WebWatcher

A Lightweight Tool for Analyzing Web Server Logs

Hervé DEBAR
IBM Zurich Research Laboratory
Global Security Analysis Laboratory
de@zurich.ibm.com
PROJECT GOALS

• To automatically analyze web server logs
• To detect compromise attempts through HTTP requests
• To have a very small impact in terms of resources
• To monitor HTTP servers on as many platforms as possible
• To operate both in real time and batch modes
• To use our knowledge of malicious HTTP requests signatures
• To have a flexible and rich attack signature format
• To track hosts exhibiting malicious behavior
• To discover and learn new attack signatures
• To remove false alarms intelligently
ATTACKS TARGETED

• Penetration of the system via HTTP server vulnerabilities
  – Vulnerable CGI program requests
  – Password guessing
  – Access to sensitive information (guessing CGI names, accessing system files)

• Denial-of-service attacks
  – Repeated accesses to non-existing resources
  – Repeated accesses to resources that cause server errors

• Legal but undesirable activity
  – Borderline use of the HTTP protocol
    • e.g. % encoding of normal characters
  – Sensitive documents accesses

• Policy violation (when used on firewall HTTP proxy)
  – External / internal policies governing access to web sites.
TECHNICAL CHOICES

• Input: CLF/ECLF format
  
  host - authenticated_user [date] "request string" status bytes

• Implementation language: Perl5
  – Portable
  – Well accepted in web server environments
  – Regular expression matching

• Signature language: perl regular expressions
  – Easy to create simple signatures
  – Possible to create very complex ones to reduce false alarms

• Pipelined architecture
  – Each filter corresponds to a set of verifications
ARCHITECTURE
MODULES

• Parser
  – Reads the request
  – Breaks the log entry into its constituent parts and check integrity
  – Refines the URL into its parts and check the format / empty string
  – Decodes any encoded characters and verify appropriateness of % characters

• Pattern
  – Looks for signatures
  – Signatures are relevant to fields
  – Signatures are grouped into classes
  – Negative matching

• Combination
  – Logical combination of signatures (if sig1 and sig2 then sig3)

• Refined
  – Signature dependencies (if sig1 then match sig2)
MODULES (2)

• Suspicious
  – Keeps track of suspicious hosts effectively (not signatures !)

• Trusted
  – Eliminates alerts based on signatures

• Decision
  – Ages and updates the tree of suspicious hosts

• Print module
  – Prints out the alert
    • Syslog
    • Internal format

• HTML Reporting facility
  – Overview of the results
  – Intended for batch processing
EXPERIMENTS

• Data collected from (batch runs)
  – 2 medium sized commercial sites
  – University logs
  – 1 day of the Nagano Olympics website (Courtesy of Jim Challenger)

• Data collected from an apache web server (Real time)
  – RS/6000 250 running apache 1.3.3

• Initial signature base
  – 50 vulnerable cgi programs (now 150)
  – Directory tricks
  – Interpreters in cgi-bin
  – Sensitive files
  – ...
TRAFFIC ANALYSIS

Since the malicious traffic is far smaller than the normal traffic, all days have been marked with a number, signifying the alarms raised by the monitor that particular day.
The total number of requests were 80,030.

As can be seen, most requests handled by the server are successful, with 96% benign ones and only 4% in the category of client or server error.

The malicious requests discovered are within these slices.
The server was accessed by a total of 7049 distinct host names during the analyzed time.

The majority of the hosts only asks for requests which are handled successfully by the server. The likely cause is that these hosts access the main page and then follows one or two links. If the site is working, this should not cause any errors.

All serious attacks were within this slice. One host did not follow this pattern and is thus found in the slice "mixed status codes."

Hosts asking only for status code:
- 2xx Success
- 3xx Redirection
- 4xx Client Error
- 5xx Server Error
- mixed status codes
**ATTACK PATTERNS**

- **Host using the tool cgiScan to perform the attack.**
- **Internal tests of the setup of the WWW server.**
- **Unidentified tool**
- **A simple probe made by hand?**

<table>
<thead>
<tr>
<th>Days (since supervision started)</th>
<th>Number of Attacks (malicious requests)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
</tbody>
</table>

Hosts trying only the three programs: phf, test-cgi, and handler in a very short time interval.
DEPLOYMENT

• WebWatcher is in operation for IBM customers
  – Batch processing
  – Weekly reporting

• Many attack attempts detected
  – Sites are highly visible and make attractive targets
  – WebWatcher signature database is growing richer ….

• WebWatcher interacts with the Tivoli Management Framework
  – Alerts are sent into the event correlation facility (TEC).
  – Alerts follow the IDWG data model definitions.
  – Alerts are correlated with ones coming other intrusion-detection systems (network-based).
pattern(cgi)  
(3) POST /vti_bin/shtml.exe/_vti_rpc HTTP/1.0  
(1) POST /vti_bin/shtml.exe/_vti_rpc HTTP/1.0

decision(followup)  
(4) GET /vti_inf.html HTTP/1.0  
(3) POST /vti_bin/shtml.exe/_vti_rpc HTTP/1.0  
(1) POST /vti_bin/shtml.exe/_vti_rpc HTTP/1.0

pattern(file)  
(4) GET /vti_inf.html HTTP/1.0

pattern(suspicious Cgi)  
(2) GET/phf?Qalias=x%0Aless%020/etc/passwd HTTP/1.1

pattern(cgi)  
(2) GET/phf?Qalias=x%0Aless%020/etc/passwd HTTP/1.1

decision(followup)  
(2) GET/phf?Qalias=x%0Aless%020/etc/passwd HTTP/1.1

207.71.2.61 -- [14/Aug/1999:21:28:01 +0000] "GET /vti_inf.html HTTP/1.0" 404 207

<table>
<thead>
<tr>
<th>Name</th>
<th>Threshold</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>warnings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pattern(file)</td>
<td>1</td>
<td>current level: 1.0</td>
</tr>
<tr>
<td>decision(followup)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>alerts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pattern(clientError)</td>
<td></td>
<td>^404</td>
</tr>
<tr>
<td>pattern(badStatus)</td>
<td></td>
<td>^[23]</td>
</tr>
<tr>
<td>pattern(file)</td>
<td></td>
<td>vti_inf.html</td>
</tr>
<tr>
<td>decision(followup)</td>
<td></td>
<td>warnings</td>
</tr>
</tbody>
</table>
pattern(cgi)

(1) GET/cgi-bin/test-cgi HTTP/1.0
(1) GET/carbo.dll HTTP/1.0
(1) GET/cgi-bin/wwwvsh名人.pl HTTP/1.0
(1) GET/lissamples/sdk/asp/docs/codebrws.asp HTTP/1.0
(1) GET/scripts/tools/newdsn.exe HTTP/1.0
(1) GET/cgi-bin/finger HTTP/1.0
(4) GET/cfdocs/expelval/openfile.cfm HTTP/1.0
(1) GET/cgi-bin/view-source HTTP/1.0
(1) GET/cgi-bin/bnbform.cgi HTTP/1.0
(1) GET/cgi-bin/textcounter.pl HTTP/1.0
(1) GET/cgi-bin/jj HTTP/1.0
(1) GET/cgi-bin/environ.cgi HTTP/1.0
(1) GET/cgi-bin/wrap HTTP/1.0
(1) GET/cgi-bin/websendmail HTTP/1.0
(1) GET/cgi-bin/rguest.exe HTTP/1.0
(1) GET/cgi-bin/unlg1.1 HTTP/1.0
(1) GET/lissamples/exair/howitworks/codebrws.asp HTTP/1.0
(1) GET/cgi-bin/classifieds.cgi HTTP/1.0
(1) GET/cgi-bin/edit.pl HTTP/1.0
(1) GET/cgi-bin/webgals HTTP/1.0
(1) GET/cgi-bin/survey.cgi HTTP/1.0
(1) GET/cgi-bin/handler HTTP/1.0
(1) GET/cgi-bin/info2www HTTP/1.0
(1) GET/cgi-bin/wwwboard.pl HTTP/1.0

<table>
<thead>
<tr>
<th>warnings</th>
<th></th>
<th>Threshold</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern(file)</td>
<td></td>
<td>1</td>
<td>current level: 1.0</td>
</tr>
<tr>
<td>decision(followup)</td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>alerts</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern(clientError)</td>
<td></td>
<td>^404</td>
<td></td>
</tr>
<tr>
<td>pattern(badStatus)</td>
<td></td>
<td>^[23]</td>
<td></td>
</tr>
</tbody>
</table>
FUTURE WORK

• Detect denial of service attacks through legitimate requests:
  – “The Slashdot effect”.
  – Distributed denial of service attacks can be carried out effectively nowadays.
  – This requires statistical tracking of legitimate requests -> quite costly.

• Deploy in distributed environment:
  – Challenge of distributed web servers (clusters, SP2, …).
  – The problem of sharing the suspicious hosts tree information is being studied.