

Very Basic Linux Exploits

Level 0: Simply Login using the command “ssh bandit.labs.overthewire.org -p 2220 -l bandit0” and enter the password “bandit0”

Level 1: “cat readme” to get the password

boJ9jbbUNNfktd78OOpsqOltutMc3MY1

Level 2: “cat ./-“ to see the file “-“ in the home directory

CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9

Level 3: since there are spaces in the filename, we use “cat ‘spaces in this filename’” to view the password

UmHadQclWmgdLOKQ3YNgjWxGoRmb5luK

Level 4: since the password is in a hidden file, uses “ls -al” to show all the files in the directory and use “cat .hidden” to get the password

plwrPrtPN36QITSp3EQaw936yaFoFgAB

Level 5: in “inhere” directory, there are 9 files and only 1 of them is human-readable. And we use “file ./*” and find out the only “-file07” is ASCII text and is human-readable, so we “cat” the file and get the password

koReBOKuIDDepwhWk7jZC0RTdopnAYKh

Level 6: in “inhere” directory, we know that the file is 1033 bytes. So we can simply find the file by using “find . -size 1033c” to see which file is the one we are looking for. It turns out that “.file2” in “maybehere07” directory is the one. So we “cat” it and get the password

DXjZPULLxYr17uwoI01bNLQbtFemEgo7

Level 7: I tried using the previous command to find the file, but it shows that lots of files are “Permission denied” and only 1 file is available to see. So we use “cat /var/lib/dpkg/info/bandit7.password” to get the password **HKBPTKQnlay4Fw76bEy8PVxKEDQRKTzs**

Level 8: we are hinted that the password is written next to the word “millionth” in the “data.txt” file, so we need to use “grep” to find the file. Command: “cat data.txt | grep millionth” so we get the password **cvX2JJJa4CFALtqS87jk27qwqGhBM9pIV**

Level 9: we are hinted that the password is the only line of text that occurs only once, so we need to use “sort” command to sort the text inside the “data.txt” file. But there are many repeating statements so we use “uniq” command in addition to “sort” to find the password. Command “cat data.txt | sort | uniq -u” so we get **UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR**

Level 10: it’s said the password is followed by several ‘=’ characters, and if we use the “cat” command now then the screen would be filled with unreadable rubbish. So we use “strings” command to print character sequences that are at least 4 characters long, so the command to get the password is “strings data.txt | grep =” and we get **truKLdjsbJ5g7yyJ2X2R0o3a5HqJFuLk**

Level 11: the file is encrypted in base64, so simply use the command “cat data.txt | base64 - -decode” to decode the file and get the password **IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR**

Level 12: the password is in a file where all lowercase and uppercase letters have been rotated by 13 positions. To convert the text, we use “tr” command and translates characters depending on the parameters

provided, we use n-z and a-m because “tr” won’t continue to translate after the Z. So we use command “cat data.txt | tr a-zA-Z n-za-mN-ZA-M” to get the password **5Te8Y4drgCRfCx8ugdWuEX8KFC6k2EUu**

Level 13: the password file is repeatedly compressed. Now decompress the file and create a directory with read and write permissions, we created one using “mkdir” under “tmp” directory and named “fqy”. We use “xxd” command to make the hexdump of a file and it’s also used to reverse this process. We use command “file data.txt” “xxd -r data.txt data1” to retrieve the file and named it “data1”. And it’s a gzip compressed file, so we use “file data1” “mv data1 data2.gz” “gzip -d data2.gz” to decompress the file. After we checked it, it’s a bzip2 compressed file. Now we are just repeating the commands until we get to the final stage of decompressing the file, after done it multiple times it shows that there is one ASCII text file inside, and we use “cat” to get **8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL**

Level 14: there’s no password for this level, instead, we are given an ssh private key. First use “ls” command to find the private key, now use command “ssh bandit14@localhost -i sshkey.private” to get the the next level. **4wcYUJFw0k0XLShIDzztnTBHixU3b3e**

Level 15: successfully connect as user bandit14 so we can view the password by using “cat /etc/bandit_pass/bandit14”. To get to the next level, we need to submit the password for lvl 14 to port 30000 on localhost. So we use “telnet localhost 30000” to connect to the port and submit the current password, and the password for the next level is retrieved **BfMYroe26WYalil77FoDi9qh59eK5xNr**

Level 16: we are informed that the password for the next level is retrieved by submitting the password of the current level to port 30001 on localhost using SSL encryption. We use the openssl command with parameters like s_client that implements that we are connecting as the client using the hostname localhost at port 30001. We use -ign_eof to inhibit shutting the connection when the end of file is reached in the input. After establishing the connection, we submit the current password and get the password for the next level **cluFn7wTiGryunymYOu4RcffSxQluehd**

Level 17: we are informed that the credentials for the next level can be retrieved by connecting to a port within the range of 31000 to 32000 and submitting the password of bandit16. We use Nmap to scan the ports to get the exact port from the range. As we can see in the output of the Nmap scan that on port 31790 there is a message that hints that we need to enter the password on that port. So we use “openssl s_client -connect localhost:31790” to connect to the port using openssl. Now we enter the password for bandit16 and we get a RSA key, so we create a directory under “tmp” and store the RSA key in a created file. Since SSH won’t allow any private key with open permissions, we have to change permissions. Use command “chmod 600 pavan.private” to apply the permissions equivalent to 600 so we can read and write the file. And we get to the next level **xLYVMN9WE5zQ5vHacb0sZEVqbrp7nBTn**

Level 18: the password is the only line that has been changed between 2 password files, so we use “diff passwords.old passwords.new” to get the required password. We use this password and get to an SSH connection as bandit18. Now on providing with the correct password our connection was closed. This is because the authors of this level have modified the

“.bashrc” file to log us out of ssh. We will use the “-t” parameter to disable the pseudo -tty allocation. As this is making our session vulnerable to get closed. Connect SSH again, and we get a shell. After trying “ls” command it gives use a readme file, use “cat” and we get password

kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd

Level 19: run “ls” and we get a script file, and it runs as another use bandit20.and the password is stored at /etc/bandit_pass/, so we run the script using “./bandit20-do cat /etc/bandit_pass/bandit20” and we get our password **lueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x**

Level 20: After successfully getting the ssh to user bandit19, we start with ls command to see what we got this time. We have a file that seems like a script. We tried to run to see the working of the script. We are shown that the script runs a command as another user. Now we were informed that the password is stored at /etc/bandit_pass/. So, we run the script with the cat command to read the password for the next level. We will use this password to get an SSH connection as bandit20.

GbKksEFF4yrVs6il55v6gwY5aVje5f0j

Level 21: there is a setuid binary in this level whose job is to make a connection to a localhost on a port and read the password used to login as bandit20 and then send the password for the next level. Use “ls” and we see a script “suconnect”. Run “./suconnect” and we know it requires a port to connect too. We will execute to the point where we run “suconnect” without parameters and create other instance of the same shell. Run a “netcat” listener over another instance on the same port we are planning to “suconnect”. But we need to start listener before running the “suconnect”. On running the “suconnect”. Netcat will grab a session.

Now we enter the password that we used to login as user bandit20. As we can see that the password, we entered is read by the “suconnect” and when the password is verified. Password for the next level is sent to the listener. Run “./suconnect 4444” and it reads password and send next password, next run “nc -lvp 4444” to execute the second instance, and we get the password for the next level

gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr

Level 22: there is a cron script running and we need to enumerate /etc/cron.d/ for the password. We traverse to that path and use ls to see the files inside the directory. Read cronjob_bandit22 using “cat” and it shows that there is a script at /usr/bin/cronjob_bandit22.sh. So we use “cat /usr/bin/cronjob_bandit22.sh” and we see there is a file under “tmp”, we read that file using “cat” and get the password

Yk7owGAcWjwMVRwrTesJEwB7WVOiLLI

Level 23: similar to the previous level, we traverse to the path and use “ls” to see the files in the directory. Use “cat cronjob_bandit23” and finds there is a script at /usr/bin/cronjob_bandit23.sh we read the “.sh” using “cat” and it says it has a variable named “myname” which is the output of the command “whoami”. It prints “I am user bandit22” and it is encrypted in MD5. This hash is used to name the file which has the password for the next level. Now to get the password for the bandit23 user, we run the command with the value for the variable “myname” set to bandit23. This will give us the hash value which further gives the name of the file in the tmp directory. Using “echo I am user bandit23 | md5sum | cut -d ‘ ‘ -f 1” to get the password for the next level **jc1udXuA1tiHqjlsL8yaapX5XIAI6i0n**

Level 24: Like the previous levels, we read “cronjob_bandit24.sh” and see there’s a script with a variable named “myname” which consists of the output of the “whoami” command. The script first changes the name directory to /var/spool and then executes files with the variable “myname” file. And after executing it delete all files inside that directory. Now to get the password for the next directory we will have to create a script of our own so that we can put it inside the /var/spool that will cat the password file from the /etc/bandit_pass/bandit24. We will have to save the file with the name of the next user in order to run the file as a cron job successfully. Use “mkdir /tmp/lvl24” to create a directory under “tmp” and write the script that will read the password from the /etc/bandit_pass and writes in the file inside the directory we just created. And in that directory, use “nano bandit24.sh” to create a file write the script that will read the password from the /etc/bandit_pass and writes in the file inside the directory we just created. To execute successfully, we need to change permissions to the script, run “chmod 777 bandit24.sh” “chmodd 777 /tmp/lvl24” to do that. [UoMYTrfrBFHyQXmg6gzctqAwOmw1lohZ](#)

Level 25: to apply Brute force we will have to create a Dictionary. As always, we will be needing to read and write permissions to create a script. So, we will create a directory inside the tmp directory using “mkdir /tmp/qyf”. And we use “nano bruteforcer.sh” to create a brute force script. After creating the script file, we will have to create a file that would act as a dictionary. We are told that we will have to feed the daemon running on port 30002 the password of the current level followed by a 4-digit passcode. So, we ran a loop that lists all the 4 digits and writes those inside a file called output. This file will act as a dictionary.

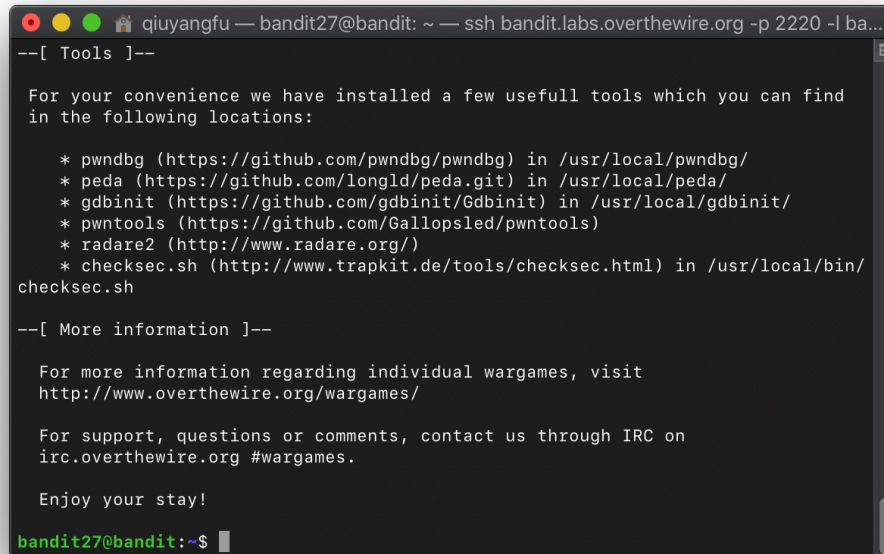
```
“./bruteforcer.sh  
cat output.txt | nc localhost 30002 >> result.txt  
sort result.txt | uniq -u  
ssh bandit25@localhost”
```

Level 26: The shell for user bandit26 is not /bin/bash, but something else. Run “cat /etc/passwd | grep bandit26” “cat /usr/bin/showtext” to check the file. Before we try to SSH into bandit26, shrink the Console down to only few lines and there is a “Show more” displayed on the screen, and run the “ssh” command. Once the command is ran, enter VIM by pressing “V”, then type: “:r /etc/bandit_pass/bandit26” and get the password

5czqV9L3Xx8JPOyRbXh6lQbmIOWvPT6Z



Level 27: Run “ls” and we see a script “bandit27-do”. Run the program by using “./bandit27-do” and it tells us to run it as another user. Run “./bandit27-do whoami” and it says bandit27. So we run the script as user bandit27, and we use “./bandit27-do cat /etc/bandit_pass/bandit27” to get the password **3ba3118a22e93127a4ed485be72ef5ea**

A screenshot of a terminal window titled "qiuyangfu — bandit27@bandit: ~ — ssh bandit.labs.overthewire.org -p 2220 -l ba...". The terminal displays a list of tools installed for convenience, including pwndbg, peda, gdbinit, pwntools, radare2, and checksec.sh, along with their locations. It also provides information on where to find more details about wargames and how to get support. The prompt at the bottom is "bandit27@bandit:~\$".

```
--[ Tools ]--

For your convenience we have installed a few usefull tools which you can find
in the following locations:

* pwndbg (https://github.com/pwndbg/pwndbg) in /usr/local/pwndbg/
* peda (https://github.com/longld/peda.git) in /usr/local/peda/
* gdbinit (https://github.com/gdbinit/Gdbinit) in /usr/local/gdbinit/
* pwntools (https://github.com/Gallopsled/pwntools)
* radare2 (http://www.radare.org/)
* checksec.sh (http://www.trapkit.de/tools/checksec.html) in /usr/local/bin/
checksec.sh

--[ More information ]--

For more information regarding individual wargames, visit
http://www.overthewire.org/wargames/

For support, questions or comments, contact us through IRC on
irc.overthewire.org #wargames.

Enjoy your stay!

bandit27@bandit:~$
```

PROOF

Level 28: The password is in a git repository, so we just clone the repository by running “git clone ssh://bandit27-git@localhost/home/bandit27-git/repo” under a newly created directory. Then we simply go inside the “repo” and view “README” for password **0ef186ac70e04ea33b4c1853d2526fa2**

```

qiuyangfu — bandit27@bandit: /tmp/lv27 — ssh bandit.labs.overthewire.org -p 2...
[bandit27@bandit:~$ mkdir /tmp/lv27
[bandit27@bandit:~$ cd /tmp/lv27
[bandit27@bandit:/tmp/lv27$ git clone ssh://bandit27-git@localhost/home/bandit27-
git/repo
Cloning into 'repo'...
Could not create directory '/home/bandit27/.ssh'.
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:98UL0ZW85496EtCRkKlo20X30PnyPSB5tB5RPbhczc.
[Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit27/.ssh/known_hos
ts).
This is a OverTheWire game server. More information on http://www.overthewire.or
g/wargames

[bandit27-git@localhost's password:
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.
[bandit27@bandit:/tmp/lv27$ ls
repo
[bandit27@bandit:/tmp/lv27$ cat repo/README
The password to the next level is: 0ef186ac70e04ea33b4c1853d2526fa2
bandit27@bandit:/tmp/lv27$

```

PROOF

Level 29: we repeat the previous steps like creating a new directory under /tmp/ and clone the git repository. We open up the “README.md” file and see the password is actually missing, so we get the log of the file

```

qiuyangfu — bandit28@bandit: /tmp/lv28/repo — ssh bandit.labs.overthewire.org...
commit 073c27c130e6ee407e12faad1dd3848a110c4f95
Author: Morla Porla <morla@overthewire.org>
Date: Tue Oct 16 14:00:39 2018 +0200

    fix info leak

diff --git a/README.md b/README.md
index 3f7cee8..5c6457b 100644
--- a/README.md
+++ b/README.md
@@ -4,5 +4,5 @@ Some notes for level29 of bandit.
 ## credentials

- username: bandit29
-- password: 0bc95374b4e061778ee9976372716b7
+- password: xxxxxxxxxx

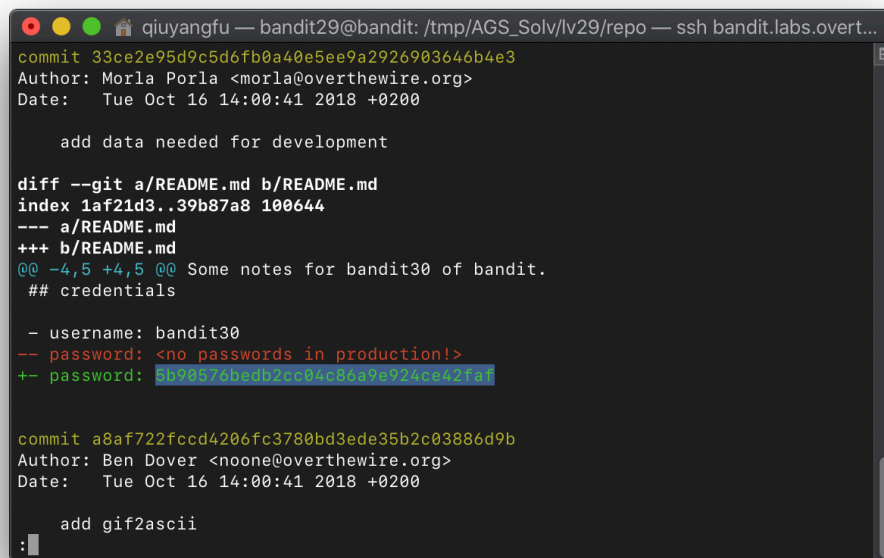
commit 186a1038cc54d1358d42d468cdc8e3cc28a93fcb
Author: Morla Porla <morla@overthewire.org>
Date: Tue Oct 16 14:00:39 2018 +0200

    add missing data

```

by using “git log -p README.md” and get the password
bbc96594b4e001778eee9975372716b2

Level 30: repeat previous steps and we see that we cant find the password by using “git log -p” so we check the branches in this git using “git branch -a” and it shows there’s another branch called “dev”. So we switch to that branch using “git checkout dev” and view the “README.md” file for password **5b90576bedb2cc04c86a9e924ce42faf**



```

commit 33ce2e95d9c5d6fb0a40e5ee9a2926903646b4e3
Author: Morla Porla <morla@overthewire.org>
Date: Tue Oct 16 14:00:41 2018 +0200

    add data needed for development

diff --git a/README.md b/README.md
index 1af21d3..39b87a8 100644
--- a/README.md
+++ b/README.md
@@ -4,5 +4,5 @@ Some notes for bandit30 of bandit.
 ## credentials

- username: bandit30
-- password: <no passwords in production!>
+- password: 5b90576bedb2cc04c86a9e924ce42faf

commit a8af722fccd4206fc3780bd3ede35b2c03886d9b
Author: Ben Dover <noone@overthewire.org>
Date: Tue Oct 16 14:00:41 2018 +0200

    add gif2ascii
  
```

PROOF

Level 31: repeat previous steps and we see the “README.md” file has nothing meaningful, so we enumerate this git. Use “git tag” to find the tags in the repo, and there is a “secret”. Then we use “git show secret” to find the password **47e603bb428404d265f59c42920d81e5**

```

qiyangfu — bandit30@bandit: /tmp/AGS_Solv_lv29/repo — ssh bandit.labs.overt...
[bandit30@bandit:/tmp/AGS_Solv_lv29/repo$ git tag
secret
[bandit30@bandit:/tmp/AGS_Solv_lv29/repo$ git show secret
47e603bb428404d265f59c42920d81e5
bandit30@bandit:/tmp/AGS_Solv_lv29/repo$

```

PROOF

Level 31: repeat previous steps and view “README.md” file, it says we need to push a file to the remote repository with specific file name, content

```

qiyangfu — bandit31@bandit: /tmp/lv31/repo — ssh bandit.labs.overthewire.org...
ts).
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

[bandit31-git@localhost's password:
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 314 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: ### Attempting to validate files... ###
remote:
remote: .o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.
remote:
remote: Well done! Here is the password for the next level:
remote: 56a9bf19c63d650ce78e6ec0354ee45e
remote:
remote: .o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.
remote:
To ssh://localhost/home/bandit31-git/repo
! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'ssh://bandit31-git@localhost/home/bandit31-git/repo'
bandit31@bandit:/tmp/lv31/repo$

```


and branch. Commands used: “nano” “git add -f” “git commit -m” “git push origin”. **56a9bf19c63d650ce78e6ec0354ee45e**

Level 32: when logged into this level, there’s a message says “Welcome to the uppercase shell”. I try to use “ls” but it doesn’t work. The level hint says “its time for another escape”, so I try to use “\$0” and it works now. Then I run “ls -al” and find out the owner of the uppercases shell is “bandit33”. So we access /etc/bandit_pass/bandit33 file and get the

```

qiu yangfu — bandit31@bandit: /tmp/lv31/repo — ssh bandit.labs.overthewire.org...
http://www.overthewire.org/wargames/

For support, questions or comments, contact us through IRC on
irc.overthewire.org #wargames.

Enjoy your stay!

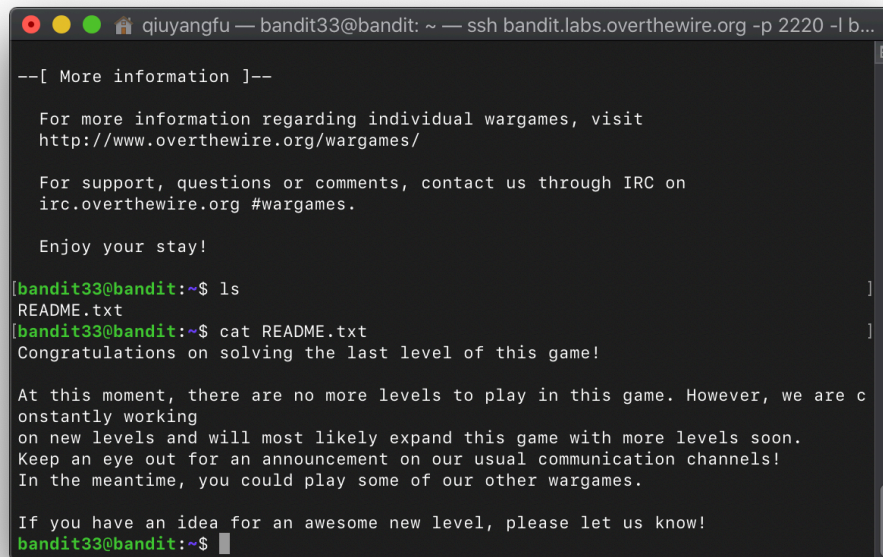
WELCOME TO THE UPPERCASE SHELL
[>> ls
sh: 1: LS: not found
[>> $0
[$ lss -al
sh: 1: lss: not found
[$ ls -al
total 28
drwxr-xr-x  2 root    root    4096 Oct 16  2018 .
drwxr-xr-x 41 root    root    4096 Oct 16  2018 ..
-rw-r--r--  1 root    root     220 May 15  2017 .bash_logout
-rw-r--r--  1 root    root    3526 May 15  2017 .bashrc
-rw-r--r--  1 root    root     675 May 15  2017 .profile
-rwsr-x---  1 bandit33 bandit32 7556 Oct 16  2018 uppershell
[$ cat /etc/bandit_pass/bandit33
c9c3199ddf4121b10cf581a98d51caee
$

```

PROOF

password **c9c3199ddf4121b10cf581a98d51caee**

Level 33: this its the final level and it doesn’t have a password.



```
qiuyangfu — bandit33@bandit: ~ — ssh bandit.labs.overthewire.org -p 2220 -l b...

--[ More information ]--

For more information regarding individual wargames, visit
http://www.overthewire.org/wargames/

For support, questions or comments, contact us through IRC on
irc.overthewire.org #wargames.

Enjoy your stay!

[bandit33@bandit:~$ ls ]
README.txt
[bandit33@bandit:~$ cat README.txt ]
Congratulations on solving the last level of this game!

At this moment, there are no more levels to play in this game. However, we are c
onstantly working
on new levels and will most likely expand this game with more levels soon.
Keep an eye out for an announcement on our usual communication channels!
In the meantime, you could play some of our other wargames.

If you have an idea for an awesome new level, please let us know!
bandit33@bandit:~$
```

PROOF

Time spent on this challenge: approximately 2 days.