Application Defense Tactics & Strategies - WAF at the Gateway

Shreeraj Shah
Blueinfy Solutions Pvt. Ltd.

Dubai – HackInTheBox 2009

Who Am I?

http://shreeraj.blogspot.com
shreeraj@blueinfy.com
http://www.blueinfy.com

- Founder & Director
  - Blueinfy Solutions Pvt. Ltd. (Brief)
  - SecurityExposure.com
- Past experience
  - Net Square, Chase, IBM & Foundstone
- Interest
  - Web security research
- Published research
  - Articles / Papers – Securityfocus, O'erilly, DevX, InformIT etc.
  - Tools – wsScanner, scanweb2.0, AppMap, AppCodeScan, AppPrint etc.
  - Advisories - .Net, Java servers etc.
- Books (Author)
  - Web 2.0 Security – Defending Ajax, RIA and SOA
  - Hacking Web Services
  - Web Hacking

Agenda

- Application Security Landscape
- Application Security Approaches
- Application Vulnerabilities - Demo
- WAF – A Quick Look
- .NET and HTTP processing
- Introducing IHTTPModule
- Security Framework through set of Modules
- Conclusion

Methods – Concepts, Code Walk and Demos

Application Security Landscape

Case of Portal

- Web 2.0 Portal – Buy / Sell
- Technologies & Components – Dojo, Ajax, XML Services, Blog, Widgets
- Scan with tools/products **failed**
- Security issues and hacks
  - **SQL injection over XML**
  - **Ajax driven XSS**
  - **Several XSS with Blog component**
  - **Several information leaks through JSON fuzzing**
  - **CSRF on both XML and JS-Array**
    - » HACKED
    - » DEFENSE

Blueinfy

Case of Banking

- Scanning application for vulnerabilities
- Typical banking running with middleware
- Vulnerabilities
  - **Profile manipulation (Logical and Hidden values)**
  - **XSS**
  - **Strong session management but URL rewriting**
  - **SQL is impossible in this case**

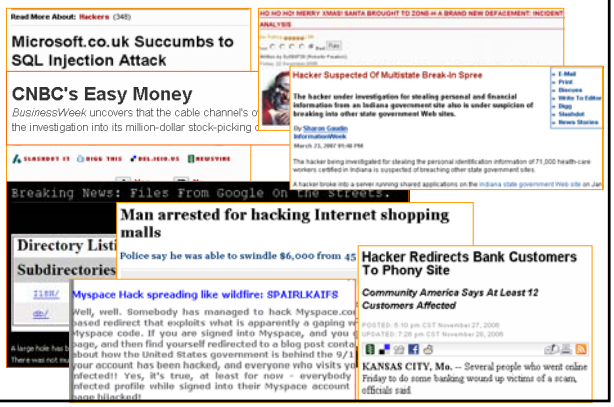
Blueinfy

Application Security State

- 95% companies hacked from web ports [FBI/CSI]
- 3 out of 4 web sites are vulnerable to attack (Gartner)
- Every 1500 lines of code has one security vulnerability (IBM Labs)
- 2000 attacks / week for unprotected web site

Blueinfy

Real life hacks & trends

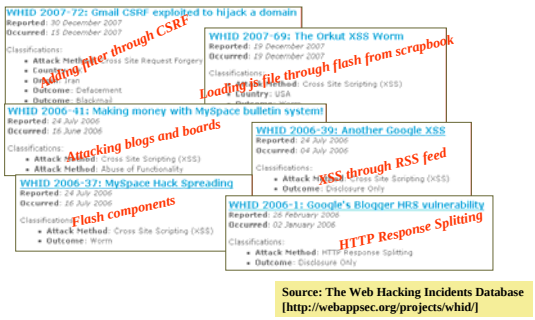


Next Generation Applications - 2.0

- 80% of companies are investing in Web Services as part of their Web 2.0 initiative (McKinsey 2007 Global Survey)
- By the end of 2007, 30 percent of large companies have some kind of Web 2.0-based business initiative up and running. (Gartner)
- 2008. Web Services or Service-Oriented Architecture (SOA) would surge ahead. (Gartner)

Blueinfy

Real life Cases – 2.0



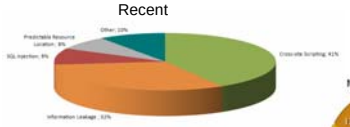
Source: The Web Hacking Incidents Database [http://webappsec.org/projects/whid/]

Blueinfy

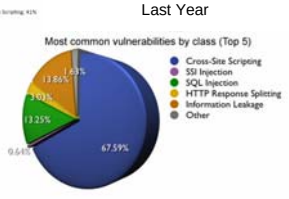
Generic vectors 1.0/2.0

- Most common vulnerabilities

Recent



Last Year



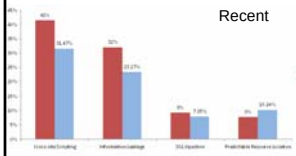
Source – Web Application Security Consortium

Blueinfy

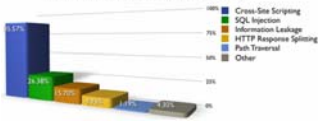
Generic threats – 1.0/2.0

- Threat types

Recent



Last Year

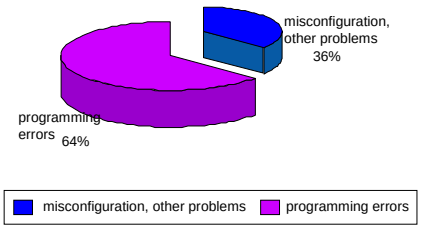


Source – Web Application Security Consortium

Blueinfy

Root cause of Vulnerabilities

CSI Security Survey : Vulnerability Distribution



Blueinfy

OWASP Top 10

A1 - Cross Site Scripting (XSS)	XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.
A2 - Injection Flaws	Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.
A3 - Malicious File Execution	Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users.
A4 - Insecure Direct Object Reference	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects without authorization.
A5 - Cross Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.
A6 - Information Leakage and Improper Error Handling	Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data, or conduct more serious attacks.
A7 - Broken Authentication and Session Management	Account credentials and session tokens are often not properly protected. Attackers compromise passwords, keys, or authentication tokens to assume other users' identities.
A8 - Insecure Cryptographic Storage	Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.
A9 - Insecure Communications	Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.
A10 - Failure to Restrict URL Access	Frequently, an application only protects sensitive functionality by preventing the display of links or URLs to unauthorized users. Attackers can use this weakness to access and perform unauthorized operations by accessing those URLs directly.

Source – OWASP – http://owasp.org/

Blueinfy

CVE/CWE - Errors

- **Insecure Interaction Between Components**
  - **These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.**
  - **CWE-20: Improper Input Validation**
  - **CWE-116: Improper Encoding or Escaping of Output**
  - **CWE-89: Failure to Preserve SQL Query Structure (aka 'SQL Injection')**
  - **CWE-79: Failure to Preserve Web Page Structure (aka 'Cross-site Scripting')**
  - **CWE-78: Failure to Preserve OS Command Structure (aka 'OS Command Injection')**
  - **CWE-319: Cleartext Transmission of Sensitive Information**
  - **CWE-352: Cross-Site Request Forgery (CSRF)**
  - **CWE-362: Race Condition**
  - **CWE-209: Error Message Information Leak**

Source – CWE/CVE - http://cwe.mitre.org/top25/index.html

Blueinfy

CVE/CWE - Errors

- **Risky Resource Management**
  - The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.
  - CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer
  - CWE-642: External Control of Critical State Data
  - CWE-73: External Control of File Name or Path
  - CWE-426: Untrusted Search Path
  - CWE-94: Failure to Control Generation of Code (aka 'Code Injection')
  - CWE-494: Download of Code Without Integrity Check
  - CWE-404: Improper Resource Shutdown or Release
  - CWE-665: Improper Initialization
  - CWE-682: Incorrect Calculation

Source – CWE/CVE - http://cwe.mitre.org/top25/index.html

Blueinfy

CVE/CWE - Errors

- **Porous Defenses**
  - The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.
  - CWE-285: Improper Access Control (Authorization)
  - CWE-327: Use of a Broken or Risky Cryptographic Algorithm
  - CWE-259: Hard-Coded Password
  - CWE-732: Insecure Permission Assignment for Critical Resource
  - CWE-330: Use of Insufficiently Random Values
  - CWE-250: Execution with Unnecessary Privileges
  - CWE-602: Client-Side Enforcement of Server-Side Security

Source – CWE/CVE - http://cwe.mitre.org/top25/index.html

Blueinfy

PCI-DSS requirements

- To secure application
  - **Put WAF at the gateway**
  - **Get your source code audited**

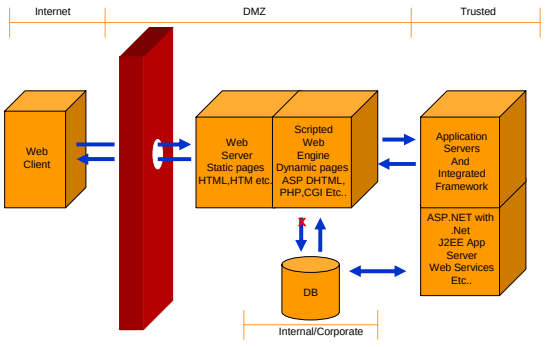
Blueinfy

New Attack Vectors

- XML manipulation
- SOAP and XML-RPC attacks and tempering
- CSRF with Ajax and Flash
- XSS with JSON streams
- Mashup and RSS attacks

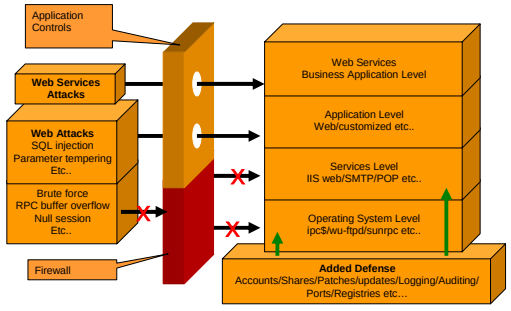
Blueinfy

Web Application Layout



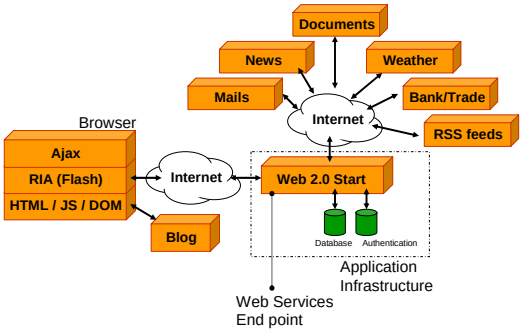
Blueinfy

Attack Surface and Controls



Blueinfy

Web 2.0 Architecture



Blueinfy

Application Security Approaches

Blueinfy

How to defend?

- Two approaches
  - **Secure Coding and having proper validations at all levels to guard application layer. (Strategic)**
  - **Application layer traffic filtering to detect and block malicious requests/responses. (Tactical)**

Blueinfy

Secure Coding

- It is perfect and ideal approach.
- But…
  - **Needs recoding**
  - **Takes longer time in fixing**
  - **Quick fix is required many times**
  - **QA process after changes**
  - **High cost**
- Any work around?

Blueinfy

Web Application Firewall (WAF)

- HTTP request and response filtering like traditional firewall.
- But it is specific to Application layer and rules should be well crafted.
- It is catching up and successful in detecting and blocking unintended traffic.
- It can block SQL injection, XSS, CSRF and many other attack vectors.

Blueinfy

Application Vulnerabilities

- Let's look at some vulnerabilities
  - **SQL (JSON and Traditional)**
  - **XSS**
  - **XPATH**
- Detecting it….
  - **Scanning**
  - **Code Analysis**

Blueinfy

WAF – A Quick Look

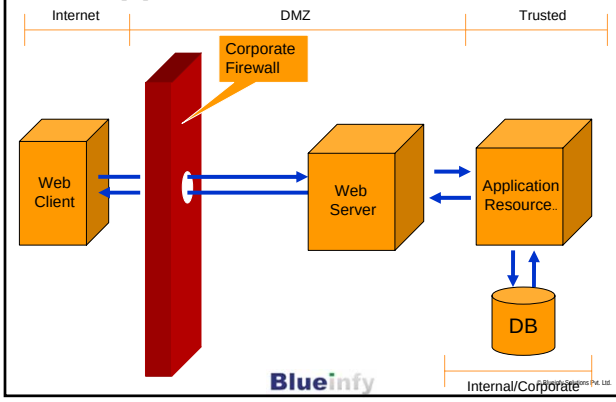
Blueinfy

Web Application Firewall (WAF)

- Advantages
  - **Quick to add rules**
  - **Can act as first line of defense**
  - **No recoding is required**
  - **Easy to implement and manage**
- Disadvantage
  - **Performance a major hit**
  - **Rule based and bypass is possible**

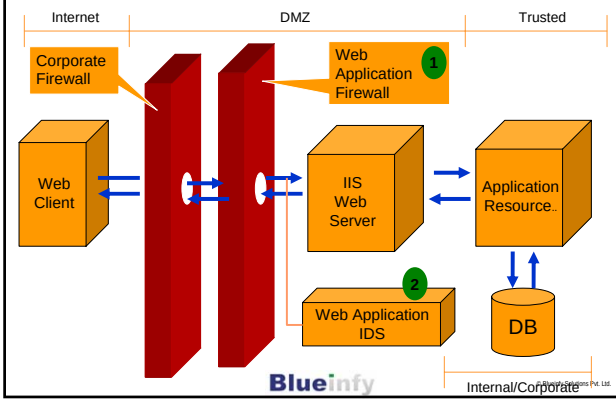
Blueinfy

Application Infrastructure



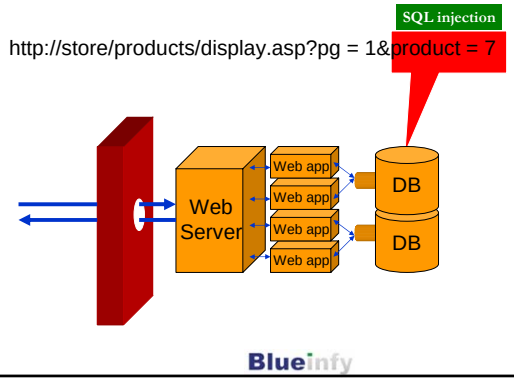
Blueinfy

WAF in Action



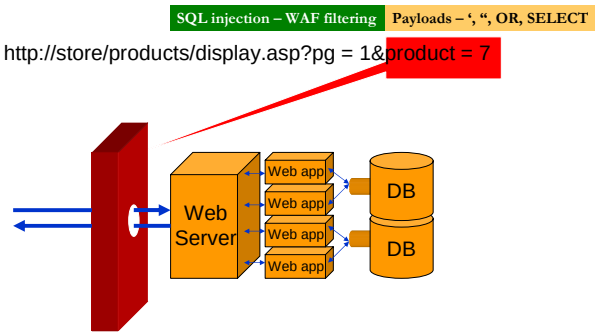
Blueinfy

SQL injection attack



Blueinfy

SQL injection attack



Blueinfy

WAF models

- Following models are possible
  - **Network traffic level filtering [SSL is an issue]**
  - **Host level at Web Server**
  - **Host level + Reverse Proxy**

Blueinfy

.NET and HTTP processing

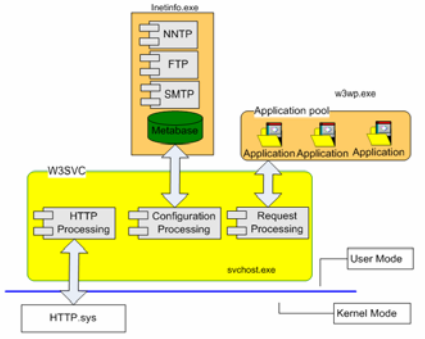
Blueinfy

IIS architecture

- It is important to understand how IIS works?
- .NET gets integrated into IIS and applications can leverage the events
- IIS7.0 is coming up with a change that can help in building WAF

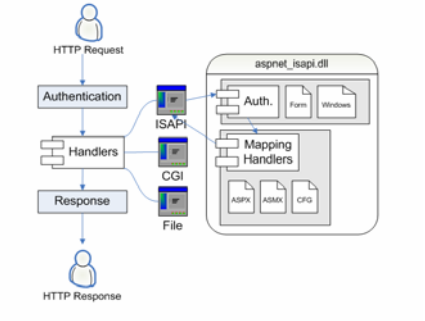
Blueinfy

IIS higher level view



Blueinfy

IIS 6.0 + ASP.NET



Blueinfy

IIS 6.0 - Limitation

- ASP.NET is not having direct access to the HTTP pipe
- Can access ASP.NET requests only
- Framework is part of ISAPI and hooked to IIS
- Needs C++ based hooks to access generic pipe

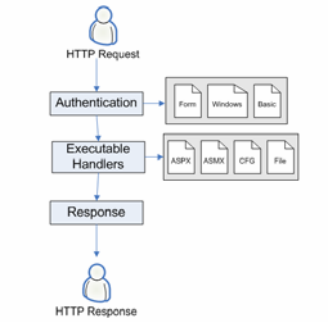
Blueinfy

Solved!

- IIS 7.0 – Change in Architecture
- Integrated mode
- .NET assemblies can be hooked directly to the pipe
- Full access to HTTP requests
- Can handle both .NET based as well as generic requests
- Access to all incoming requests…

Blueinfy

IIS 7.0 – Integrated Mode



Blueinfy

Introducing IHTTPModule

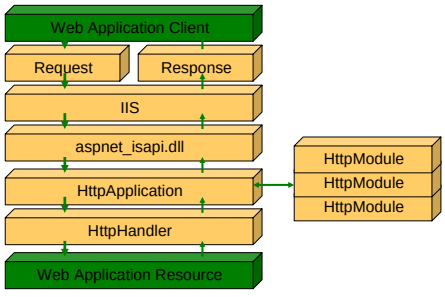
Blueinfy

How to hook?

- Web application has separate scope and HTTP pipeline can be accessed.
- HTTP request can be accessed before it hits application resources.
- HTTPModule and HTTPHandler are defense at your gates. …

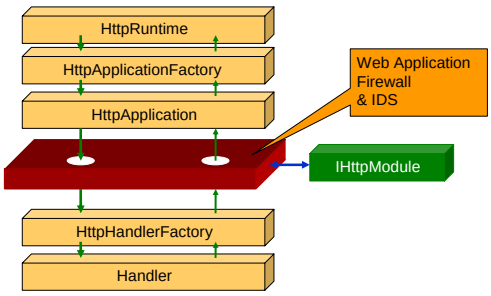
Blueinfy

HTTP pipe for .NET



Blueinfy

Interfaces and Hooks



Blueinfy

Leveraging Interfaces

- HTTPModule and HTTPHandler - can be leveraged.
- Application layer firewall can be cooked up for your application.
- Similarly IDS for web application can be developed.
- It sits in HTTP pipe and defend web applications.

Blueinfy

For IIS 7.0

- Integrated mode with full access
- Possible to cook up reverse proxy as well
- Traffic can be controlled at the gates
- Sound defense can be created with minimal coding
- Your module can be on top of the pipe
- Can access
  - **HttpResponse.Headers**
  - **HttpRequest.Headers**
  - **HttpRequest.ServerVariables**

Blueinfy

Implementing IHTTPModule

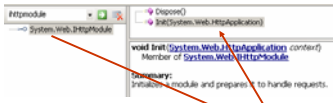
Blueinfy

IHTTPModule

- Managed code in C# can be hooked into HTTP pipe.
- Module can help in filtering HTTP requests.
- Let's see its implementation.

Blueinfy

IHTTPModule



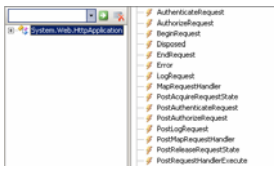
```
public class iAppFilter : IHttpModule
{

}
```

Access to HttpApplication

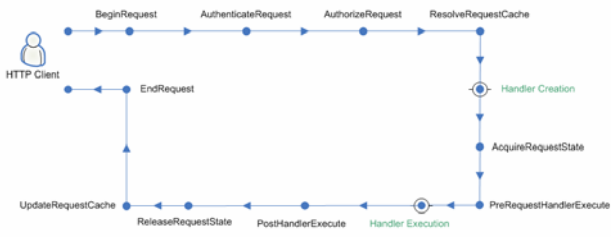
Blueinfy

HttpApplication



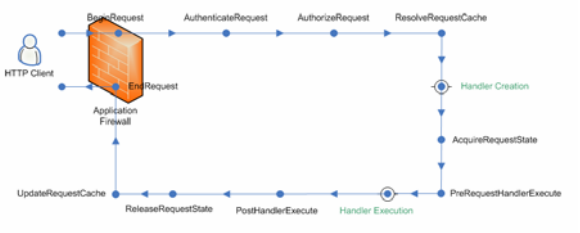
Blueinfy

Event Mapping



Blueinfy

Event Trapping and Firewall



Blueinfy

Accessing HTTP request

- Access with BeginRequest
  - **Access to Http Context**
  - **Access to headers**
  - **All server variable**
  - **Complete access for filtering**

Blueinfy

Hooking to HTTP pipe

```
public void Init(HttpApplication application)
  {
                application.BeginRequest +=
      (new EventHandler(this.Application_BeginRequest));
```

```
private void Application_BeginRequest(Object source,
      EventArgs e)
  {
    HttpApplication application = (HttpApplication)source;
    HttpContext context = application.Context;
```

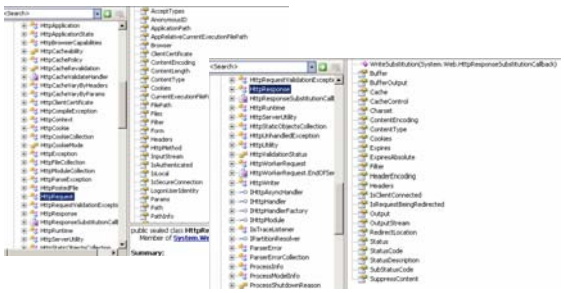
Blueinfy

Processing POST

```
if (app.Request.ServerVariables["REQUEST_METHOD"] == "POST")
{
        long streamLength = app.Request.InputStream.Length;
        byte[] contentBytes = new byte[streamLength];
        app.Request.InputStream.Read(contentBytes, 0, (int)streamLength);
        postreq = System.Text.Encoding.UTF8.GetString(contentBytes);
```

Blueinfy

Request / Response



Blueinfy

Putting it in action

- DLL get created after compilation
- Module in Bin folder
- Adding to config file
- It is different with IIS 7.0 for integrated mode
- Directives are different
- Let's see in detail

Blueinfy

Security Framework through set of Modules

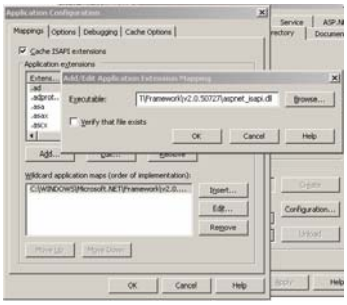
Blueinfy

Accessing all requests

- It is important to access all incoming HTTP requsts
- IIS 6.0 limitation – can be overcome by using wildcard mapping [Some what]
- IIS 7.0 – Leveraging integrated mode

Blueinfy

IIS 6.0 – Wildcard mapping



Blueinfy

IIS 7.0 – Integrated mode

```
<modules>
  <add name="iAppWall" type="iAppWall"/>
</modules>
```

Blueinfy

Security Modules

- Various module can be cooked up.
- Authorization, Authentication, Filtering, XML processing, IDS etc.
- All of them can be part of one DLL or multiple.

Blueinfy

Authorization Module

- Limited access to IP addresses
- Blocking sensitive directories
- Session based access to various area of application

Blueinfy

Validation Module

- Detecting attack vectors like XSS or SQL injection
- Blocking those requests at the module level
- Total security to all incoming parameters both over GET and POST

Blueinfy

Web 2.0 Security Module

- Web 2.0 runs on XML, JSON, JS-Array etc..
- Intelligent module to detect these sort of traffic and block malicious requests
- Protecting Web Services running over SOAP, XML/JSON-RPC, REST etc.

Blueinfy

CSRF Defense Module

- Cross Site Request Forgery is a big concern for sensitive forms
- Protection by referrer tag or token by HTTP module
- Securing application against CSRF attack vectors

Blueinfy

Response Filtering Module

- Limited response filtering for critical resources
- Monitoring outgoing requests
- Capturing suspicious traffic and blocking them
- Web 2.0 framework defense – RSS or proxy based responses

Blueinfy

IDS Module

- Logging all suspicious requests for forensic use
- Logging and monitoring can be improved
- Logging to central database, file or OS events.

Blueinfy

Reverse Proxy Module

- Defending non IIS applications with reverse tunneling.
- IIS 7.0 as front end server and securing internal servers
- Complete control over full traffic going in/out

Blueinfy

Conclusion

- Next generation .NET application can be defended by IHTTPModules
- IIS 7.0 – Integrated mode is going to play a big role
- Web 2.0 application needs better filtering capabilities and IHTTPModule can deliver it

Blueinfy

http://shreeraj.blogspot.com
shreeraj@blueinfy.com
http://www.blueinfy.com

Questions

Blueinfy