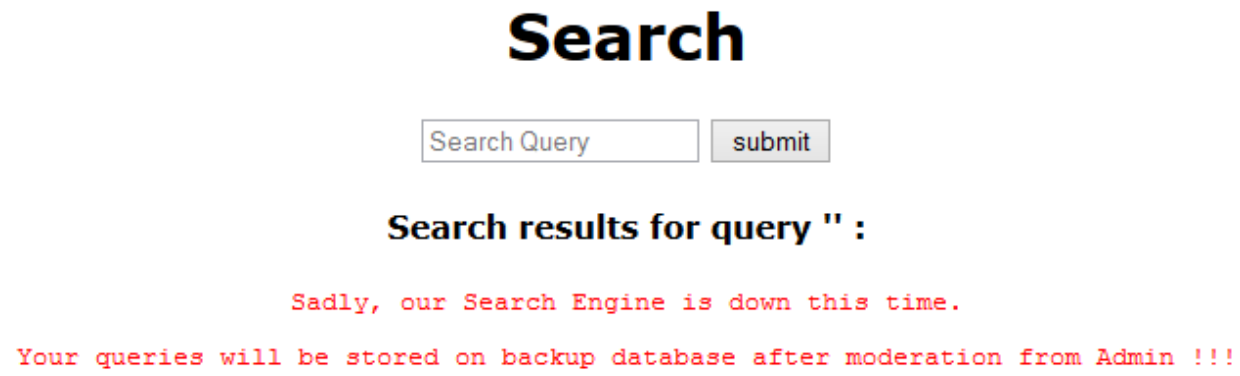


When viewing the source code of this page we see a comment at the bottom after some newlines which states:

<!-- Admin name is the flag -->

With this information it is safe to say that we need to somehow find out the admin's login name.

Entering any name or none at all and clicking login allows us to go to the next page: search.php. It is also good to note that using LiveHeaders in Firefox or even Firebug, we can see that the login.php page sends a POST which then stores a cookie in our browser with the attribute "uName" followed by the value we entered (so long as uName is not null).



On this search.php page we are given the option to submit a query. Further testing shows that anything we submit will result in the red text being displayed and the query that we submitted displaying in the quotes. This is also all done using a GET. SQL Injections was assumed at first, but after testing for SQL Injections it simply posted the code onto the page. Injecting with PHP commands will not work either. HTML and JavaScript are the only two things that will work.

Since JavaScript works we know that we have to do some sort of XSS Injection. Since we also know that the login name is stored in the cookies and the Admin will be checking our queries we can try to steal the Admin's cookies using this code:

<script> alert(document.cookie);</script>

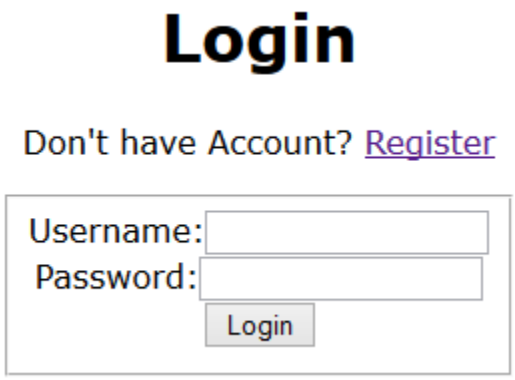
Upon entering this code the search.php page gives us another hint: Hijack admin ... but not with this payload!!!. In fact, the key phrase is "document.cookie." So some other form of XSS is needed. Encoding in base64 allows us to pass the XSS injection code through. This works, but it appears that the

admin may not be vulnerable to this kind of XSS, even when sending the cookie information to another website.

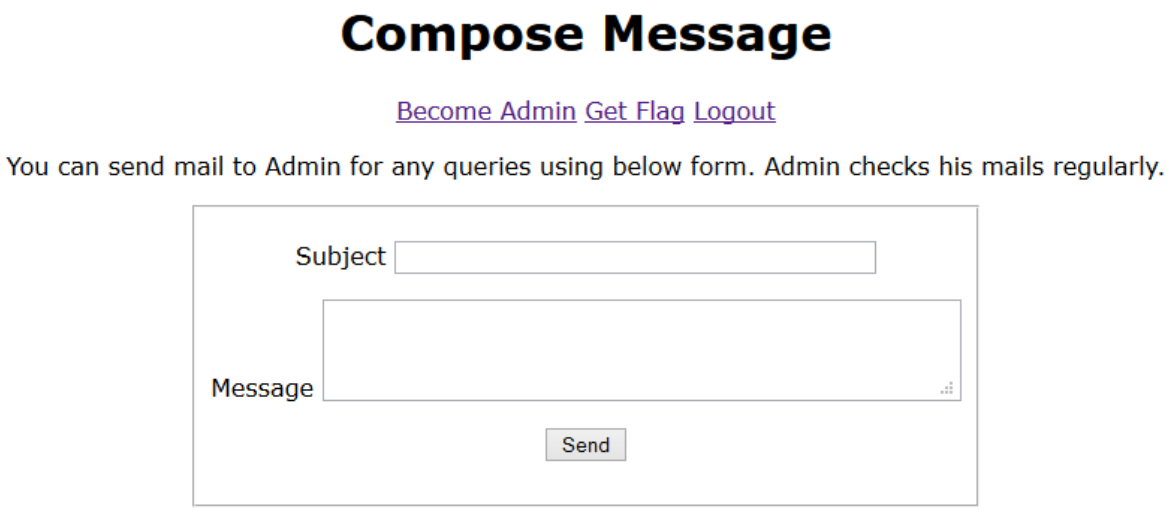
Web 4

index.php

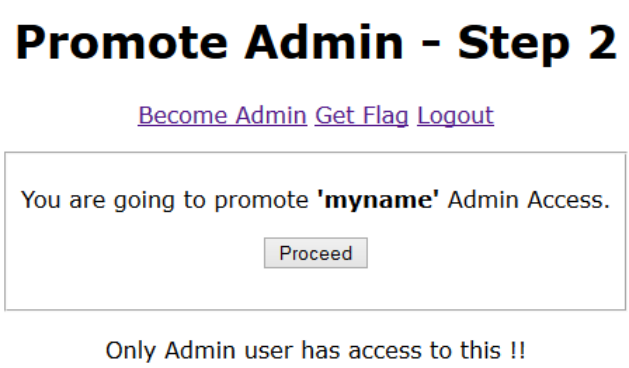
Immediately upon viewing this page we are greeted with: Access Denied: Log-in to access this page.



It asks us for a username and password, but we don't have one so we need to register first.



Once we have registered we can compose a message to admin, click become admin, or get flag. Get flag takes us to a page and tells us that we need to be admin in order to get the flag. Become admin takes us to step1.php where we are asked to enter a user's name to make them admin. When submitting on this page we see that a get action is made with the user value set to what we entered and Set=set. The next page (step2.php) requires us to click submit in order to proceed.



With this information I assumed that we would need to send some javascript to the Admin in two steps. Step one needs to setup the admin's step2.php with our username and Set=set. The second injection is to cause the admin to submit the step2.php.

I personally could not figure out how to get the admin passed step2.php I saw that it was expecting POST, but apparently GET still worked. SaxX was kind enough to give me his script he used after the competition was over:

<script>var req = new XMLHttpRequest(); req.open("GET", "http://54.237.107.251/web4/step1.php?user=myusername&Set=Set", true); req.open("GET","http://54.237.107.251/web4/step2.php?user=myusername &Set=Set", true);</script>

Web 5

index.php

On this page we are greeted with two words: Hack me. Nothing else appears on the page and the url appears to be set at index.php?page=home which tells us that perhaps there is some sort of URL Injection that needs to be done. I try pages such as flag and index.php to see if I can load their data, but I only got File not found. So I try to view the source code of index.php. I found out that, similar to web 100, the source of the web page has a comment all the way at the bottom after some newlines which states:

<!-- etc/flag -->

This tells us the location of the flag. I try to put index.php?page=etc/flag in the url, but this results in a page which states that Attack Detected!!!. Trying index.php?page=etc/ or index.php?page=etc results in a blank page. I try url encoding the / but that doesn't seem to work either. In another attempt I try adding a second /: index.php?page=etc//flag. This leads to the page saying:

Congrats .. Flag for this level - 2f0f7c516d268843341b3d2577ca744a

Misc 1

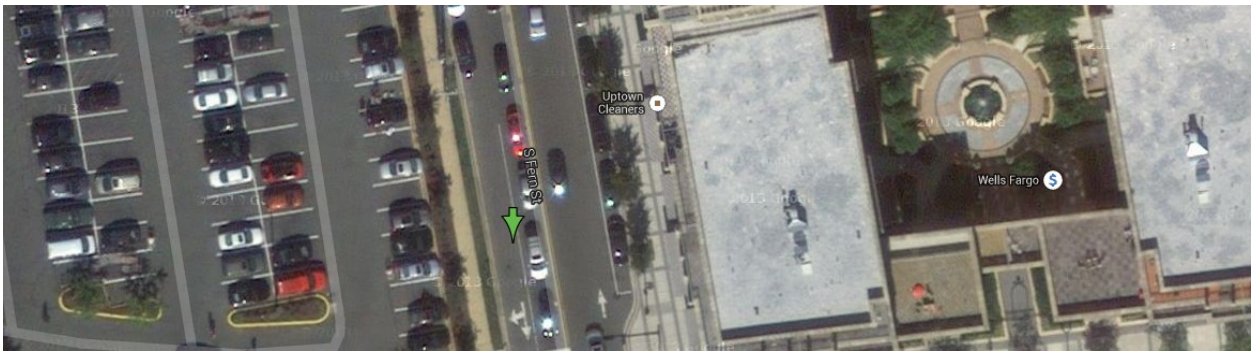
Sam has parked his car in front of a store. Find the name of the store. Level 1.pcap

For this challenge we are given a large pcap file. Initially I chose to open the pcap file in wireshark and browse the data. I figured I would first try and find the router and maybe get its global IP. I found out that information, but this was just misdirection. Further research into the pcap file lead me to find a PNG file that was loaded on a webpage, nullCTF.png.

At this point I knew this was a huge hint. I then rebuilt the PNG from the packets using Network Miner (http://sourceforge.net/projects/networkminer/). This made mining the pcap file so much easier. I then opened the image to see if maybe there were any visible answers.

All the png showed was NULLCON GOA 2014. My next plan was to look for hidden data. I browsed the HEX but none of the information was anything out of the ordinary. There were no original images so steganography would not be the answer. An Exif viewer would be the next best thing.

Heading over to Jeffrey's Exif viewer (http://regex.info/exif.cgi) and uploading the file shows us the a google maps view of where the image was taken. I then proceeded to guess the answer using the nearby businesses. Uptown Cleaners was the closest, but this was not the answer. The answer was next door to them, Wells Fargo.



Misc 2

Tracked sam in level 1? Cool. But now you more details. His USB drive gave you a swf file. Now, think like level 1, but more like a pro, and find his email id.

Level2.swf

For this challenge we are lead to a flash file that continuously loops. My first impressions were to look for anything hidden on the page (usually there is an xml file or something). There wasn't, however I made sure I downloaded the .swf as I figured there could be some hidden data within the file.

Using SWFWire Inspector I was able to decompile the flash file and it was even nice to show me the raw data structure. I saw various attributes and their bytes. I browsed through the SWFHeader attributes and FileAttributesTag but did not find anything. In the MetadataTag there was something:

"metadata": "X: %68%74%74%70%3a%2f%2f%62%69%74%2e%6c%79%2f%31%61%4c%49%59%76%57"

My first assumption was that it was URL encoded. So, I head on over to http://www.asciitohex.com/ and decode it. It decodes to the following link: http://bit.ly/1aLIYvW . Going to this link we see it is a dropbox file available for download: ctfnullcon.apk. The file was created shortly before the CTF.

Using the Android's adb to install the apk to an android device results in failure due to the apk not being signed. Signing the app still causes it to fail. Using a tool called apktool (http://forum.xda-developers.com/showthread.php?t=1989533) I was able to decompile the apk and browse all its contents. Viewing the contents in Eclipse shows us that it is an app called Hacme Bank (bitcoin banking) developed by a group called Foundstone (McAfee).

Using the following regular expression for finding email strings: "[A-Z0-9._%+-]+@" I was able to find all the emails within the .apk when I searched using Eclipse. research@foundstone.com was the first one I had found just by examining the layouts, but a regular expression search finds the second address: sam0908nlu771@gmail.com. This address was located within the Loginservice.smali file and is the answer. I did end up getting the app to compile and run after fixing its errors, but it was never meant to be compiled and run.

Misc 3

What you hear is not what you always hear. Channel your energy in finding the unknown.

Level3.mp3

With this challenge we are given an audio file. I immediately opened it up in audacity as I assumed there was some funny business going on in the audio. It appeared that the left channel was some sort of song (backwards) and the right channel was morse code. Decoding the morse code yielded gibberish, however, the song being backwards is a clue. Using Audacity I reversed the track. The song then comes in clear and the morse code becomes decipherable:

.. .-.. --- ...- . --. --- .- -.-.-- -. ..- .-.. .-.. -.-. --- -. ..--- ----- .---- ....- .. ... .- .-- . ... --- -- . -.-.-- -.-.--

Putting this morse code into a translater (http://rumkin.com/tools/cipher/morse.php) translates this into: "ILOVEGOA!NULLCON2014ISAWESOME!!" which happens to be the flag.

Forensics 1

For all the forensics challenges we are given a raw image disk. Following the guide at http://cyfor.isis.poly.edu/16-view_modules.html?view=181-raw_disk_image_to_virtual_machine we create a Virtual Box image out of the raw image.

VBoxManage convertfromraw d:\Desktop\ctf.dd.001 D:\Desktop\ctf.vmdk

The Client complained that whenever he boots up the machine, all files in his document folder automatically gets deleted. Can you identify the culprit executable process doing this?

If files are being deleted on Windows 7 and we want to track this, we need to turn on the Audit object access policy in Local Security Policy.

Computer Configuration --> Windows Settings --> Security Settings --> Local Policies --> Audit Policy --> Audit object Access

Once that is in place, go to the folder you want to monitor, right click and go to properties

Click the security tab --> Advanced --> Auditing Tab --> Edit --> Add --> then add the group that has access to that folder --> Select the events you want to audit and click OK --> Select Replace all existing inheritable audit entries, to apply the audit on all sub folders and files and click OK

Once this is done we look for Event ID 4660 in the Event Viewer. We see that cmd.exe is deleting the file, but this most likely means that something is being run within Command Prompt.

Now that we know Command Prompt is causing it lets find out if we can see what it's actually doing. To do so we will force cmdfiles and batfiles to stay open. We make the following registry edits:

HKCR\cmdfile\shell\open\command and change the Default key value with **cmd.exe /k "%1" %***

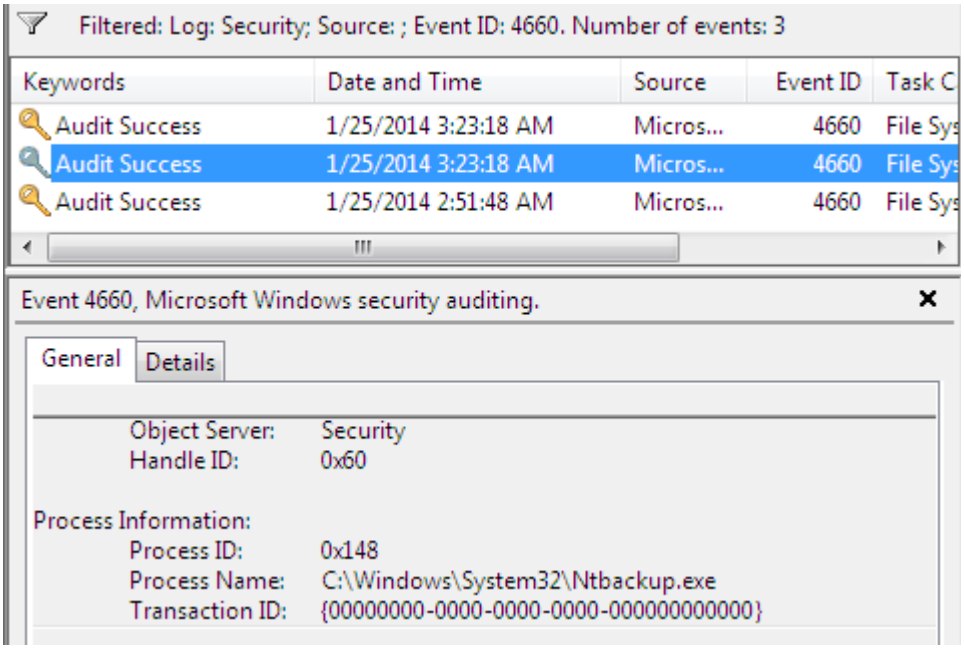
HKCR\batfile\shell\open\command, and change the Default key value with **cmd.exe /k "%1" %***

Once these Registry Edits are done we can now logout and log back in or reboot to see who the culprit is.

Upon doing so we find out that it is a bat file, but more interestingly is that this batfile does not exist if we try and open it! Even more interesting is that the batfile is randomly generated on each login and deleted moments after. So something else is creating the bat file and running the batfile. Opening the file tells us that it is created in C:\Users\boonlia\AppData\Local\Temp\ztmp\. First off, in order to access this area you need to tell windows you want to see Hidden Folders/Files in the File options. After that is done you can see everything there. Let's see what is there.

Browsing this directory yields no results, however there are a lot of interesting setup files that were involved in the making of this vmware and even a Null_final1.pdf. Lets add the ztmp to the audit list to see who is modifying this directory.

Upon adding the folder to the audit list we see that Ntbackup.exe is causing the problem.



NTBackup.exe is the answer.