H D Moore
Director of Security Research
BreakingPoint Systems

Bitten on the ASP

(How NOT to deploy ASP.NET applications)

Blue Hat 3 Conference

Agenda

- Introductions
- Practical ASP.NET security
- Common ASP.NET 1.1 flaws
- Changes in ASP.NET 2.0
- Security and auditing tools
- Real-world ASP.NET stats

Introductions - Who?

- BreakingPoint Systems
  - Director of Security Research
  - We build hardware to break things
- The Metasploit Project
  - Founder, developer, researcher
  - We build software to break things

Introductions - Why?

- ASP.NET is a success!
  - Simple migration path for legacy ASP apps
  - Cleaner code, easier to manage, less bugs
  - Still integrates with legacy COM objects
- Security features integrated
  - Framework handles common use cases
  - Prevents common security mistakes

Introductions - What?

- Developers are still lazy ;-)
  - Copy and paste from example code
  - Leave debugging features in production
  - Seek the shortest path to resolve a problem
- Lots of “magic” contributes to flaws
  - How many devs understand the ViewState?
  - Information leaks are still very common...

Practical – ASP.NET configuration

- Configuration files (web/machine)
  - Defaults stored in machine.config (1.1)
  - Web.config stored in application directories
  - Control nearly all security settings
  - Simple to edit XML format
- Default configuration is great
  - Information leaks limited to localhost
  - Debug disabled, Trace disabled, etc

Practical – IIS configuration

- 18 file mappings in ASP.NET 1.1
  - All extensions processed by aspnet_filter.dll
  - Dispatched by extension to processing code
- Prevents remote access to source files
  - .csproj, .resources, .config, .licx, .cs, .resx
  - .webinfo, .vb, .vbproj, .vsdisco
- Does not prevent access to data files
  - Data sources: .mdb, .xls, .xml, etc
  - IDE leftovers: .vsc, .vsc, .xslt, etc

Common Flaws – Configuration

- Most common flaws are the simplest
  - The customErrors setting is disabled
  - Debugging is left enabled in production
  - Tracing accidentally left enabled
- customErrors=”Off”
  - Every ASP.NET app can be forced to error
  - Stack traces, physical paths, fun messages...
  - Allows for trivial SQL injection exploitation

Common Flaws – Configuration

- <compilation debug=”true”/>
  - Exposes code snippets with customErrors=Off
  - Allows the DEBUG HTTP verb...
- <trace enabled="true" localOnly="false"/>
  - Exposes everything an attacker could want
  - The least common but most dangerous issue
  - Trace + auth == remote user/pass list

Common Flaws – Exposed data files

- MDB data sources
  - Commonly left in the web directories
  - Security is based on IIS permissions
  - Permissions often lost during deployment
  - MS's IssueTracker sample does this...
- XML data files
  - Often used for credentials with Forms auth.
  - Example buried in the MSDN documentation
- Guessing file names generally trivial...

Common Flaws – “Invisible” controls

- Applications often hide certain controls
  - Trying to restrict access based on rights
  - Hiding features that are still in development
  - Set Visible to false or just remove the <a> link.
- Remember that PostBack feature?
  - Invisible controls can still be accessed!
  - __EVENTTARGET='invisibleCtrl1'
- Control names exposed in ViewState
  - Hard to guess names are still trivial to find...
  - Assuming VS encryption is off :-)

Common Flaws – Cookieless Sessions

- Session ID stored in the URL
  - Passed from page to page as user navigates
  - Works around the “no cookies” .GOV issue
- Session IDs are exposed in referrers
  - Clicking an external link gives away the ID
- Exposed to “session fixation” attacks
  - Attacker obtains a valid session ID
  - Sends URL to victim with ID already in it
  - Victim authenticates to the target site
  - Attacker follows victim using the same ID

Common Flaws – Miscellaneous

- SQL injection
  - Still a problem with ASP.NET apps
  - Easy to avoid, but people are lazy...
- XML injection
  - XML injection can be just as a bad as SQL
  - Data sources, AJAX, other XML-RPC...
- Unmanaged code
  - If the app actually wants to **do** something...
  - Many .NET features rely on Native interfaces
    - OLE, ODBC, CryptoAPI, StateServer, GDI+...

Common Flaws – ViewState

- ViewState basics
  - Base64 string of encoded 'tuples'
  - Client-side storage of control state
  - Can expose sensitive data...
- ViewStateMac
  - Hash appended to the clear-text data
  - Prevents user-modification of data
- Disabling ViewStateMac
  - Increases page load performance
  - Exposes the app to manipulation...

Common Flaws – Conclusion

- ASP.NET vs Developers
  - Classic ASP left all security up to the user
  - Obviously this didn't work :-)
  - ASP.NET is a major improvement...
- But expectations have changed!
  - Developers now rely on the Framework
  - More “magic”, less knowledge required
  - Everything now depends on the Framework

ASP.NET 2.0 – Security improvements

- Major improvements!
  - Consistent data file protection (App_*)
  - ValidateEvent() now prevents PostBack tricks
  - SiteMapProvider now has securityTrimming
  - Cookieless sessions slightly less vulnerable
  - ViewState can perform “smart” encryption
- IIS integration
  - Maps 42 extensions to aspnet_filter.dll!
  - Better integration with IIS 6.0 features
  - Security features not backwards compatible…

Tools – Remote security auditing

- Vulnerability assessment tools
  - Nessus includes plugins for ASP.NET
  - Commercial: eEye, nCircle, Qualys,etc
- Application assessment tools
  - OWASP's Berreta Project
  - Nikto, Whisker, Paros
  - Commercial: AppSec Inc, SPI Dynamics, etc
- ASP.NET specialty tools
  - DNAScan.pl :-)

Tools – Local security auditing

- ASP.NET Baseline Security (ANBS)
  - Finds unpatched flaws and bad configurations
  - Exposes cross-client issues w/shared hosting
  - Classic ASP version available too (ACSA)
- SAM'SHE
  - ANBS for non-technical users
  - Exposes poor shared hosting security
  - Non-intrusive, doesn't include exploits

Tools – Application “firewalls”

- Validator.Net
  - External request validation for your application
  - Useful for securing third-party applications
  - Does not require app source code to use
- DefApp
  - Validator.Net enhanced with mod_security
  - Filter requests and block known attacks
- Other solutions
  - mod_security, filtering proxies, SecureIIS

Tools – Local security analyzers

- PermCalc
  - Determine what permissions your app needs
  - Restrict everything else via CAS :-)
- Reflector
  - “Source” browser for managed binaries :-)
  - Hurray for Microsoft's lack of obfuscation!
- .NetMon (Foundstone)
  - Function tracing and application profiling
  - Equivalent to 'ltrace' for managed apps

Stats – How bad is it?

- Sample of 200 web sites
  - 62% allow remote NTLM authentication
  - 28% have Debugging enabled
  - 15% have customErrors disabled
  - 15% disclose physical web path
- ASP.NET versions
  - 70% running 1.1.*
  - 19% running 2.0.*
  - 11% hiding their version

Stats – MSFT examples

- Physical path disclosure
  - <censored>
- Debug compilation enabled
  - <censored>
- NTLM authentication enabled
  - <censored>

Questions?

Contact information:

hdm[at]metasploit.com

http://metasploit.com/