



Welcome to

6. Web Application Security: Offensive and Defensive Security in Web Development Elective, KEA

Henrik Kramselund he/him han/ham hkj@zecurity.com @kramse  

Slides are available as PDF, kramse@Github
6-web-app-hacking-offensive-security-in-web.tex in the repo security-courses

Goals for today



Todays goals:

- Web Application Hacking – moving from offensive towards defensive
- Talk about a few tools for testing web applications
- Start doing defensive – using Nginx as example TLS endpoint, load balancer and filter

Photo by Thomas Galler on Unsplash

Plan for today



Subjects

- Offense against web application, finish up Denial of Service
- The list of offensive methods from the Web Application Security book

Exercises

- Try using a few web testing tools – performance / stress testing
- Nginx as TLS endpoint
- Nginx as Load Balancer
- Nginx for logging and filtering

Time schedule



- 1) Denial of Service and stress testing 45min
- 2) Defensive begin: Veracode document 45min
- 3) Logging and filtering with Nginx as example 45min
- 4) Exercises and workshop 45min

Exercises are also mixed in during 1-3

Reading Summary



Web Application Security, Andrew Hoffman, 2020, ISBN: 9781492053118

Part II. Offense, chapters 9-16, very short chapters

14. Denial of Service (DoS)

15. Exploiting Third-Party Dependencies

16. Part II Summary

What we learnt from the offensive part



Part II Summary

Ultimately, thoroughly testing a web application will require knowledge of common vulnerability archetypes, critical thinking skills, and domain knowledge so that deep logic vulnerabilities outside of the most common archetypes can be found. The foundational skills presented in Part I and Part II should be sufficient to get you up and running on any web application security pen-testing project you take part in in the future.

From this point forward, you should pay attention to the business model in any application you test. All applications are at risk of vulnerabilities like XSS, CSRF, or XXE, but only by gaining a deep understanding of the underlying business model and business logic in an application can you identify more advanced and specific vulnerabilities.

Source: *Web Application Security*, Andrew Hoffman, 2020, ISBN: 9781492053118

- Depending on data stored in your application, it may or may not be critical if there is a XSS vulnerability

14. Denial of Service (DoS)



Perhaps one of the most popular types of attacks, and the most widely publicized, is the distributed denial of service (DDoS) attack. This attack is a form of denial of service (DoS), in which a large network of devices flood a server with requests, slowing down the server or rendering it unusable for legitimate users.

DoS attacks come in many forms, from the well-known distributed version that involves thousands or more coordinated devices, to code-level DoS that affects a single user as a result of a faulty regex implementation, resulting in long times to validate a string of text. DoS attacks also range in seriousness from reducing an active server, to a functionless electric bill, to causing a user's web page to load slightly slower than usual or pausing their video midbuffer.

Because of this, it is very difficult to test for DoS attacks (in particular, the less severe ones). Most bug bounty programs outright ban DoS submissions to prevent bounty hunters from interfering with regular application usage.

- There has also been some hash-based attacks and attacks using data to create bad performance, see PHP security advisories

Riorey Taxonomy of DDoS Attacks



What are DDoS? and DoS?

Denial of Service attack - prevents authorized users access to resources

Can be a single request to HTTP service, sequence of network packets

Distributed Denial of Service attack - many (spoofed) sources

https://en.wikipedia.org/wiki/Denial-of-service_attack



RioRey Taxonomy of DDoS Attacks

Attack Types		Attack Matrix Dimensions									
		Nature of IP	Handshake	Source IP Range	Packet Rate	Packet Size	Packet Content	Fragmenting	Session Rate	Session Duration	VERB Rate
TCP BASED	1 SYN Flood	Spoofed	None	Large	High	Small	---	---	---	---	---
	2 SYN-ACK Flood	Spoofed	None	Large	High	---	---	---	---	---	---
	3 ACK & PUSH ACK Flood	Spoofed	None	Large	High	---	---	---	---	---	---
	4 Fragmented ACK	Spoofed	None	Large	Moderate	Large	---	High	---	---	---
	5 RST or FIN Flood	Spoofed	None	Large	High	---	---	---	---	---	---
	6 Synonymous IP	Spoofed	None	Single IP	High	---	---	---	---	---	---
	7 Fake Session	Spoofed	None	Large	Low	---	---	---	---	---	---
	8 Session Attack	Non-Spoofed	Yes	Small	Low	---	---	---	Low	Long	---
	9 Misused Application	Non-Spoofed	Yes	Small	Variable	---	---	---	High	Short	---



TCP HTTP BASED	10 HTTP Fragmentation	Non-Spoofed	Yes	Small	Very Low	Small	Valid	High	Very Low	Very Long	Very Low
	11 Excessive VERB	Non-Spoofed	Yes	Small	High	---	Valid	---	High	Short	High
	12 Excessive VERB Single Session	Non-Spoofed	Yes	Small	Low	---	Valid	---	Low	Moderate	High
	13 Multiple VERB Single Request	Non-Spoofed	Yes	Small	Very Low	Large	Valid	---	Low	Long	High
	14 Recursive GET	Non-Spoofed	Yes	Small	Low	---	Valid	---	Low	Short	Low
	15 Random Recursive GET	Non-Spoofed	Yes	Small	Low	---	Valid	---	Low	Short	Low
	16 Faulty Application	Non-Spoofed	Yes	Small	Low	---	Valid	---	Low	Short	Low

NOT a complete list, but examples, see another example

[https://en.wikipedia.org/wiki/Slowloris_\(computer_security\)](https://en.wikipedia.org/wiki/Slowloris_(computer_security))



U D P B A S E D	17 UDP Flood	Spoofed	---	Very Large	Very High	Small	Not Valid	---	---	---	---
	18 Fragmentation	Spoofed	---	Moderate	Very High	Large	Not Valid	High	---	---	---
	19 DNS Flood	Spoofed	---	Very Large	Very High	Small	Valid	---	---	---	---
	20 VoIP Flood	Spoofed	---	Very Large	Very High	Small	Valid	---	---	---	---
	21 Media Data Flood	Spoofed	---	Very Large	Very High	Moderate	Valid	---	---	---	---
	22 Non-Spoofed UDP Flood	Non-Spoofed	---	Small	Very High	---	Valid	---	---	---	---

UDP is not used in most Web applications. My advice is to bandwidth limit UDP flow into parts of the network with HTTP/HTTPS servers.

ICMP



ICMP BASED	23 ICMP Flood	Spoofed	---	Very Large	Very High	Variable	Not Valid	---	---	---	---
	24 Fragmentation	Spoofed	---	Moderate	Very High	Large	Not Valid	High	---	---	---
	25 Ping Flood	Spoofed	---	Very Large	Very High	Small	Valid	---	---	---	---

ICMP is a control protocol for sending messages about a problem. It does not carry data for web applications, so could be restricted and bandwidth limited in most network.

If you have a 10Gigabit connection, having 1Gbit UDP and 100Mbit ICMP is often enough.



Stress testing: Apache benchmark and others

```
$ ab -n 100 https://www.kramse.org/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

```
Benchmarking www.kramse.org (be patient).....done
```

```
...
```

Multiple stress testing tools available

Apache Benchmark is old, but easy to run

Apache Benchmark output - 1



Server Software: nginx
Server Hostname: www.kramse.org
Server Port: 443
SSL/TLS Protocol: TLSv1.2,ECDHE-RSA-AES256-GCM-SHA384,4096,256
Server Temp Key: ECDH P-384 384 bits
TLS Server Name: www.kramse.org

Document Path: /
Document Length: 5954 bytes

Concurrency Level: 1
Time taken for tests: 9.596 seconds
Complete requests: 100
Failed requests: 0
Total transferred: 651100 bytes
HTML transferred: 595400 bytes
Requests per second: 10.42 [#/sec] (mean)
Time per request: 95.962 [ms] (mean)
Time per request: 95.962 [ms] (mean, across all concurrent requests)
Transfer rate: 66.26 [Kbytes/sec] received

Apache Benchmark output - 3



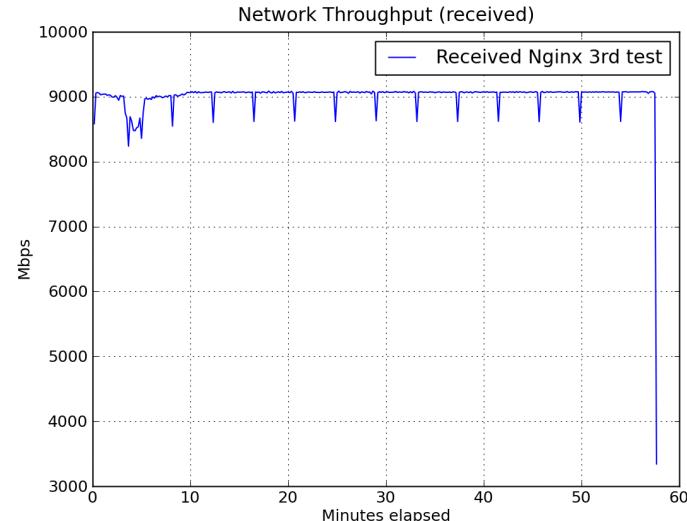
Connection Times (ms)

	min	mean [+/-sd]	median	max
Connect:	48	71 105.5	61	1114
Processing:	6	25 157.8	9	1587
Waiting:	5	9 1.9	9	16
Total:	55	96 189.0	70	1649

Percentage of the requests served within a certain time (ms)

50%	70
66%	72
75%	73
80%	74
90%	76
95%	78
98%	1124
99%	1649
100%	1649 (longest request)

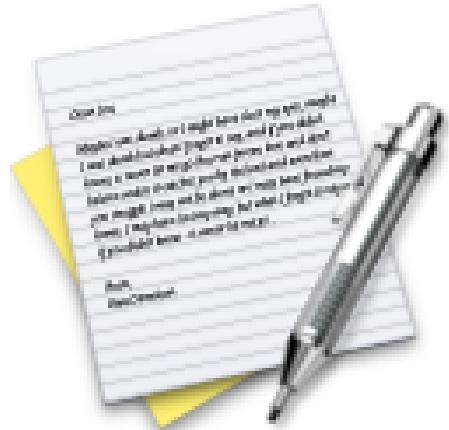
More application testing



Tsung can be used to stress HTTP, WebDAV, SOAP, PostgreSQL, MySQL, LDAP and Jabber/XMPP servers <http://tsung.erlang-projects.org/>

Some are targetted towards JSON and REST API testing

Exercise

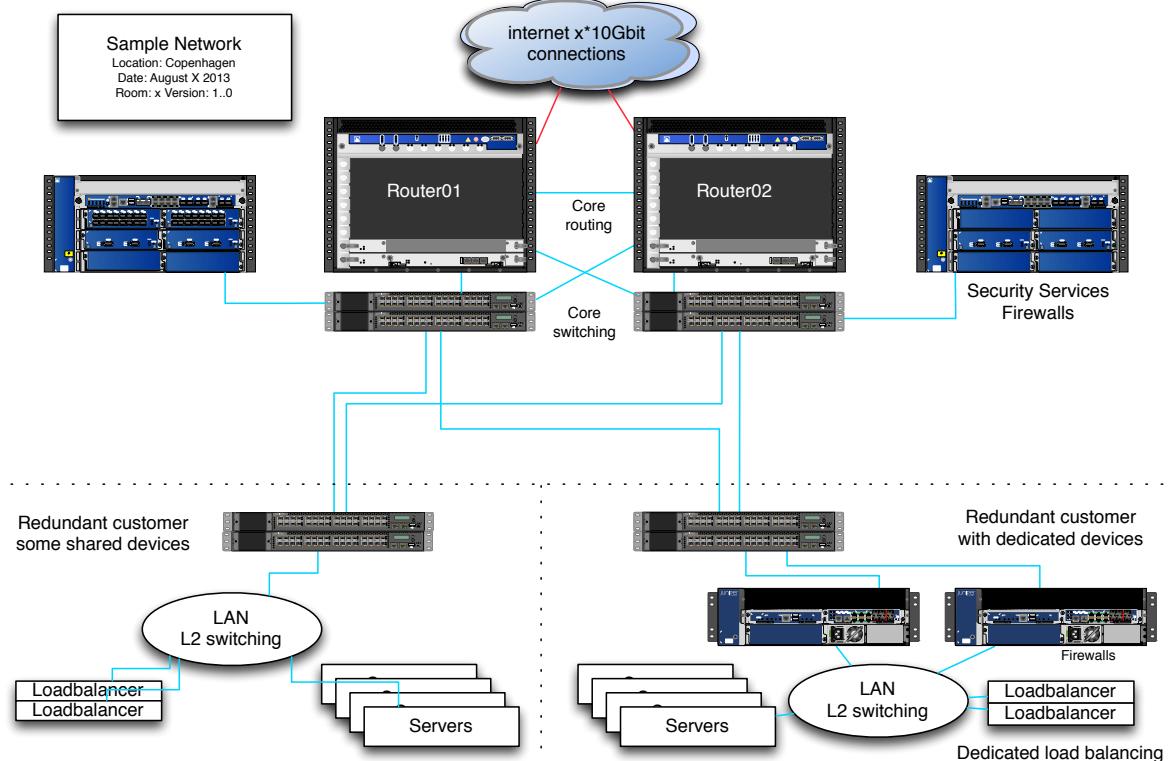


Now lets do the exercise

Apache Benchmark 20 min

which is number **30** in the exercise PDF.

Networks today





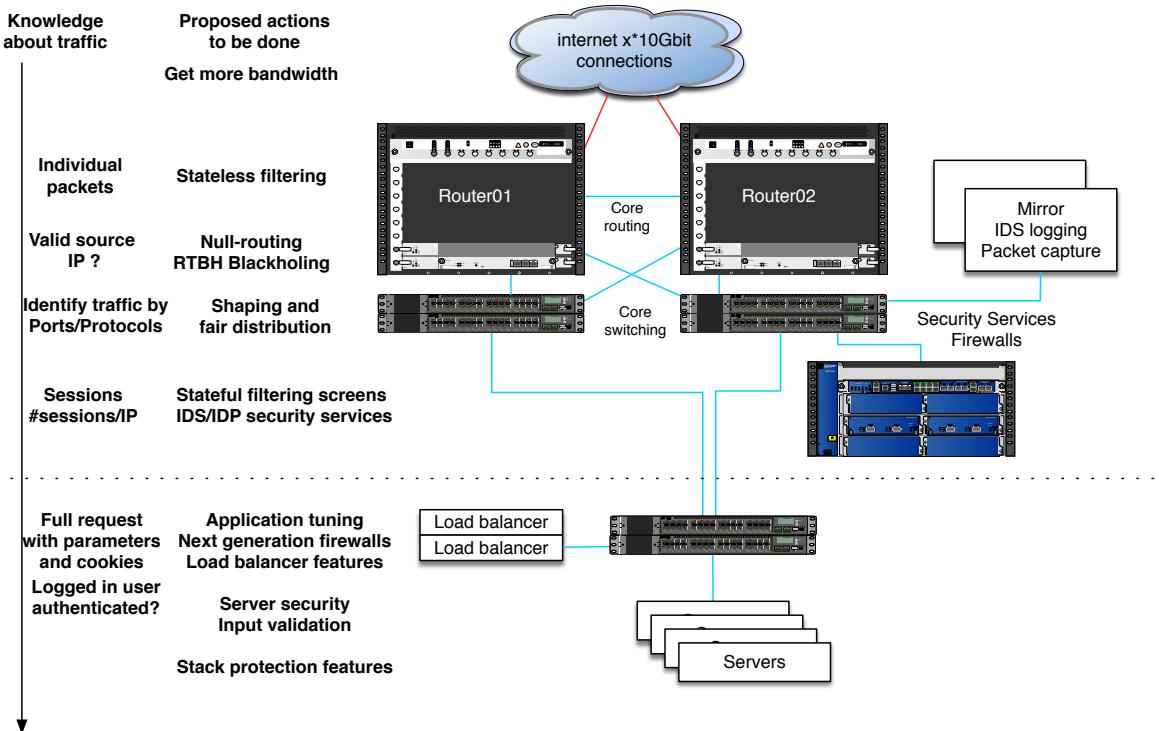
Cisco router DoS exploit script

```
#!/bin/sh
# 2003-07-21 pdonahue cisco-44020.sh
# -- this shell script is just a wrapper for hping (http://www.hping.org)
#   with the parameters necessary to fill the input queue on
# exploitable IOS device
# -- refer to "Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packets"
# (http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml)
...
for protocol in $PROT
do
    $HPING $HOST --rawip $ADDR --ttl $TTL --ipproto $protocol
    --count $NUMB --interval u250 --data $SIZE --file /dev/urandom
done
```

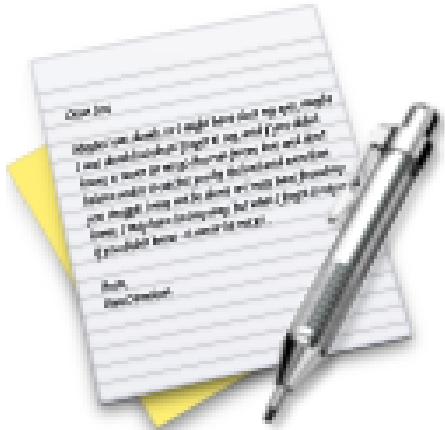
Produced a sequence of packets, that required the router to be powered OFF and ON again to return to normal

I have multiple presentations about network DDoS testing, see my repository
<https://github.com/kramse/security-courses/search?q=ddos>

Defense in depth - multiple layers of security



Exercise



Now lets do the exercise

Bonus:TCP SYN flooding 30min

which is number **31** in the exercise PDF.

Veracode slideshow: A short example



Imagine this:

You are working as a web developer. You recognize there are security issues in web applications. How would you proceed?

You might stumble upon these resources:

- *The book Web Application Security* by Andrew Hoffman
- The OWASP organisation and web sites, like:
https://cheatsheetseries.owasp.org/cheatsheets/PHP_Configuration_Cheat_Sheet.html
- Misc resources, like *Secure Coding Best Practices Handbook* from Veracode

We will now look into this resource, and then return to look at Nginx for Defense

Nginx for defense



Nginx (pronounced "engine X"[8] / ˌnd ən ks/ EN-jin-EKS), stylized as NGI X, is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. The software was created by Igor Sysoev and publicly released in 2004.[9] Nginx is free and open-source software, released under the terms of the 2-clause BSD license. A large fraction of web servers use NGINX,[10] often as a load balancer.[11]

<https://en.wikipedia.org/wiki/Nginx>

Nginx is a popular web server, but can also be used for other things. These are enabled when used as a Proxy, reverse proxy in front of services.

Nginx Use-Cases

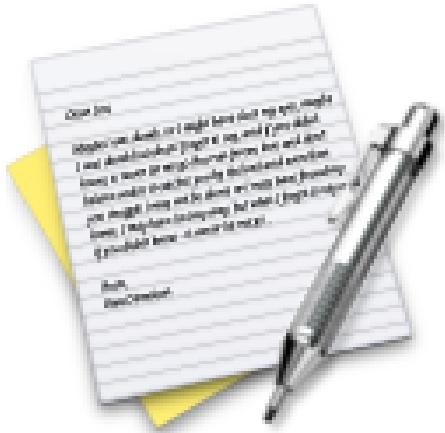


We will now look at a few use-cases, and discuss

- Nginx as a Transport Layer Security (TLS) endpoint – collect ALL your TLS related configuration in one place
- Nginx as a Load Balancer
- Nginx logging – look into what is logged normal access and error logging – run Nikto to produce bad logs
- Nginx filtering – prevent people from accessing a path like /administration

Feel free to do the exercises in any order, depending on your own interests.

Exercise



Now lets do the exercise

Nginx as a Transport Layer Security (TLS) endpoint 20 min

which is number **32** in the exercise PDF.

Exercise



Now lets do the exercise

Run Nginx as a load balancer

which is number **33** in the exercise PDF.

Exercise



Now lets do the exercise

Nginx logging 20 min

which is number **34** in the exercise PDF.

Exercise



Now lets do the exercise

Nginx filtering 40 min

which is number **35** in the exercise PDF.

For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools