Pentesting your Django apps

Keira Paterson & Sky Christensen

Slides: github.com/red-and-black/conferences

We care about security.

Some caveats.

We are not security professionals.

We can’t make you a security professional either.

(Sorry.)

Use security professionals if you can.

The next 20 minutes

1. What is pentesting?
2. Why should you learn?
3. How do you pentest a Django app?
4. How do you learn?

What is pentesting?

A scan for vulnerabilties

A scan for vulnerabilties

Attacking to exploit those vulnerabilities

A scan for vulnerabilties

Attacking to exploit those vulnerabilities

With the purpose of securing the app

A scan for vulnerabilties

Attacking to exploit those vulnerabilities

With the purpose of securing the app

With the permission of the owner

A scan for vulnerabilties

Attacking to exploit those vulnerabilities

With the purpose of securing the app

With the permission of the owner

“

If you fail a penetration test you know you have a very bad problem indeed.

If you pass a penetration test you do not know that you don’t have a very bad problem.

”

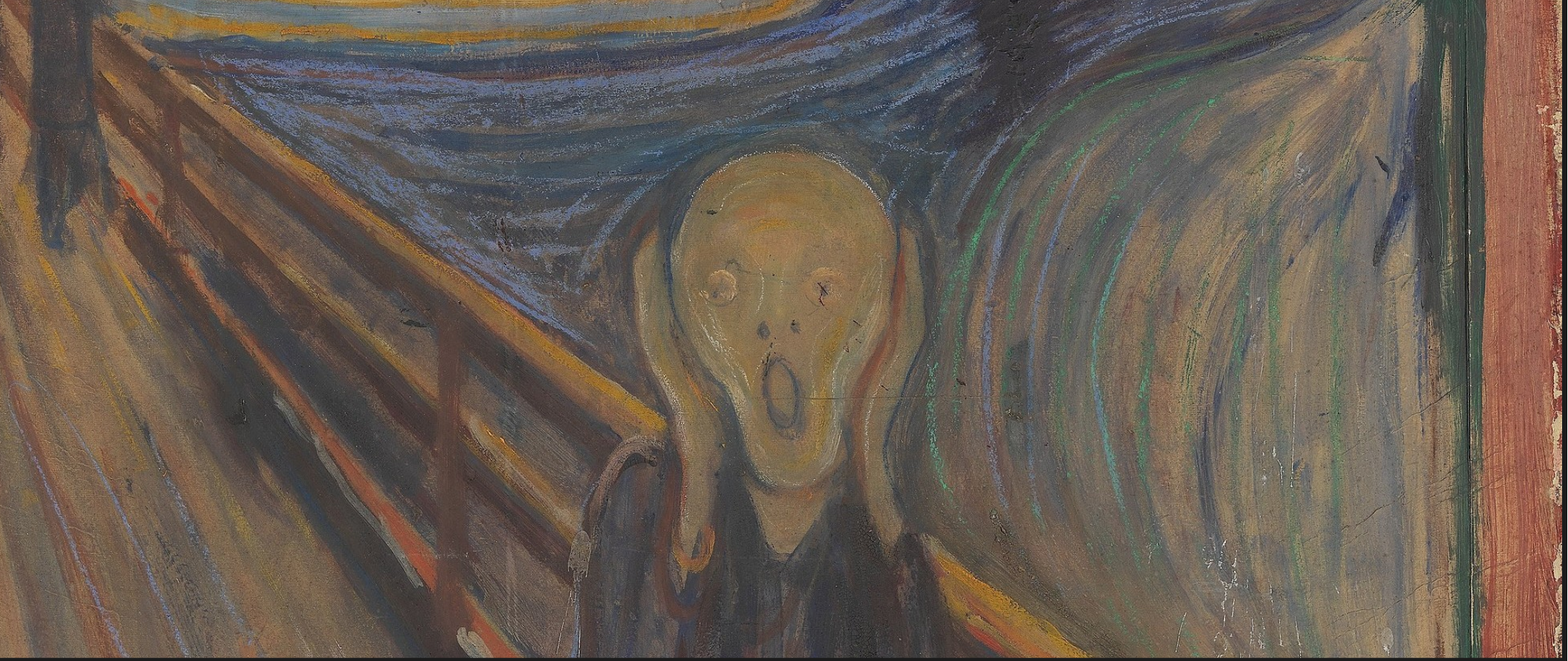
Gary McGraw

Why should I learn to pentest?

KNOWLEDGE

BUDGETS

FUN



FEAR

QUALITY

EMPATHY

RESPONSIBILITY

“

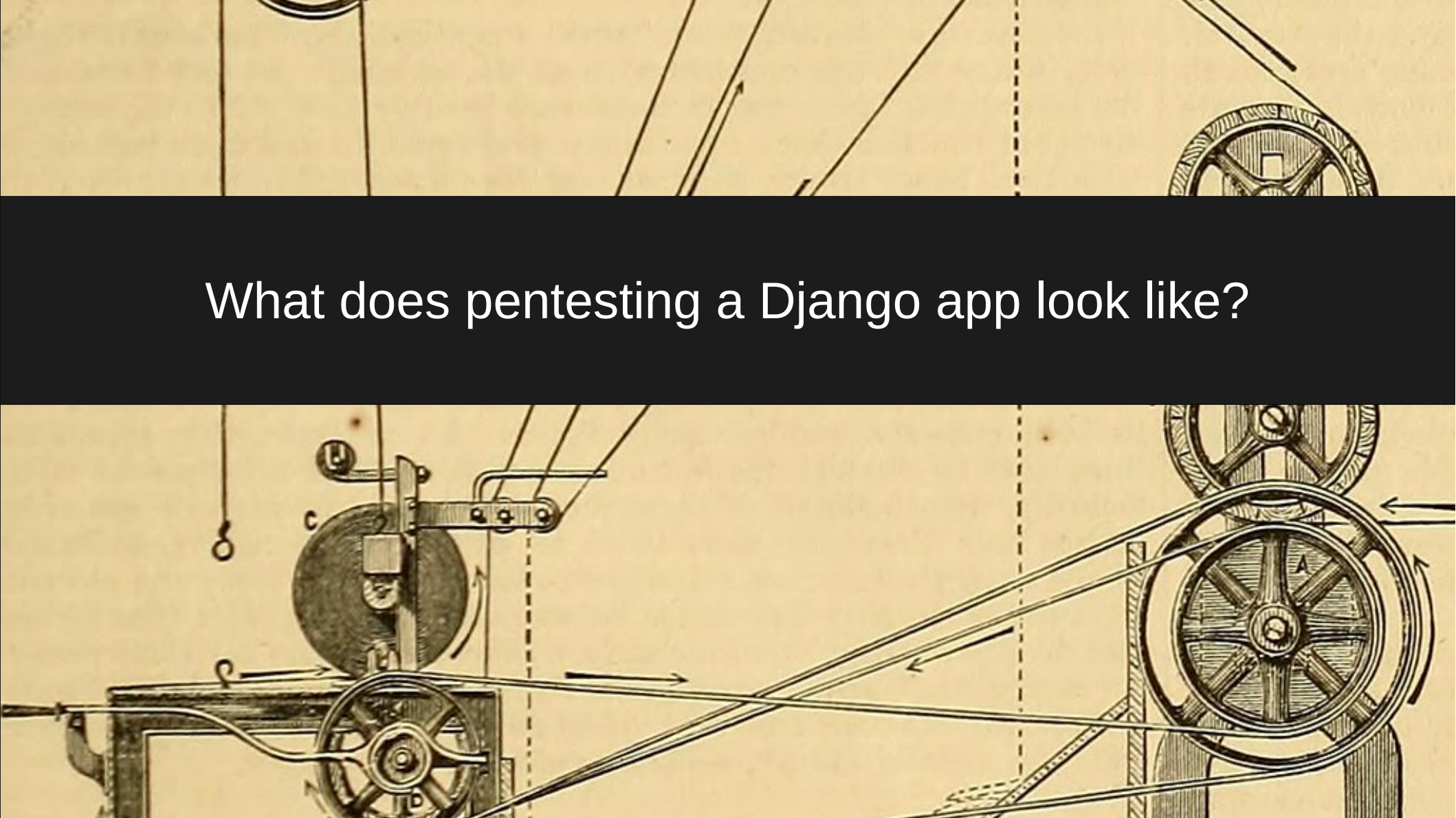
This is bigger than computer security.

Technology now permeates society in a way it didn't just a couple of decades ago...

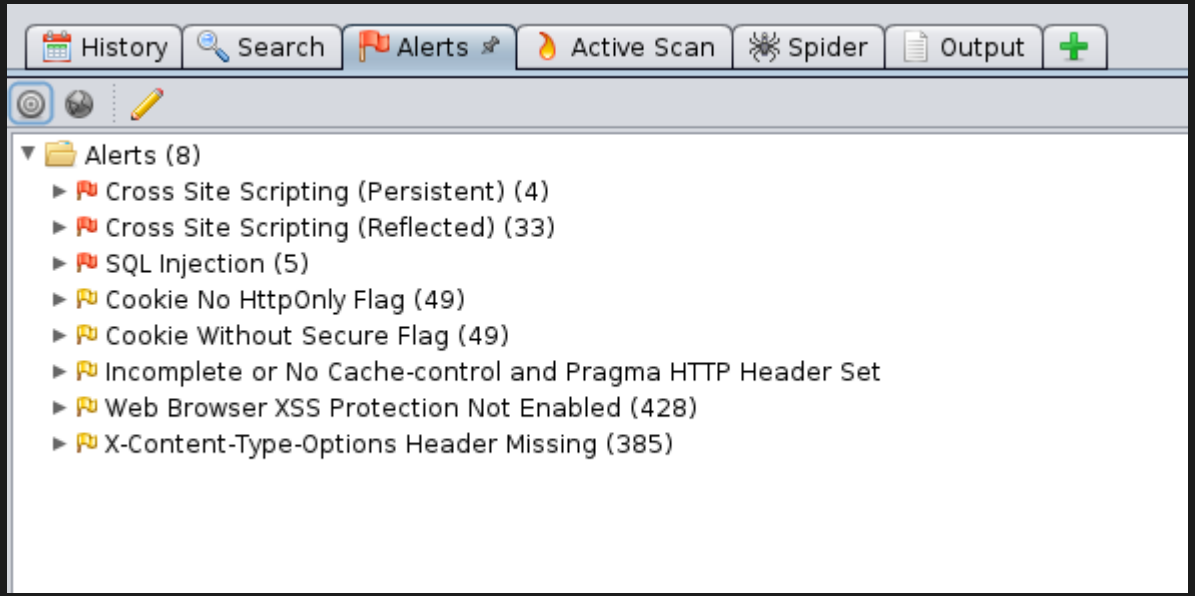
That means technologists now are relevant to all sorts of areas that they had no traditional connection to: climate change, food safety, future of work, public health, bioengineering.

”

Bruce Schneier



What does pentesting a Django app look like?


History
Search
Alerts
Active Scan
Spider
Output
Alerts (8)
Cross Site Scripting (Persistent) (4)
Cross Site Scripting (Reflected) (33)
SQL Injection (5)
Cookie No HttpOnly Flag (49)
Cookie Without Secure Flag (49)
Incomplete or No Cache-control and Pragma HTTP Header Set
Web Browser XSS Protection Not Enabled (428)
X-Content-Type-Options Header Missing (385)

ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	3
Medium	0
Low	5
Informational	0

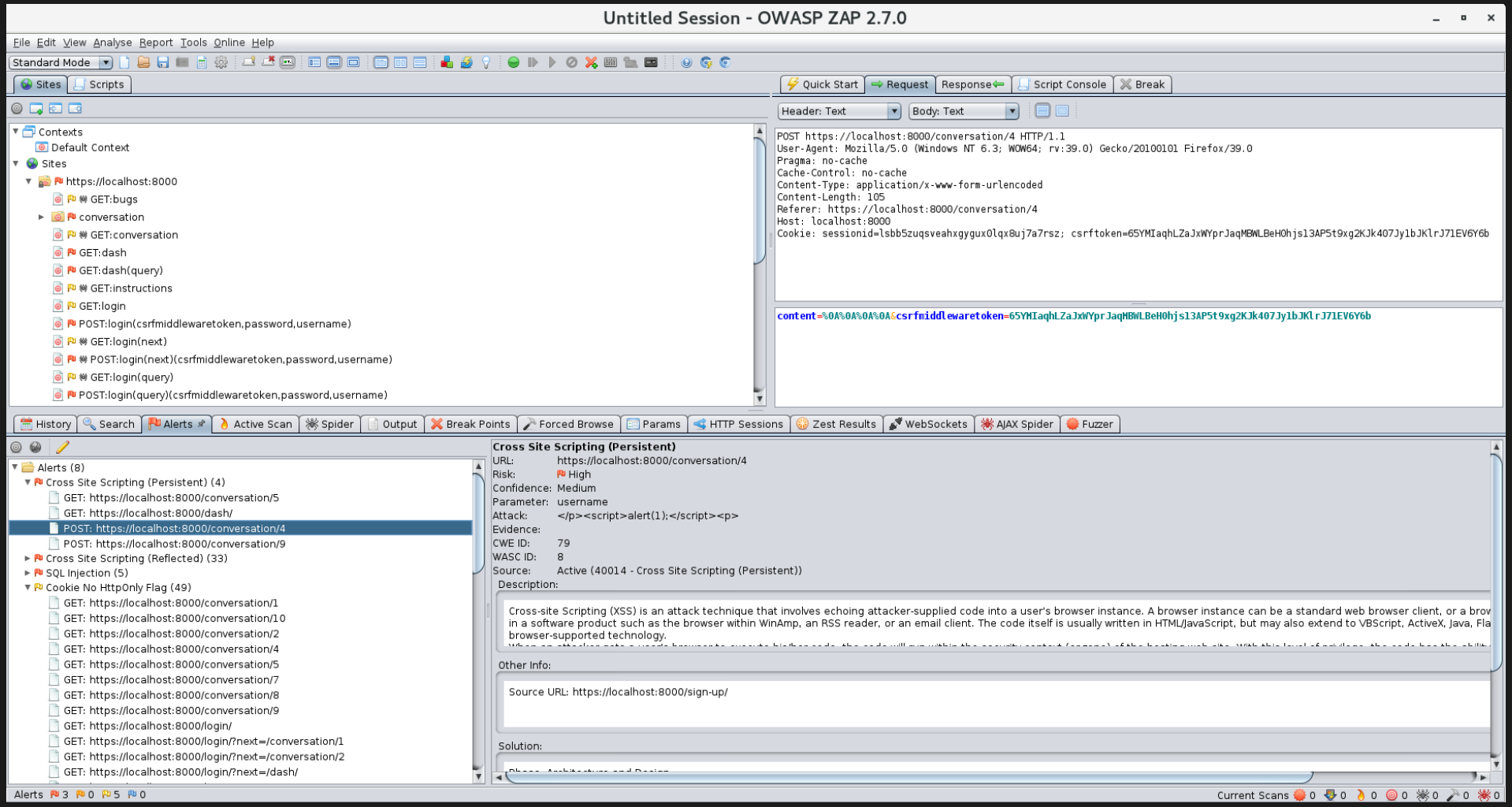
Alert Detail

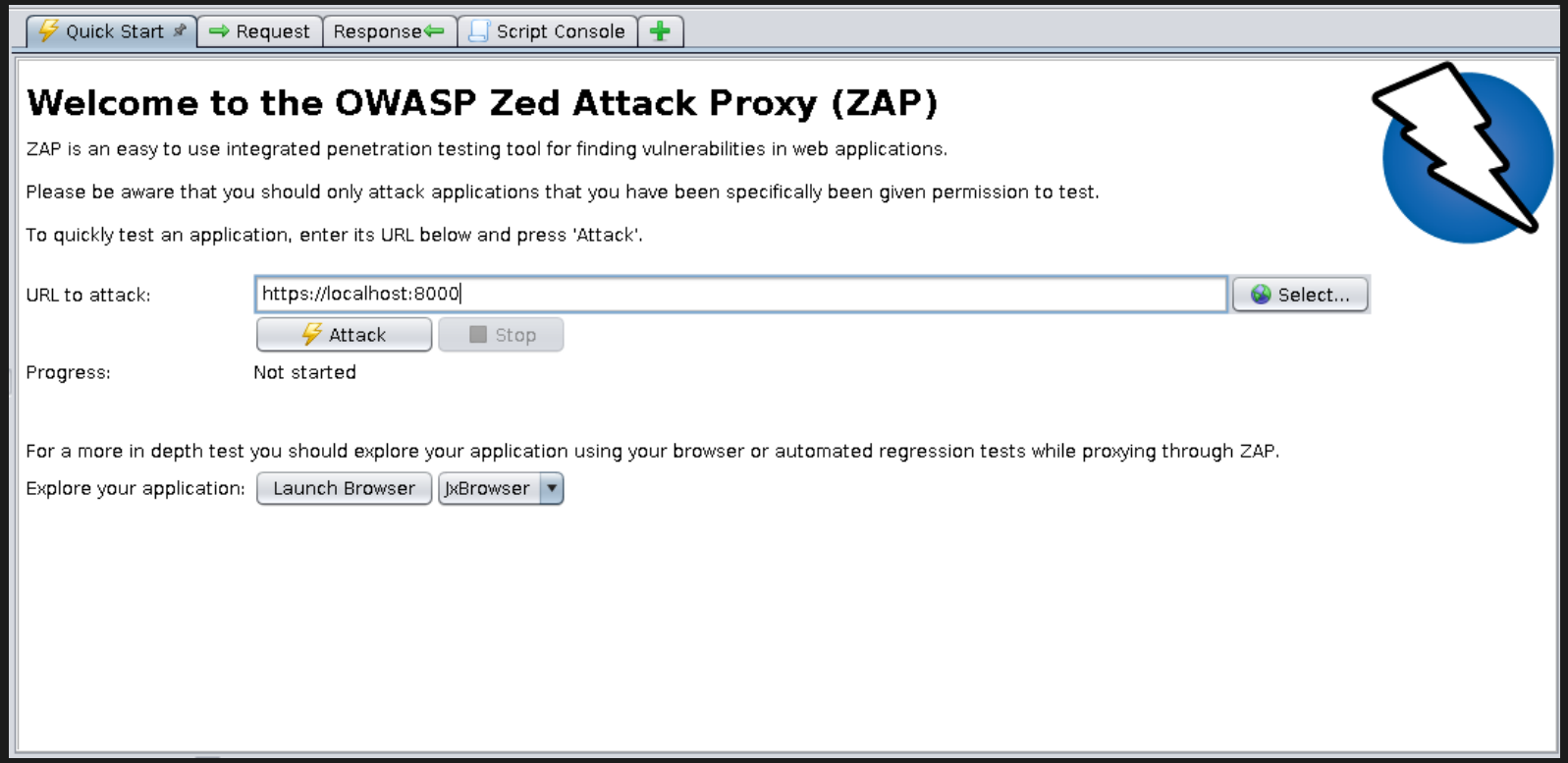
High (Medium)	Cross Site Scripting (Reflected)
	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology.

User – Browser – Server – App

User – Browser – ZAP – Server – App

User – Browser – ZAP – Server – App


Untitled Session - OWASP ZAP 2.7.0
File Edit View Analyse Report Tools Online Help
Standard Mode
Sites
Scripts
Contexts
Default Context
Sites
https://localhost:8000
GET:bugs
conversation
GET:conversation
GET:dash
GET:dash(query)
GET:instructions
GET:login
POST:login(csrfmiddlewaretoken,password,username)
GET:login(next)
POST:login(next)(csrfmiddlewaretoken,password,username)
GET:login(query)
POST:login(query)(csrfmiddlewaretoken,password,username)
Quick Start
Request
Response
Script Console
Break
Header: Text
Body: Text
POST https://localhost:8000/conversation/4 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0
Pragma: no-cache
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
Content-Length: 105
Referer: https://localhost:8000/conversation/4
Host: localhost:8000
Cookie: sessionid=lsbb5zuqsveahxgygux0lqx8uj7a7rsz; csrftoken=65YMIaqhLZaJxWYprJaqMBWLBeH0hjs13AP5t9xg2KJk407Jy1bJKlrJ71EV6Y6b
content=%0A%0A%0A%0A&csrfmiddlewaretoken=65YMIaqhLZaJxWYprJaqMBWLBeH0hjs13AP5t9xg2KJk407Jy1bJKlrJ71EV6Y6b
History
Search
Alerts
Active Scan
Spider
Output
Break Points
Forced Browse
Params
HTTP Sessions
Zest Results
WebSockets
AJAX Spider
Fuzzer
Alerts (8)
Cross Site Scripting (Persistent) (4)
GET: https://localhost:8000/conversation/5
GET: https://localhost:8000/dash/
POST: https://localhost:8000/conversation/4
POST: https://localhost:8000/conversation/9
Cross Site Scripting (Reflected) (33)
SQL Injection (5)
Cookie No HttpOnly Flag (49)
GET: https://localhost:8000/conversation/1
GET: https://localhost:8000/conversation/10
GET: https://localhost:8000/conversation/2
GET: https://localhost:8000/conversation/4
GET: https://localhost:8000/conversation/5
GET: https://localhost:8000/conversation/7
GET: https://localhost:8000/conversation/8
GET: https://localhost:8000/conversation/9
GET: https://localhost:8000/login/
GET: https://localhost:8000/login/?next=/conversation/1
GET: https://localhost:8000/login/?next=/conversation/2
GET: https://localhost:8000/login/?next=/dash/
Cross Site Scripting (Persistent)
URL: https://localhost:8000/conversation/4
Risk: High
Confidence: Medium
Parameter: username
Attack: <script>alert(1);</script>
Evidence:
CWE ID: 79
WASC ID: 8
Source: Active (40014 - Cross Site Scripting (Persistent))
Description:
Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a brov
in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Fla
browser-supported technology.
Other Info:
Source URL: https://localhost:8000/sign-up/
Solution:
Alerts 3 0 5 0
Current Scans 0 0 0 0 0 0 0


Quick Start
Request
Response
Script Console
Welcome to the OWASP Zed Attack Proxy (ZAP)
ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.
Please be aware that you should only attack applications that you have been specifically been given permission to test.
To quickly test an application, enter its URL below and press 'Attack'.
URL to attack:
https://localhost:8000
Select...
Attack
Stop
Progress:
Not started
For a more in depth test you should explore your application using your browser or automated regression tests while proxying through ZAP.
Explore your application:
Launch Browser
JxBrowser

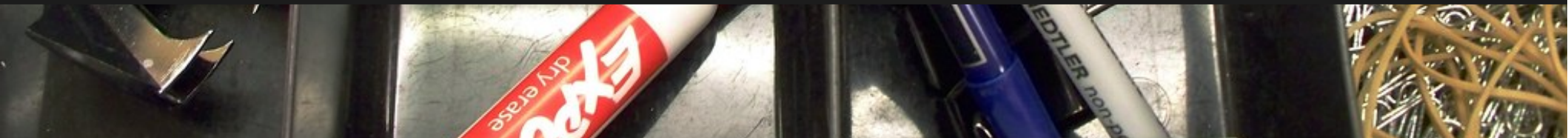
Behave – Selenium – Firefox – Gunicorn

Behave – Selenium – Firefox

Gunicorn

Behave – Selenium – Firefox – ZAP – Gunicorn

String and duct tape



```python
# environment.py

def before_all(context):
    start_zap()
    use_fixture(start_firefox, context)
```

Behave – Selenium – Firefox – ZAP – Gunicorn

```python
# environment.py

def start_zap():
    subprocess.Popen(['zaproxy', '-deamon', '-config', 'api.disablekey=true'])
    sleep(10)
```

Behave – Selenium – Firefox – ZAP – Gunicorn

```python
# environment.py

def start_firefox(context):
    options = Options()
    options.headless = True
    desired_capabilities = webdriver.DesiredCapabilities.FIREFOX
    desired_capabilities['proxy'] = {
        # Set proxy config here...
    }
    context.browser = webdriver.Firefox(
        executable_path='geckodriver',
        options=options,
        capabilities=desired_capabilities,
    )
```

Behave – Selenium – Firefox – ZAP – Gunicorn

```python
# environment.py

def after_all(context):
    # Set up.
    zap = ZAPv2(apikey=None)
    base_url = 'https://localhost:8000'
    # Spider.
    zap.spider.scan(base_url)
    # Active scan.
    zap.ascan.scan(base_url)
    # Report.
    with open('report.html', 'w') as f:
        f.write(zap.core.htmlreport())
```

Behave – Selenium – Firefox – ZAP – Gunicorn

```
openssl req -x509 -nodes -newkey rsa:2048 -keyout server.key -out server.crt

gunicorn --certfile=server.crt --keyfile=server.key djangogoat.wsgi
```

Behave – Selenium – Firefox – ZAP – Gunicorn

Custom scripts

DjangoAuthentication.js

CSRFInterceptor.js

Don’t reinvent the wheel.

Where do I start to learn this?

OWASP resources

Top 10

OWASP resources

Top 10

ZAP + video tutorials

OWASP resources

Top 10

ZAP + video tutorials

Testing Guide v.4

Try out DjangoGoat

Home Code Bugs Instructions Log In Sign Up

DjangoGoat

IS NOT A REAL APP

DjangoGoat is an intentionally vulnerable django app, to help django developers learn how to pentest and secure their apps. If you are seeing it online, please don't put any real data in it.

This is a voluntary, not-for-profit effort and we welcome help! If you would like to contribute code, content or documentation to this project, send us a pull request at GitHub

This app uses Django 2.1.7. There are other vulnerable testing apps for earlier versions of django, run by other teams. See DjanGoat and django nV

Come to our workshop!

You are good enough.

You can do it!

Pentesting. You can do it!

@RedAndBlackTech | github.com/red-and-black