



0131197 2576 0616026

.NET from the Hacker's Perspective: Part II

Drew Miller

drewm@blackhat.com



0121197 2576 0616026

.NET from the Hacker's Perspective: Part II

Authentication Systems

Hardcore Authorization

Cryptography Ignored

Distributed Denial-of-Service

Summary



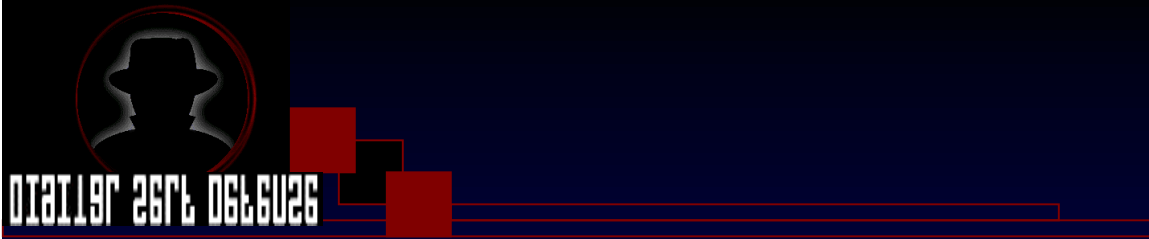
Authentication Systems

- Username and password
- Where is challenge and response?
- Using certificates
- Single Sign-On (SSO)
- Clients are easier to hack anyway to gain credentials



Username and Password

- Authentication systems allow anonymous hackers to access the authentication system
 - Information leakage
 - Errors tell more than they should
 - “Username and or password are invalid”



Authentication Systems

- Username and password
- **Where is challenge and response?**
- Using certificates
- Single Sign-On (SSO)
- Clients are easier to hack anyway to gain credentials



0121197 2576 0616026

Challenge and Response

- Server gives a piece of data to the client which assumed is packet sniffed by hackers
- Client encrypts piece of information with password hash and sends it back
- Server can validate the client by decrypting the data with the client's password hash which is stored on the server



Challenge and Response (cont.)

- Encrypted data is sniffed and can be brute forced
- How often are passwords changed for publicly accessible systems?
- Violates rule about changing secrets which can be brute forced before they can be brute forced



0121197 2576 0666026

Challenge and Response (cont.)

- Whether you use this method, plain text or transmitting hashes over the network...
- You must establish a secure tunnel using SSL (minimum) or for TCP applications a custom cryptographic key exchange and session key generation *BEFORE* authenticating the client.



Authentication Systems

- Username and password
- Where is challenge and response?
- **Using certificates**
- Single Sign-On (SSO)
- Hack the Client
 - Clients are easier to hack anyway to gain credentials



Using Certificates

- Prove I'm the server or client
- Smart cards
 - What you have and what you know...
- Better, but not invulnerable
- What you have is a must!



Authentication Systems

- Username and password
- Where is challenge and response?
- Using certificates
- **Single Sign-On (SSO)**
- Hack the Client
 - Clients are easier to hack anyway to gain credentials



Single Sign-On (SSO)

- .NET Passport
 - <http://archives.neohapsis.com/archives/ntbugtraq/2003-q2/0066.html>
 - Reset password
 - Personal information leakage / Identity theft
- Instead of hacking authentication and getting access to one system, now I have access to hundreds



Authentication Systems

- Username and password
- Where is challenge and response?
- Using certificates
- Single Sign-On (SSO)
- **Hack the Client**
 - Clients are easier to hack anyway to gain credentials



Hack the Client

- Bandwidth for everyone!
 - Broadband delights
- Camping and Wireless
- My older Windows box does all that it needs to... why upgrade to a new Windows operating system



0121197 2576 0666026

.NET from the Hacker's Perspective: Part II

Authentication Systems

Hardcore Authorization

Cryptography Ignored

Distributed Denial-of-Service

Summary



Hardcore Authorization

- .NET Principals
 - Defines security context under which code is running
 - Create with identity object
- Windows Identities
- Administrators can add themselves but can't access the files or process by default



Windows Identities

- .NET Web Event OnAuthenticate()
 - Gives you an identity passed from IIS
- Local non-web .NET
WindowsIdentity.GetCurrent()
 - Gives you an identity from local login



0121197 2576 0616026

Windows Identities (cont.)

- `IsAnonymous()`
- `IsAuthenticated()`
- `IsGuest()`
- `IsSystem()`



Windows Identities (cont.)

- Demand a user is in a specific group
 - Does NOT have to be “administrators”
- Group name comes from domain or local machine name + group
 - (E.g. BLACKHAT\CustomUserGroup)
- `if(current_principal.IsInRole(“BLACKHAT\CustomUserGroup”) == true)`
- Hackers get administrator access but still can’t access an application without some changes to users and groups if they can figure out why.



Administrators and Trust

- A web server stores credit card information in a database... what does it take for a disgruntled employee to get them all?
- What if he/she was the administrator?
- Who can you really trust?



0121197 2576 0666026

.NET from the Hacker's Perspective: Part II

Authentication Systems

Hardcore Authorization

Cryptography Ignored

Distributed Denial-of-Service

Summary



0121197 2576 0666026

Cryptography Ignored

- Hashing
- Symmetric Key Generation
- Encrypting Server and Client Data
 - Cookie data
 - Credit card numbers
- Crypto Streams



0121197 2576 0666026

Hashing

- Pick your bit length
 - 128 to 512 bits
- Hash everything you can and encrypt what you cannot hash
- Quick demonstration



0121197 2576 0666026

Cryptography Ignored

- Hashing
- **Symmetric Key Generation**
- Encrypting Server and Client Data
 - Cookie data
 - Credit card numbers
- Crypto Streams



Symmetric Key Generation

- Using password hashes as keys
- Administrators can't see data stored in plaintext
- Sensitive data is only decrypted in memory
WHEN the user is logged into the system and
accessing the data
- Quick demonstration



0121197 2576 0666026

Cryptography Ignored

- Hashing
- Symmetric Key Generation
- **Encrypting Server and Client Data**
 - **Cookie data**
 - **Credit card numbers**
- Crypto Streams



Encrypting Server and Client Data

- Data stored on server that is owned by client is encrypted and only the client's secret can help decrypt it
- Data stored on client that is owned by server is encrypted and only the server's secret can help decrypt it
- Always encrypt everything sensitive, always and forever



0121197 2576 0666026

Cryptography Ignored

- Hashing
- Symmetric Key Generation
- Encrypting Server and Client Data
 - Cookie data
 - Credit card numbers
- **Crypto Streams**



Crypto Streams

- Functional modular libraries can give your application a way to maintain encrypted data in memory and on disk with minimal programming requirements.
- A little coding goes a long way!



Crypto Streams (cont.)

- Encrypted Serialization
 - Serialize your class to a memory stream
 - Create ICryptoTransform from symmetric algorithm where key and IV are generated from a password hash
 - Create a file stream with your filename
 - Create a crypto stream from the ICryptoTransform and file stream objects.
 - Read from the memory stream and write to the crypto stream.



0121197 2576 0616026

.NET from the Hacker's Perspective: Part II

Authentication Systems

Hardcore Authorization

Cryptography Ignored

Distributed Denial-of-Service

Summary



Distributed Denial-of-Service

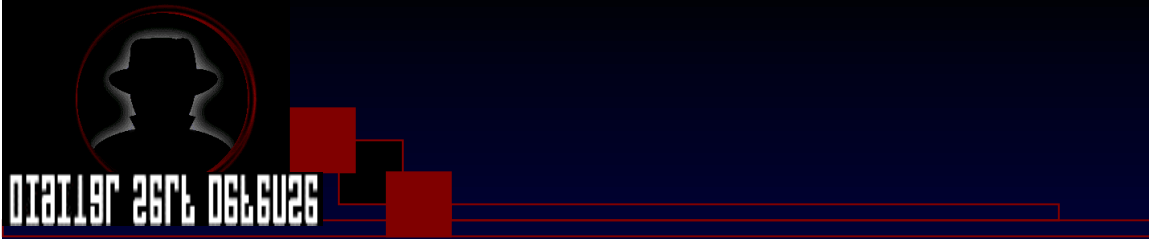
- *.NET and Auto-Sizing Variables*
- Diving in the UI
 - Defaults again!
- Time and Bandwidth
- Monitoring



0121197 2576 0666026

.NET and Auto-Sizing Variables

- May fix buffer overflows
- Now as long as a client can send lots of data it is easier to cause denial-of-service through resource starvation.



Distributed Denial-of-Service

- .NET and Auto-Sizing Variables
- **Diving in the UI**
 - Defaults again!
- Time and Bandwidth
- Monitoring



Diving in the UI

- Textbox
 - (32 KB – 1) default max size input
- Rich Text Box
 - (2 GB – 1) default max size input



Distributed Denial-of-Service

- .NET and Auto-Sizing Variables
- Diving in the UI
 - Defaults again!
- Time and Bandwidth
- Monitoring



Time and Bandwidth

- Assume a web page has 10 fields
- $32 \text{ KB} * 10 \text{ fields} * 64 \text{ sessions} = 20,970,880 \text{ bytes} = 19.99 \text{ mega bytes}$
- Assuming a standard DSL user with 128 kilobits of upload speed. The hacker can have your server allocate ~20mb of data in how many minutes?
- $128 * 1024 = 131,072 \text{ bits per second}$
- $131,072 / 8 = 16384 \text{ bytes per second}$
- $20,970,880 \text{ total data required} / 16384 \text{ data per second} / 60 \text{ seconds} = 21.33 \text{ minutes}$
- $21.33 \text{ minutes} @ 128 \text{ systems attacking} = 2,684,272,640 \text{ bytes allocated}$



Distributed Denial-of-Service

- .NET and Auto-Sizing Variables
- Diving in the UI
 - Defaults again!
- Time and Bandwidth
- **Monitoring**



Monitoring

- The failure or success of any `if()` statement might be a great place to add code which allows the monitoring of a process to determine if attacks are happening.
- Notify people not just log files!
- Record everything!



0121197 2576 0616026

.NET from the Hacker's Perspective: Part II

Authentication Systems

Hardcore Authorization

Cryptography Ignored

Distributed Denial-of-Service

Summary



0121197 2576 0616026

Summary

- The security battle is far from over!
- Use the technologies that you have access to.
They can mitigate a serious amount of risk!



0121197 2576 0616026

Summary (cont.)

- Mitigate what you can!
- Monitor what you can't!
- Never use a technology or process that you cannot monitor or mitigate.



Summary (cont.)

- Force all hackers to brute force encrypted and hashed data near the order of 2^{376} number of attempts to succeed.
- Establish secure tunnels before performing authentication to stop the loss of credentials. It makes no sense to force authentication before you get a secure tunnel.
- Clients are the targets. Force all your remote access personnel to be as secure as your internal servers.



0121197 2576 0616026

.NET from the Hacker's Perspective: Part II

Drew Miller

drewm@blackhat.com