Defeating Automated Web Assessment Tools

Saumil Shah
CEO, Net-Square
Author: “Web Hacking - Attacks and Defense”
BlackHat Briefings - Europe, Asia 2005
Purpose

• To illustrate the limitations of automated assessment tools.
• Identify new areas of research.
• Give more insight to developers.
Automated Web Assessment - Theory

- Testing the web server.
- Crawling the web application.
- Classifying the resources gathered.
- Mapping the application.
- Identifying attack points. (e.g. SQL, XSS)
- Identifying authentication points.
- Performing the attacks.
- Looking for known vulnerabilities.
Automated Web Assessment - Theory

• Testing logic:
  • Depends on HTTP response codes.
  • Responses can be easily changed.
  • e.g. all pages return 200 OK.

• Modern crawlers identify “error signatures” first.
  • e.g. Page Signatures (refer to my earlier talks on advanced HTTP assessment techniques).
Error Signatures

- Send a request for a non-existent page.
  - Record the response signature. (404 signature)
- Send a malformed HTTP request.
  - 400 / 500 signature.
- Proceed with crawling by identifying signatures from the responses, and not looking at the response codes.
Elimination of false positives

- Error signatures.
- String comparison.
- Regular expressions comparison.
- Certain heuristic techniques.
What is a crawler, actually?

- A functional HTTP client.
- Must mimic the browser as far as possible.
- Send HTTP requests.
- Receive the HTTP response.
- Parse the HTTP response header.
- Parse the HTTP response contents.
  - Sift through the HTML.
  - Recover from malformed HTML errors.
Ways to defeat crawlers

- Are you really a browser?
- Are you really a human sitting on a browser?
- … or are you a dog?
- Crawlers have overcome lots of hurdles so far...
- …but even they have limits.
- Humans and crawlers “use” the web application in different ways.
Browsers vs. Crawlers

- Well formed HTTP request header:
  - User-Agent string
  - HTTP referrer
- Cookie handling and cookie replay:
  - Easy.
  - Many crawlers do this quite well.
- Forced HTTP compression:
  - Not a lot of crawlers have gzip decoding.
  - Not difficult at all.
Browsers vs. Crawlers

- Javascript interpretation.
  - Difficult proposition for browsers.
  - Not entirely impossible.
  - Can cause loss of hair.
Humans vs. Crawlers

- Attacking the WYSIWYG principle:
  - Humans don’t click on clear pixels.
  - Developers still believe HIDDEN fields are secure!
- Humans do not cause a lot of errors.
  - …crawlers do.
- Visual recognition of an error situation:
  - “Something’s not right here”.
- Crawlers can fail all these tests.
Ways to bog down crawlers

• Random error responses, never the same response each time.
  • Will cause false positives in error identification.
  • Keep altering the HTML structure.
  • Use dictionary words.
• Custom error handlers.
  • Most web servers allow this
• Make the crawler crawl through errors.
Ways to bog down crawlers

- Random hyperlinks.
  - Links that lead to nowhere.
  - Cause errors that generate more links.
- Throw up non-existent error conditions:
  - e.g. SQL injection error messages.
  -Browsable directory outputs.
- Throw up non-existent HTML forms.
- A prototype crawler defeating mechanism.
- Causes the best of crawlers and assessment tools to throw up useless reports.
- To illustrate the point that nothing is as good as manual analysis and testing.
• Implemented as a set of PHP scripts.
• Easy to incorporate in any PHP driven application.
• Concepts are not rocket science:
  • can be ported to other platforms as well (e.g. ASP, ASP.NET, JSP, etc).
• Actively seeking collaborators!
• Publicly available soon.
PHP_GUARD - features

- Enforces strict session control.
  - Uses PHP’s session management APIs.
  - No cookies - no pages.
- Forced HTTP compression:
  - Coming soon!
- Random error generator.
PHP_GUARD - random error generator

- Varying HTTP response codes:
  - 404, 302, 200
- Structurally different HTML all the time.
- Based on dictionary words.
- Contains hyperlinks galore!
- Includes error strings to catch regexp matching.
- Includes HTML authentication forms.
PHP_GUARD - error count limit

- Error count limits set a threshold to the maximum number of errors a web client is allowed to cause.
- Per-session basis.
- If count exceeds the threshold...
- … you’re blacklisted.
- Ability to slow down responses.
  - Crawl 1000 links took a whole day!
PHP_GUARD setup

• /usr/local/apache/htdocs/php_guard
  • index.html
  • error_control.php
  • set_session.php
  • force_session.php
  • clearpixel.php
  • clearpixel.gif
PHP_GUARD Apache configuration

- httpd.conf
  - ErrorDocument 404 /php_guard/error_control.php
  - ErrorDocument 403 /php_guard/error_control.php
  - ErrorDocument 500 /php_guard/error_control.php
• Sample index.php file (starting point):

```php
<?php

    // initialize PHP_GUARD
    include("php_guard/set_session.php");

    // include globals
    include("include/global.php");

    // generate random clearpixel links
    include("php_guard/clearpixel.php");

?>
```
PHP_GUARD - use within applications

• Any other php file (not the starting point):

```php
<?php
    // initialize PHP_GUARD
    include("php_guard/force_session.php");
    ...
    ...
?>
```
PHP_GUARD - tests

• wget
• Paros
• NTO insight
Closing Thoughts

• “You need to know what you are doing!”
• Web Hacking: Attacks and Defense
  Saumil Shah,
  Shreeraj Shah,
  Stuart McClure
Goodies to follow

- New version of httprint coming out soon.
- NStools:
  - Net-Square’s toolkit.
- Contributions to Sensepost’s Wikto.
Thank you!

saumil@net-square.com

http://net-square.com/

BlackHat Briefings - Europe, Asia 2005