

# *Stopping Automated Application Attack Tools*

*Black Hat 2006 - Amsterdam  
March, 2006*

Gunter Ollmann  
*Director of X-Force*  
**Internet Security Systems**



## *Introduction*

- **Automated Attack Methods**
- **Common Protection Strategies**
- **Protection with Client-side Code**
- **Forcing a Client-side Overhead**
- **Thwarting Distributed and Future Attack Tools**



# ***Automated Attack Methods***

*“Greater is our terror of the unknown”*

*Titus Livius (59 BC – 17 AD)*

## *Automated Tool Functionality*

### **Most Common Methods:**

- Copying or **mirroring** a complete site
- Navigating a site by **scraping** or **Spidering**
- Identifying files and scripts through **CGI Scanning**
- **Brute Forcing** of variables and submissions
- Intelligent manipulation of variables by **Fuzzing**

## *Functions: Mirroring*

- **Theft** of intellectual property
- **Repackaging** of intellectual property
- **Key component of criminal deception**
  - Man-in-the-middle attacks
  - Phishing
  - Identity theft

## *Functions: Site Scraping & Spidering*

- Harvesting of **email addresses** for spam lists
- **Social engineering** attacks using personal data
- **Fingerprinting** server processes & software versions
- Understanding development techniques & **bypasses**
- Discovering “**hidden**” content
- **Mapping** of application functionality

## *Functions: CGI Scanning*

- Discovery of **administrative** pages or directories
- Identifying historically **vulnerable pages**
- **Default content** or samples
- Spotting “**hidden**” directories or file paths
- Cross-platform **shared** web services
- File download repository **locations**
- **Temporary** file content or backups

## *Functions: Brute Forcing*

Brute force guess an important piece of data making use of the following:

- Extensive **dictionaries**
- Common file or directory path **listings**
- Information gathered through **scraping & spidering**
- Information gathered through **CGI scanning**
- **Hybrid** dictionaries catering for obfuscation
- Automatic character **iteration**

## *Functions: Fuzzing*

- **Buffer overflows**
- **Type conversion** handling
- **Cross-site scripting - XSS**
- **SQL injection**
- **File and directory path navigation**
- **Validation differences** between client and server

## *Classes of Automated Tools*

**Can be broken down into the following:**

- **Web Spiders**
- **CGI Scanners**
- **Brute Forcers**
- **Automatic Fuzzers**
- **Vulnerability Scanners**



# ***Common Protection Strategies***

*“There is no security on this earth; there is only opportunity”*

*Douglas MacArthur (1880-1964)*

## *Server Host Renaming*

**Changing the “Server:” response in the HTTP headers to stop some types of fingerprinting**

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location:
http://www.example.com/PageIsHere.html
Date: Fri, 01 Jan 2005 01:01:01 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Fri, 01 Jan 2005 01:01:01 GMT
Content-Length: 1337
```

## *Blocking HEAD requests*

**Any HTTP HEAD request is rejected.**

```
HEAD /index.html HTTP/1.0
```

**Instead the tool must use:**

```
GET /index.html HTTP/1.0
```

**Slower to make requests – but the tool may drop the connection once the data is received**

## *Use of the REFERER Field*

**Make use of the HTTP REFERER field supplied by the client browser in the request**

```
GET /Next/ImGoingHere.html HTTP/1.1
Host: www.example.com
Referer: http://www.example.com/IWasHere.html
Accept-Language: en-gb
Content-Type: application/x-www-form-urlencoded
```

**Requires a method of validating a legitimate navigation path through the application**

## *Content-type Manipulation*

**Make use of the HTTP Content-Type defined in the server response or page contents**

```
HTTP/1.0 200 OK
Location: http://www.example.com/ImGoingHere.html
Server: Microsoft-IIS/5.0
Content-Type: text/html
Content-Length: 145
```

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=koi8-r">
```

## *Content-type Manipulation*

**Change the content page extension to anything – even image formats**

```
HTTP/1.0 200 OK
```

```
Location: http://www.example.com/ImGoingHere.jpg
```

```
Server: Microsoft-IIS/5.0
```

```
Content-Type: text/html
```

```
Content-Length: 145
```

## *HTTP Status Codes*

**Changing the status code of the response – e.g. responding with a “200 OK” instead of “404 File Not Found” etc.**

Status Code	Allocated Meaning
1xx	Informational
2xx	Successful
3xx	Redirection
4xx	Bad Request
5xx	Internal Server Error

**Every request generates a message effectively saying “the page requested exists”**

## Client-side Redirection

### Focusing on tools that make use of:

- HREF=
- 200 OK responses

```
HTTP/1.0 200 OK
Server: Microsoft-IIS/5.0
Content-Type: text/html
Refresh: 3;URL=http://www.example.com/ThisWay.html
```

```
<META HTTP-EQUIV="Refresh"
CONTENT="3;URL=http://www.example.com/ThisWay.html">
```

## Thresholds & Timeouts

### Focusing on tools that can't handle state:

- Use of cookie SessionID's
- Monitoring of time between submissions and requests
- Lockout procedures
- Timeouts
- Triggered thresholds

```
POST /Toys/IWantToBuy.aspx HTTP/1.1
Host: www.example.com
Referer: http://www.example.com/Toys/ILikeThisOne.aspx
Accept-Language: en-gb
Content-Type: application/x-www-form-urlencoded
Content-Length: 437
Cookie: SessionID=sse9d7783790
Postcode=SW11%201SA&Var1=Yes&Var2=Yes&Account=' ;--<H1>
```

### Focusing on tools that multithread submissions:

- Add tracking ID's to each URL
- Ensuring a single application navigation path
- Within page /BuyStageOne.aspx?track=1104569

`http://www.example.com/Index.aspx?track=1104569`

`http://www.example.com/BuyStageTwo.aspx?track=1104569`

- Within page /BuyStageTwo.aspx?track=1104570

`http://www.example.com/Index.aspx?track=1104570`

`http://www.example.com/BuystageTwo.aspx?track=1104570`

`http://www.example.com/BuyStageThree.aspx?track=1104570`

### Focusing on non human-readable links:

- Invalid links within HTML content
- “hidden” links such as web-bugs
- Coloured text

```
<BODY BGCOLOR="white">
Valid Links <BR>
<A HREF="http://www.example.com/index.html">Home</A><BR>
<A HREF=" ../Toys/IWantOneOfThose.html">Mine!</A><BR>
Invalid Link <BR>
<!-- HREF=" ../Bad.HTML"> -->
Hidden Link <BR>
<FONT COLOR="white"><A HREF=" ../Bad2.HTML">hidden</A></FONT>
</BODY>
```

# Graphical & Audio Turing Tests

## Focusing on non machine-readable puzzles:

- Difficult to read text against OCR systems
- Inclusion of sound recordings

### Registration Check

Type the characters that you see in this picture. [WJ](#)



[I can't see this picture.](#)

Characters are not case-sensitive.

<https://registernet.passport.net> - Type the registration characters you h...

This pop-up window links to an audio file so you can listen to and type the special characters required for registration. These characters help reduce spam that automated account registration tools create.

Instructions

1. Click the link below that begins with "Listen to the characters" to listen to the audio file.
2. Type the characters you hear into the text box below the link.
3. Click **Continue** to return to registration.

The characters are entered into the correct field on the registration page.

Notes

- Your computer must be able to play audio in .WAV format.
- If your browser plays the file in a different program, remember the characters, return to this page, and then type them in the text box below.
- The audio file is intentionally garbled.
- Characters are not case sensitive.

[Listen to the characters in the audio file.](#)

Type the characters

Done registernet.passport.net



# *Protection with Client-side Code*

*“Security puts a premium on feebleness”*

*H.G.Wells*

## *Strengths of Client-side Code*

- **Misconception of bypassing client-side code**
- **Bypassing is trivial, but not if you must execute it to do/calculate something that is validated at the server-side.**
- **Practically all current tools can't fully interpret scripting languages**

## *Token Appending*

- **Simplest method**
- **No calculation, just string concatenation**

```
<SCRIPT LANGUAGE="javascript">
  var token="0a37847ea23b984012"
  document.write("<A HREF='http://www.example.com/
  NextPage.aspx?JSToken="+token+"'>Link</A>")
</SCRIPT>
```

## Token Appending

```
<HTML>
<HEAD>
  <TITLE>Example Post</TITLE>
  <SCRIPT>
    function addtoken() {
      document.myform.token.value="0a37847ea23b984012";
      document.myform.submit();
    }
  </SCRIPT>
</HEAD>
<BODY>
  <FORM NAME="myform" ACTION="http://www.example.com/BuyIt.aspx"
METHOD="POST">
    <INPUT TYPE="TEXT" NAME="ItemName" >Item Name<BR>
    <INPUT TYPE="RADIO" NAME="Buy" VALUE="Now">Now
    <INPUT TYPE="RADIO" NAME="Buy" VALUE="Later">Later<BR>
    <INPUT TYPE="HIDDEN" NAME="token" VALUE="Fail">
    <INPUT TYPE="BUTTON" VALUE="SUBMIT" onClick="addtoken()">
  </FORM>
</BODY>
</HTML>
```

## Token Calculator

- Improved method
- Relies upon mathematical routines
- Can include complex routines that also incorporate other submission variables
- Harder to bypass using “smart” tools

```
<HEAD>
  <TITLE>Example Post</TITLE>
  <SCRIPT TYPE="text/javascript" SRC="crc32.js"></SCRIPT>
  <SCRIPT TYPE="text/javascript" SRC="cookies.js"></SCRIPT>
  <SCRIPT>
    function encodetoken() {
      var token = document.myform.token.value;
      var cookie = getCookie("SessionID");
      var page = location.pathname;
      document.myform.token.value = crc32(token + cookie + page);
      document.myform.submit();
    }
  </SCRIPT>
</HEAD>
```

## *Token Resource Metering*

- **Complex method**
- **Relies upon mathematical routines that require processing time to calculate**
- **Incurs an overhead at the client-side**
- **Something difficult to calculate by quick to validate**

$$y = \sqrt{y} \times \sqrt{y}$$



## ***Forcing a Client-side Overhead***

***“Do, or do not. There is no ‘try’.”***

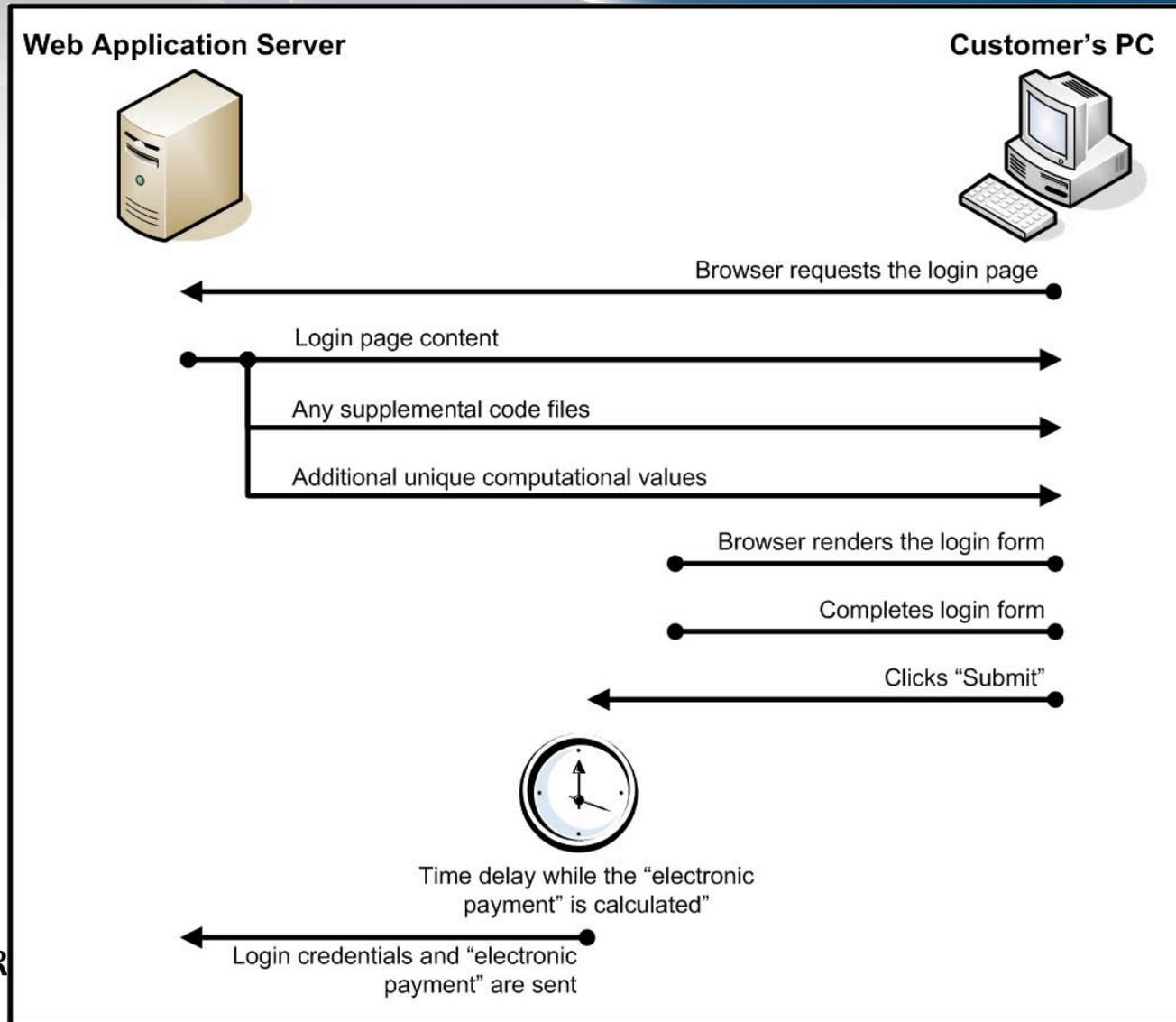
*Yoda (‘The Empire Strikes Back’)*

## *Understanding Resource Metering*

### **Why not just use server-side wait states?**

- **Shift computational load to client**
- **Better in load-balancing infrastructure**
- **Break non-script-aware tools**
- **Force an attacker to write custom attack tools**
- **...why not?**

# Understanding Resource Metering



## Borrowing from HashCash

HashCashDemo

HashCashDemo by Lapo Luchini		About
Challenge string	050316 hashcash	
bits to collide	<input type="range" value="26"/>	
Expected time	128 secs	Find collision
Hashcash	27 bits 0:050316:hashcash:KEcf030NCGi	
Hashcash hash	[000000156C0A04C3873F7655C1F519E95D13828A]	

[Download the Java ARchive](#) itself to use it locally or to see source code (and its [PGP signature](#)).

Done

Figure 1: Screenshot of a Java-based "hashcash" calculation at <http://www.lapo.it/hashcash.html>



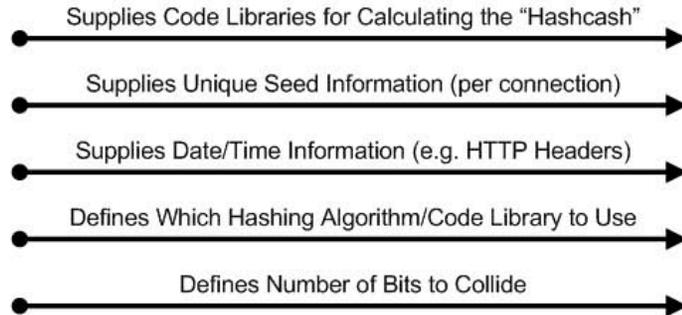
**Application Server**



**Customer's PC**



**Customer**



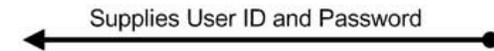
**SHA-1, MD5, etc...**

**050318**

**20/03/2005**

**0**

**20**



**Gunter**

Combines Necessary Data Variables into "Hashcash" Prefix:

**0:050318:UID=Gunter:20/03/2005:**

Using Algorithm "0", Collision Garbage is Added and SHA-1 Hash Calculated

0:050318:UID=Gunter:20/03/2005:K0m1h0M4gvA

0:050318:UID=Gunter:20/03/2005:K0m1h0M5gxB

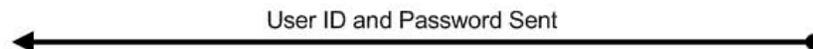
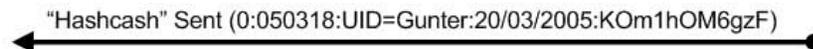
0:050318:UID=Gunter:20/03/2005:K0m1h0M6gyC

0:050318:UID=Gunter:20/03/2005:K0m1h0M7gaD

...

**0:050318:UID=Gunter:20/03/2005:K0m1h0M6gzF**

Until first "20" bits of the Hash Collide (e.g. "00000000000000000000")



Validates Structure of the Received "Hashcash"

If OK - Calculates Hash of Received "Hashcash"

Validates that first "20" Bits Collide to "00000000000000000000"

If OK - Proceeds with Next Authentication Process

If Not - Authentication Fails

If Not - Authentication Fails





# ***Thwarting Distributed and Future Attack Tools***

*“Never interrupt your enemy when he is making a mistake”*

*Napoleon Bonaparte (1769-1821)*

### **What about Distributed attack tools?**

- **Multiple IP sources of attack**
- **Variable levels of computing power**
- **Master/slave configuration of DDoS agents**

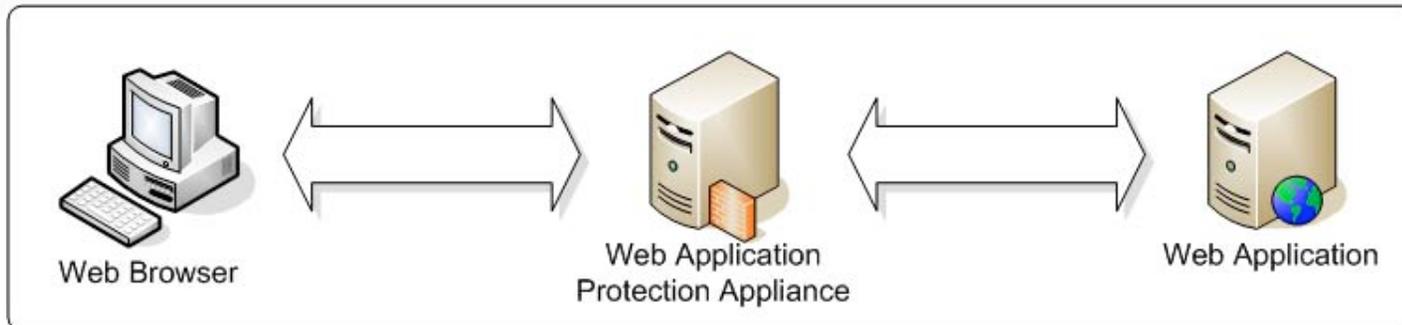
### **Focus upon slowing down the attack**

- **Techniques that force single navigation threads**
- **Techniques that force a computational overhead**
- **Use of thresholds and invisible wait states**

## *Protection Appliance?*

### Application Firewalls

- Failed technology – too complex & costly to setup
- Better value to pentest and code application securely



### Anti-tool Protection as an Appliance?

- Need to have zero or minimal configuration
- Proxy browser requests and server responses
- Rewrite server responses

## Protection Appliance?

### Automated attack protection with an appliance?

■ Server Host Renaming	Yes	Trivial
■ Blocking of HEAD Requests	Yes	Trivial
■ Use of REFERER Field	Yes	Easy
■ Content-Type Manipulation	Yes	Easy
■ HTTP Status Codes	Yes	Easy (with config.)
■ Client-side Redirection	Maybe	
■ Thresholds & Timeouts	Yes	Difficult (with config.)
■ Onetime Links	No	
■ Honeypot Links	Yes	Easy
■ Touring Tests	No	
■ Token Appending	No	
■ Resource Metering	Yes	Medium (with config.)

## *Next Generation Automated Tools*

**The next generation of tools will need to:**

- **Fully understand and parse client-side code**
- **Be highly customisable to each application**
- **Have some form of “intelligence” to make sense of server responses**

## *Limitations of the Techniques*

**There are limits to each and every technique.**

**Consider the impact of:**

- **Slow computers**
- **Slow connections**
- **Shared connections and DHCP**
- **Alienation due to script language requirements**
- **Processing power**
- **Mobile computing devices**

## *Future Research Areas*

### **Probable areas of future study:**

- **Tools that utilise second-order attacks and how they detect success**
- **Sandboxing of client-side code and execution to obtain HREF information**
- **Advances in automated responses to distributed attacks at the custom application level.**

***Thank You***

**Questions?**

ENTERPRISE SECURITY PLATFORM

PROVENTIA



 **INTERNET | SECURITY | SYSTEMS®**