.NET from the Hacker’s Perspective

Drew Miller

Drew.miller@securityage.com
.NET from the Hacker’s Perspective

What Hackers Dislike
Risk
What Hackers Like
Summary
What Hackers Dislike

- .NET Buffer Overflows
- Role Security
- CAS Code Access Security
- Cryptography
- Summary
.NET Buffer Overflows

- Managed Code
- Legacy Code
- The Developer Mind Set
Self-resizing variables
.NET Framework keeps fixed sized variables from being copied to by variable sized variables
It is still very common to use previously coded modules and routines

Why reinvent the wheel?

Security?
No buffer overflows in .NET? I no longer need to bounds check my variable length variables.

Less could mean more
What Hackers Dislike

- Buffer Overflows
- Role Security
- CAS Code Access Security
- Cryptography
- Summary
Role Security

- Don’t call me… I’ll call you
- Framework for defining class and function level call security
What Hackers Dislike

- Buffer Overflows
- Role Security
- CAS Code Access Security
- Cryptography
- Summary
CAS Code Access Security

- Mobile Code
- Default user permission settings for the Internet Zone makes hard case for ignoring use in public market
- Signing Assemblies (GAC)
- Key Management (Source Safe)
What Hackers Dislike

- Buffer Overflows
- Role Security
- CAS Code Access Security
- Cryptography
- Summary
Cryptography

- Encrypt vs. Encode vs. Hashing
- Minimal Coding Requirements
- Fast
- Easy Key Management
- XML
What Hackers Dislike

- Buffer Overflows
- Role Security
- CAS Code Access Security
- Cryptography

Summary
What Hackers Dislike: Summary

- Buffer Overflow Protection
  - Always bounds check
- Role Based Security In Code
  - Validate who is allowed to call functions
- Newer Code Difficult To Trojan
  - Avoid Trojans like “FunLove”
- Everything Encrypted
  - Avoid information leakage
.NET from the Hacker’s Perspective

What Hackers Dislike
Risk

What Hackers Like

Summary
Risk

- Everyone has a deadline
- Everyone has a performance requirement
- NEW -> Everyone has a security requirement
- Dollar -> Security -> Risk
.NET from the Hacker’s Perspective

What Hackers Dislike
Risk

What Hackers Like

Summary
What Hackers Like

- Information Leakage
  - View state
  - XML
  - SQL errors
  - Web errors
  - Cookies
  - URLs

- Does easy to develop mean easy to exploit?
  - Cross Site Scripting
  - Reaply/Hijacking
  - Injection XML/SQL
Information Leakage: View State

- **View State**
  - Base64 encoded
  - Dynamic properties of server-side controls
  - Map to exposures and vulnerabilities
Information Leakage: XML

- The world of plaintext
- Sniffed traffic can lead to information leakage
- Encrypting XML can be cumbersome and degrades performance
- Signing XML is also difficult and degrades performance
Information Leakage: SQL errors

- Not once, not twice, but N times
- The exploitation road map to accessing your data…
- The small to medium company go-to-guy
Programmers are logical
Hackers are logical
Login example
  - Password Invalid
  - User Invalid
  - User or Password Invalid
Enumeration functions
Information Leakage: Cookies

- Stored on client
- Modifiable
- Extents to any client side persisted state information
- Serialization
- Client to server program configuration files (non-HTTP)
Information Leakage: URLs

- URLs tell a story
  - System Administrator/Deployment Know-How
  - Incrementing variables
  - Arguments to functions
What Hackers Like

- Information Leakage
  - View state
  - XML
  - SQL errors
  - Web errors
  - Cookies
  - URLs

- Does easy to develop mean easy to exploit?
  - Cross Site Scripting
  - Replay/Hijacking
  - Injection XML/SQL
If I do not incorporate security knowledge and processing during development and deployment of all resources, regardless of whether the access to that resource is anonymous or authenticated, is exploitation possible? YES.
What Hackers Like

- Information Leakage
  - View state
  - XML
  - SQL errors
  - Web errors
  - Cookies
  - URLs

- Does easy to develop mean easy to exploit?
  - Cross Site Scripting
  - Replay/Hijacking
  - Injection XML/SQL
Cross Site Scripting

- HTML inputs for everyone
- How do I validate?
- Just don’t do it if you can avoid it… good design makes for good security
What Hackers Like

- Information Leakage
  - View state
  - XML
  - SQL errors
  - Web errors
  - Cookies
  - URLs

- Does easy to develop mean easy to exploit?
  - Cross Site Scripting
  - Replay/Hijacking
  - Injection XML/SQL
Replay / Hijacking

- Session Hijacking
  - HTTP Session IDs
  - .NET Forms Authentication
- Got SSL?
  - Hey! Cross Site Scripting to the rescue...
- Validation = ( Authentication -> Session )
  * Each Request
What Hackers Like

- Information Leakage
  - View state
  - XML
  - SQL errors
  - Web errors
  - Cookies
  - URLs

- Does easy to develop mean easy to exploit?
  - Replay/Hijacking
  - Injection XML/SQL
Injection XML/SQL

- SOAP
- Dynamic SQL
- .NET SqlParameter
.NET from the Hacker’s Perspective

What Hackers Dislike
Risk
What Hackers Like
Summary
Parameter validation still key to a majority of vulnerabilities
Why authenticate when you can hijack?
Sign code, encrypt data, or else…
Server side security much better… communication security still difficult to secure with ease, but definitely possible
.NET from the Hacker’s Perspective

Drew Miller

Drew.miller@securityage.com