Pwn2Win CTF 2016

Dragon Sector write-ups

Access Code (net 100p) - solved and written by keidii

`TASK: nc access.pwn2win.party 8111`

Connect to port, make pcap (`tcpdump -nn -s0 -w traffic.pcap host access.pwn2win.party`). Look into pcap -> see incoming connections on a lot of tcp ports. Listen on all incoming traffic and look for any logic.

Code: https://github.com/k3idii/ctf/tree/master/2016/03-pwn2win

Death Sequence (ppc-m, 100p) - solved and written by tahti

Writeup at:
https://github.com/tahti/ctf/tree/master/2016/27_03_Pwn2Win/ppc_100_Death_Sequence

Dump (fore, 50p) - solved and written by mak

Stare a little bit at dump, guess it's from /dev/input - write parser, read the flag.

Sleeper cell (re, 50p) - solved and written by mak

Check output of As and Bs, figure out it's some addition base encoding, extract offsets, turns out they are prime numbers, apply to string set in __constructor, get the flag.

Free Web Access (web 70) - solved and written by mak

Guess it's about X-Forwarded-For header, try different injections, fuzz it a little, get django crash with <>, find out it's XML injection, create following payload:

```
</ip><admin>true</admin><ip>
```

this will log us into admin panel which contains the flag.

String Criptography (crypto, 40p) - solved and written by tahti

Writeup at:
https://github.com/tahti/ctf/tree/master/2016/27_03_Pwn2Win/crypto_40_String_Criptography

How to win (ppc-m 80) - solved and written up by adami

```
def solve(heaps):
    xor = 0
    for i in range(len(heaps)):
        xor = xor ^ heaps[i]
    if xor == 0:
        return "Player 1, you have already lost."
    minvalue = 1000000000000000000
    pos = -1
    for i in range(len(heaps)):
      left = xor ^ heaps[i]
      if left > heaps[i]:
         continue
      value = heaps[i] - left
      if value < minvalue:
         minvalue = value
         pos = i
    return "Player 1, take off " + str(minvalue) + " candies from
pile " + str(pos+1) + "."
```

Tokens (pyexp 50) - solved and written up by tkd

There was a weakness in how the string passed into ‘eval’ was sanitized: none of [a-z_’().] were stripped. Therefore inputting the following string as a command to the server immediately resulted in a shell:

```
gen __import__('os').system('bash')
```

Secure Chat (web 100) - solved and written up by valis

In this challenge we were provided with a page very similar to a login form of the horde app, however description said that there's a secure chat webapp under that.
We also learned from the description that we have to target user 'carleetos' and that he's using Debian Etch on his laptop. Our goal was to intercept messages between him and another user.

Next step was to access /~carletoos and find authorized_keys file there.
As we expected, this public key was generated using flawed predictable openssl and corresponding private key could be easily found in one of publicly available key collections.

After logging in to carletoos ssh account we didn't have access to secure chat .php files, but we did have write access to a directory that we could place our own .php scripts.
By accessing them through http we gained code execution with www-data user privileges.

We analyzed source code of the chat application and it became clear that all chats were encrypted client-side using javascript code - we couldn't decrypt them using our server access.

We also knew from examining 'ps' output that two phantomjs instances are launched every few minutes to simulate two chat user interacting with the system.

After running 'find' command from our web shell we found that one of the application files - jquery.js was writeable from www-data and so we were able to modify it to intercept chat messages in unencrypted form and send them to us.

We've added following JS code to the end of jquery.js:
https://gist.github.com/e612f8f3921148a5c7570ba4ee2c9952

It wasn't easy to get our backdoor code executed since many teams tried to exploit at the same time, but we finally succeeded and got both sides of chat conversation as GET args at our server.
One of the messages contained phone number that was the flag.

Attack Step (700) - solved by q3k/Redford/valis and written up by valis

We don’t have enough time right now to do a full write-up of this challenge, but here’s an exploit for the kernel part:

https://gist.github.com/e45b83b57e37c0132101edca04a7e4e6