Next Generation .NET Vulnerabilities

Presented by Paul Craig

Security-Assessment.com

Syscan 2007 - Singapore

- Who Am I?
  - Paul Craig, Security Consultant
  - Security-Assessment.com
- My Role
  - Web application penetration tester.
  - "I break web applications"
    - 40 hours a week.
    - 48 weeks a year.
- Based in Auckland, North Island of New Zealand

Contents

- The Basics: .NET a Brief Overview
- The Defense: Security Advancements in .NET
- Common .NET Configuration Mistakes
- Hacking .NET Web Applications
- Hacking the CLR
- New Tool : .NET Reverse Shell Dropper
- Conclusion

The Basics:

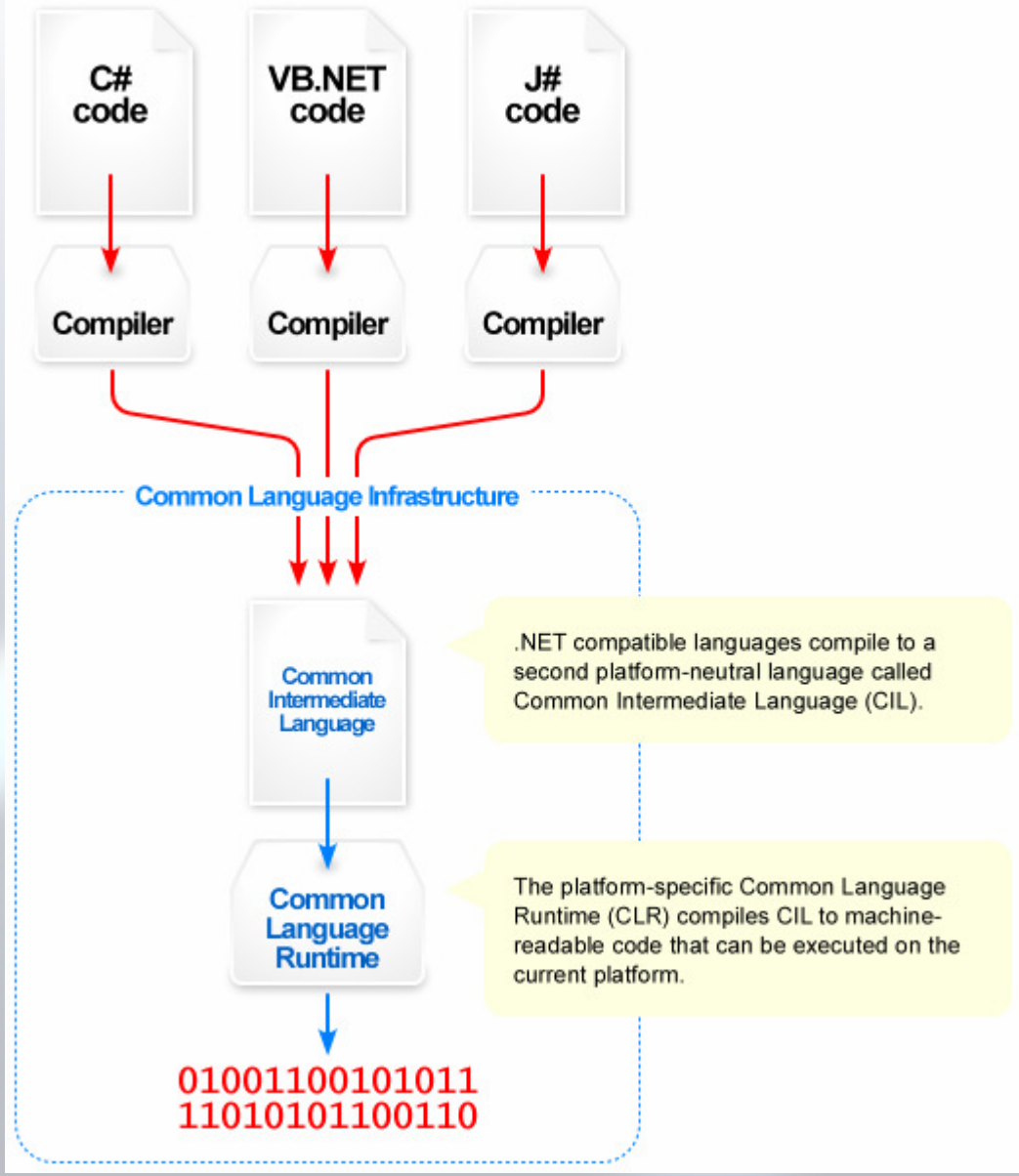
.NET An Overview

- .NET is a large topic
  - This presentation focuses on .NET web applications.

- Target Environment.
  - IIS6/7
  - CLR v1.1, v2.0, v3.0
  - www.company.com/myonlinestore/Login.aspx
  - Outsourced online store.

- .NET was designed to replace all legacy Microsoft development languages.
  - J++, C++, ASP, Visual Basic
- Offers Language Uniformity
  - J#, C#, etc, compiled into one language.
- Common Intermediate Language.
  - CIL concept based on CLI
  - CLI is a standard, not a language (ECMA-335)
  - J#, C#, VB.NET compile into an CIL language.
  - Formally called MSIL (Microsoft Intermediate Language)
  - None interpreted, Just-In-Time (JIT) compiled code.
  - Native byte code for the .NET sandbox, think Java.

- Each Client Has a CLR (Common Language Runtime)
  - Executes CIL, OP codes are ran, program runs.
- Welcome to the Sand Box
  - The CLR is your sandbox.
  - Provides a safe environment for code execution.
  - 'Saving developers from their own mistakes.'
- .NET Is A New Level of Abstraction When Compared to ASP
  - Remote users talk to the CLR, not directly to the script.
  - Suspicious requests are denied.



Security Advancements in .NET

- 1996. Microsoft released ASP v1.0 with IIS 3.0!
  - The most popular Microsoft based web development language.
  - Used in banks, e-commerce, telecommunications, government.
  - Widely exploited development language!
- Most ASP Developers Are Unable To Write Secure Code
  - Language is too versatile and flexible!
  - Allows developers to take shortcuts.
- Security must be implemented by the developer.
  - Good developers write secure ASP web apps.
  - Secure ASP is not easy to write.
  - "Hacked before my coffee is cold"

- .NET Strongly Enforces the Microsoft Mantra "Do it Our Way"
  - Microsoft know that developers write insecure applications.
    - Security by design, by default.
    - By default .NET is secure.
    - "The Microsoft Way" is secure and easy.
    - Hard work to make a .NET application insecure.
    - .NET aggressively implements security.
  - Harder to hack
    - Increased level of security
    - Idiot proof.
    - In Short: My coffee gets cold.

- Request Validation
  - Protects sites from **Cross Site Scripting** (XSS) attacks.
  - CLR validates all user supplied input for XSS attempts.
  - By default .NET protects all GET/POST variables.
    - Test.aspx?value=a Allowed
    - Test.aspx?value=< Allowed
    - Test.aspx?value=<a Denied!

A potentially dangerous Request.QueryString value was detected from the client (name="<script>").

  - So simple, so effective.
    - "Where Have All The XSS Gone?"
  - Must be explicitly disabled to be vulnerable.
  - No XSS without a 0day request validation exploit.

- Path and File Validation.
  - System.IO namespace validate all path characters.
    - System.IO.Path.CheckInvalidPathChars
    - System.Web.Util.FileUtil.IsSuspiciousPhysicalPath
  - Prohibited bytes.
    - 0x01-0x08, 0x0e-0x1f, 0x22, 0x3c, 0x3e, 0x7c
    - BackSpace LineFeed " < > ; |
    - STDOUT redirect attacks.
    - Legitimate filenames never contain > or <
  - Also used in Response.Writefile
  - Suspicious characters raise an exception.

Illegal characters in path.

- Directory Traversal Prevention.
  - Cannot access files outside the document root.
  - Test.aspx?readfile=../../../boot.ini

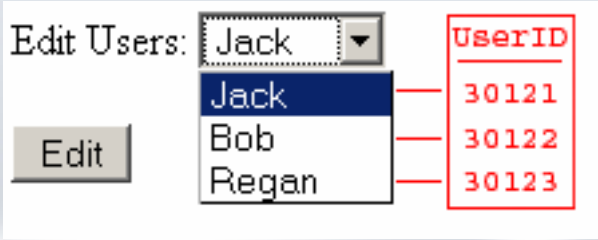
    Cannot use a leading .. to exit above the top directory.
  - Implemented in some (not all) file handling methods.
- Size Validation
  - Fully qualified filename must be less than 260 characters.
  - Directory names must be less than 248 characters.
- Path Expansion Not Supported.
  - %SYSTEMROOT% does not equal C:\windows\system32

- ViewState and EventValidation
  - ASP method of keeping application state:
    - Test.asp?user=1011&m=3&age=25&name=paul...
    - Easy to manipulate application state "I am user 1012"
  - __VIEWSTATE is an application state container.
    - Maintain state between multiple postback requests.
      - "Holds all your variables"
    - Serialized base64 encoded data.
    - Viewstate contents can be encrypted.
  - __EVENTVALIDATION
    - Integrity hash of the Viewstate.
    - Contains the path of the originating .NET page.

- Viewstate and Eventvalidation Really Kill the Fun.
- Exception Created When:
  - Modify a Viewstate variable
    - Viewstate is decoded and modified, re-encoded.
    - Invalidates Eventvalidation hash.
  - POST request missing Viewstate and/or Eventvalidation.
    - You cannot directly POST to an arbitrary .NET page.
    - Postback data must come from another page.
  - Viewstate and Eventvalidation are unique to each page
    - Cannot replay Postback information from another page.
- Manipulating Application State is No Longer Easy!
  - GREATLY reduces the amount of vulnerabilities found.

- Control Postback Validation
  - Protects pre-populated data controls from manipulation.
  - 'Data controls' are uneditable controls
    - ListBox.

    Example:



  - Each UserID value is kept in the Viewstate.
  - User supplied list control compared to Viewstate whitelist.
  - Only whitelist values allowed.
    - UserID = 30121,30122,30123
  - Protects all pre-populated data controls from **any** type of attack.
    - SQL Injection, XSS, Modification

- Cryptography
  - Cryptography is easy in .NET
    - Developers will only implement cryptography when its easy.
    - Developers encouraged to use cryptography.
    - MSDN samples use cryptography
  - "Secrets must be stored securely"
    - HashPasswordForStoringInConfigFile()
  - Strong cryptography now common in web applications.
  - ASP developers rarely implemented cryptography.

- Why no crypto in ASP?
- Generating a SHA1 hash in ASP..
- Yeah, that looks like fun.
- Developers get lazy, no cryptography.

```
Function AndW(ByRef pBytWord1Ary, ByRef pBytWord2Ary)
Dim lBytWordAry(3)
Dim lLngIndex
For lLngIndex = 0 To 3
lBytWordAry(lLngIndex) = CByte(pBytWord1Ary(lLngIndex) And pBytWord2Ary(lLngIndex))
Next
AndW = lBytWordAry
End Function
Function OrW(ByRef pBytWord1Ary, ByRef pBytWord2Ary)
Dim lBytWordAry(3)
Dim lLngIndex
For lLngIndex = 0 To 3
lBytWordAry(lLngIndex) = CByte(pBytWord1Ary(lLngIndex) Or pBytWord2Ary(lLngIndex))
Next
OrW = lBytWordAry
End Function
Function XorW(ByRef pBytWord1Ary, ByRef pBytWord2Ary)
Dim lBytWordAry(3)
Dim lLngIndex
For lLngIndex = 0 To 3
lBytWordAry(lLngIndex) = CByte(pBytWord1Ary(lLngIndex) Xor pBytWord2Ary(lLngIndex))
Next
XorW = lBytWordAry
End Function
Function NotW(ByRef pBytWordAry)
Dim lBytWordAry(3)
Dim lLngIndex
For lLngIndex = 0 To 3
lBytWordAry(lLngIndex) = Not CByte(pBytWordAry(lLngIndex))
Next
NotW = lBytWordAry
End Function
Function AddW(ByRef pBytWord1Ary, ByRef pBytWord2Ary)
Dim lLngIndex
Dim lIntTotal
Dim lBytWordAry(3)
For lLngIndex = 3 To 0 Step -1
If lLngIndex = 3 Then
lIntTotal = CInt(pBytWord1Ary(lLngIndex)) + pBytWord2Ary(lLngIndex)
lBytWordAry(lLngIndex) = lIntTotal Mod 256
Else
lIntTotal = CInt(pBytWord1Ary(lLngIndex)) + pBytWord2Ary(lLngIndex) + (lIntTotal \ 256)
lBytWordAry(lLngIndex) = lIntTotal Mod 256
End If
Next
AddW = lBytWordAry
End Function
Function CircShiftLeftW(ByRef pBytWordAry, ByRef pLngShift)
Dim lDbl1
Dim lDbl2
lDbl1 = WordToDouble(pBytWordAry)
lDbl2 = lDbl1
lDbl1 = CDbl(lDbl1 * (2 ^ pLngShift))
lDbl2 = CDbl(lDbl2 / (2 ^ (32 - pLngShift)))
CircShiftLeftW = OrW(DoubleToWord(lDbl1), DoubleToWord(lDbl2))
End Function
Private Function MD5WordToHex(lValue)
Dim lByte
Dim lCount
For lCount = 0 To 3
lByte = RShift(lValue, lCount * BITS_TO_A_BYTE) And m_lOnBits(BITS_TO_A_BYTE - 1)
MD5WordToHex = MD5WordToHex & Right("0" & Hex(lByte), 2)
DoubleToWord = lBytWordAry
End Function
Function WordToDouble(ByRef pBytWordAry)
WordToDouble = CDbl((pBytWordAry(0) * (2 ^ 24)) + (pBytWordAry(1) * (2 ^ 16)) +
(pBytWordAry(2) * (2 ^ 8)) + pBytWordAry(3))
End Function
Function DMod(ByRef pDblValue, ByRef pDblDivisor)
Dim lDblMod
lDblMod = CDbl(CDbl(pDblValue) - (Int(CDbl(pDblValue) / CDbl(pDblDivisor)) *
CDbl(pDblDivisor)))
If lDblMod < 0 Then
lDblMod = CDbl(lDblMod + pDblDivisor)
End If
DMod = lDblMod
End Function
Function F( ByRef lIntT, ByRef pBytWordBAry, ByRef pBytWordCAry, ByRef pBytWordDAry)
If lIntT <= 19 Then
F = OrW(AndW(pBytWordBAry, pBytWordCAry), AndW((NotW(pBytWordBAry)),
pBytWordDAry))
ElseIf lIntT <= 39 Then
F = XorW(XorW(pBytWordBAry, pBytWordCAry), pBytWordDAry)
ElseIf lIntT <= 59 Then
F = OrW(OrW(AndW(pBytWordBAry, pBytWordCAry), AndW(pBytWordBAry,
pBytWordDAry)), AndW(pBytWordCAry, pBytWordDAry))
Else
F = XorW(XorW(pBytWordBAry, pBytWordCAry), pBytWordDAry)
End If
End Function
Function SHA1(pStrMessage)
Dim lLngLen, lBytLenW, lLngTempWordWAry, lLngNumBlocks, lLngBlock, lIntT,
lBytTempAry
Dim lVarWordWAry(79), lVarWordKAry(3)
Dim lStrBlockText, lStrWordText, lStrPadMessage
Dim lBytWordH0Ary, lBytWordH1Ary, lBytWordH2Ary, lBytWordH3Ary, lBytWordH4Ary
Dim lBytWordAAry, lBytWordBAry, lBytWordCAry, lBytWordDAry, lBytWordEAry,
lBytWordFAry
lLngLen = Len(cstr(pStrMessage))
lBytLenW = DoubleToWord(CDbl(lLngLen) * 8)
lStrPadMessage = pStrMessage & Chr(128) & String((128 - (lLngLen Mod 64) - 9) Mod 64,
Chr(0)) & String(4, Chr(0)) & Chr(lBytLenW(0)) & Chr(lBytLenW(1)) & Chr(lBytLenW(2)) &
Chr(lBytLenW(3))
lLngNumBlocks = Len(lStrPadMessage) / 64
lVarWordKAry(0) = HexToWord("5A827999")
lVarWordKAry(1) = HexToWord("6ED9EBA1")
lVarWordKAry(2) = HexToWord("8F1BBCDC")
lVarWordKAry(3) = HexToWord("CA62C1D6")
lBytWordH0Ary = HexToWord("67452301")
lBytWordH1Ary = HexToWord("EFCDAB89")
lBytWordH2Ary = HexToWord("98BADCFE")
lBytWordH3Ary = HexToWord("10325476")
lBytWordH4Ary = HexToWord("C3D2E1F0")
For lLngBlock = 0 To lLngNumBlocks - 1
lStrBlockText = Mid(lStrPadMessage, (lLngBlock * 64) + 1, 64)
For lIntT = 0 To 15
lStrWordText = Mid(lStrBlockText, (lIntT * 4) + 1, 4)
lVarWordWAry(lIntT) = Array(Asc(Mid(lStrWordText, 1, 1)), Asc(Mid(lStrWordText, 2, 1)),
Asc(Mid(lStrWordText, 3, 1)), Asc(Mid(lStrWordText, 4, 1)))Next
For lIntT = 16 To 79 lVarWordWAry(lIntT) =
CircShiftLeftW(XorW(XorW(XorW(lVarWordWAry(lIntT - 3), lVarWordWAry(lIntT - 8)),
lVarWordWAry(lIntT - 14)), lVarWordWAry(lIntT - 16)), 1) Next
lBytWordAAry = lBytWordH0Ary
lBytWordBAry = lBytWordH1Ary
lBytWordCAry = lBytWordH2Ary
lBytWordDAry = lBytWordH3Ary
lBytWordEAry = lBytWordH4Ary
For lIntT = 0 To 79
lBytWordFAry = F(lIntT, lBytWordBAry,  lBytWordCAry, lBytWordDAry)
lBytTempAry = AddW(AddW(AddW(AddW(CircShiftLeftW(lBytWordAAry, 5),
lBytWordFAry), lBytWordEAry), lVarWordWAry(lIntT)), lVarWordKAry(lIntT \ 20))
lBytWordEAry = lBytWordDAry
lBytWordDAry = lBytWordCAry
lBytWordCAry = CircShiftLeftW(lBytWordBAry, 30)
lBytWordBAry = lBytWordAAry
lBytWordAAry = lBytTempAry
Next
lBytWordH0Ary = AddW(lBytWordH0Ary, lBytWordAAry)
lBytWordH1Ary = AddW(lBytWordH1Ary, lBytWordBAry)
lBytWordH2Ary = AddW(lBytWordH2Ary, lBytWordCAry)
lBytWordH3Ary = AddW(lBytWordH3Ary, lBytWordDAry)
lBytWordH4Ary = AddW(lBytWordH4Ary, lBytWordEAry)
Next
SHA1 = WordToHex(lBytWordH0Ary) & WordToHex(lBytWordH1Ary) &
WordToHex(lBytWordH2Ary) &  WordToHex(lBytWordH3Ary) &
WordToHex(lBytWordH4Ary)
End Function
```

- Generating a SHA1 hash in C#

```
byte[] input = new byte[100];
byte[] output;
SHA1CryptoServiceProvider sha;
sha = new SHA1CryptoServiceProvider();
output = sha.ComputeHash(input);
```

- Developers are lazy!
  - Only easy solutions are adopted!
  - 5 lines of C#.
- Microsoft encourage developers to write secure code.
  - Security is now easy.

- SQL Injection
  - ASP developers frequently used dynamic SQL
  - SQL generated on the fly from user supplied values.
  - Countless SQL injection vulnerabilities:1,440,000 Google results
  - Microsoft new approach "All Input is evil”
  - Parameterized SQL queries recommended.

```
obj.Parameters.Add("@Name", NameTextBox.Text);
obj.Parameters.Add("@Password", PasswordTextBox.Text);
```

  - Trust the framework.

- So Where's the Beef?
  - CLR has a huge impact on security.
  - Fewer attack vectors!
- Classic ASP Vulnerabilities, Gone!
  - Postback variable manipulation
  - Cross Site Scripting
  - Directory traversal
  - Path/File manipulation
  - Attacking application state
- Is my job in danger?
  - What does a penetration tester do, when the apps are secure!

- "There are Always Vulnerabilities"
  - Developers **always** make mistakes.
    - It all comes down to one person on a keyboard.
    - Developer has a bad week, makes mistakes.
  - Its all just so new.
    - Millions of C++, ASP, VB developers now migrating to C#
    - "Most .NET developers I interview are completely uneducated about how to correctly code .NET"
    - Poorly hacking C# together like as if they were coding ASP.
  - .NET is only secure when used correctly.
    - Doing things 'your way' is usually insecure.

Common .NET Configuration Mistakes

Common .NET Configuration Mistakes

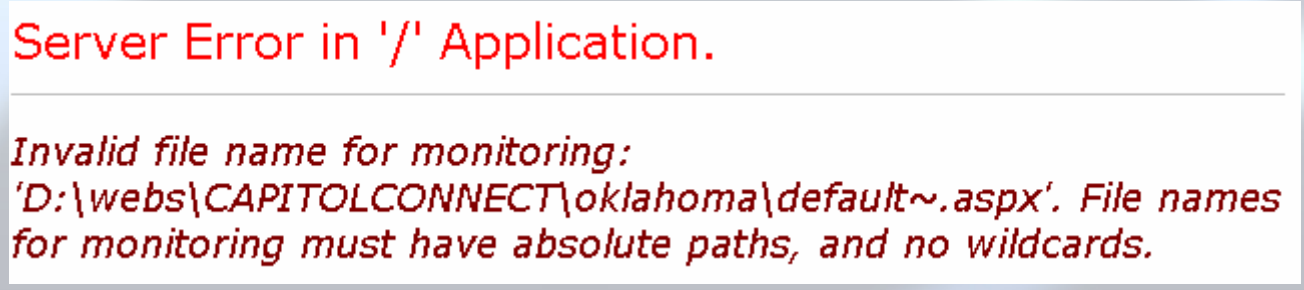
- Server Deployment Scenario.
  - 'Build Team' builds an IIS web-server for new website.
  - Developers deploy the web application.
- OPS Team Responsible for the Security of the Server.
  - Painstaking BEST security practices implemented.
  - System administrators understand security requirements.
- Developers Responsible for the Web Application
  - Does not really understand security principles.
  - "I just write code"
- Who is Responsible for the .NET Configuration File?
  - Developer Said: "OPS, it's an application configuration. "
  - OPS Guy said: "Developers, it's a code thing!"
  - Config's are poorly maintained, modified ad-hoc by developers.

- Error Messages and Stack Traces
  - .NET raised exceptions contain a stack trace.
  - By default stack traces are not shown to remote hosts.
  - First thing a developer enables, 90% of the time enabled.
    - Contains the object call stack.

```
[HttpException (0x80004005): Cannot use a leading .. to exit above the top directory.]
   System.Web.Util.UrlPath.ReduceVirtualPath(String path) +3527398
   System.Web.Util.UrlPath.Reduce(String path) +84
   System.Web.Util.UrlPath.Combine(String appPath, String basepath, String relative) +355
```

  - Error messages can also contain.
    - Physical Path
    - .NET CLR version number

- Determine if Stack Traces are Enabled.
  - http://www.host.com/default|.aspx
  - System.IO.Path.CheckInvalidPathChars Exception
  - Guaranteed to create an exception.
- Physical Path Disclosure
  - Invalidate .NET Monitoring with a virtual directory identifier.
  - http://www.host.com/default~.aspx
    - Only absolute paths allowed.



  - Only 'default' .NET error message to disclose physical path!

- Trace Information
  - .NET supports detailed application level trace information.
  - http://www.host.com/trace.axd
  - Contains the last 10 requests to the server.
    - Including POST, GET, State, Cookies, Headers
    - Easily compromise other user accounts or sessions.
  - Most fatal mistake to make in .NET configuration.
  - Highest severity, game over!
  - Only seen remotely accessible tracing enabled once!

- Storing Sensitive Values In Web.config.
  - <Appsettings>
    - Database connection strings.
    - Web application administrator password.
  - NET promotes a secure password hashing method.
    - HashPasswordForStoringInConfigFile
  - Developers not encouraged to salt hashes.
    - MSDN does not mention salts.
    - A lot of developers copy/paste from MSDN.
  - Never seen a salted hash in use!
  - Vulnerable to memory-trade-off attacks.
    - Rainbow Tables.

Hacking .NET Applications

"I Used To See Web Applications With Flaws.

But Now I See Just See The Flaws."

- All developers make mistakes.
- Most developers make the same mistakes!
- 10 developers read the same "Learn C# in 24-hours book"
  - Similar misconceptions and gaps of knowledge.
- Same Bug Different Developer

- Examples of Common Developer Mistakes.
  - Following scenarios are real!
  - Critical applications in large organizations.
  - Originally developed by 100% authentic developers.
  - Common mistakes, not how you should write .NET

- Attacking The PostBack
  - This.IsPostBack()
    - Method to identify a request as PostBack or GET.
    - Postbacks need a valid Eventvalidation/Viewstate.
    - Developers have more trust for PostBack requests.
      - "We must be in the middle of something, I trust you"
- The Scenario.
  - Administrative section.
  - Developer assumed all Postback requests came from authenticated users.
  - Admin credentials not revalidated.
  - If IsPostBack() = true will skip authorisation requirements.
  - POST requires valid EventValidation/Viewstate.

- The Script.

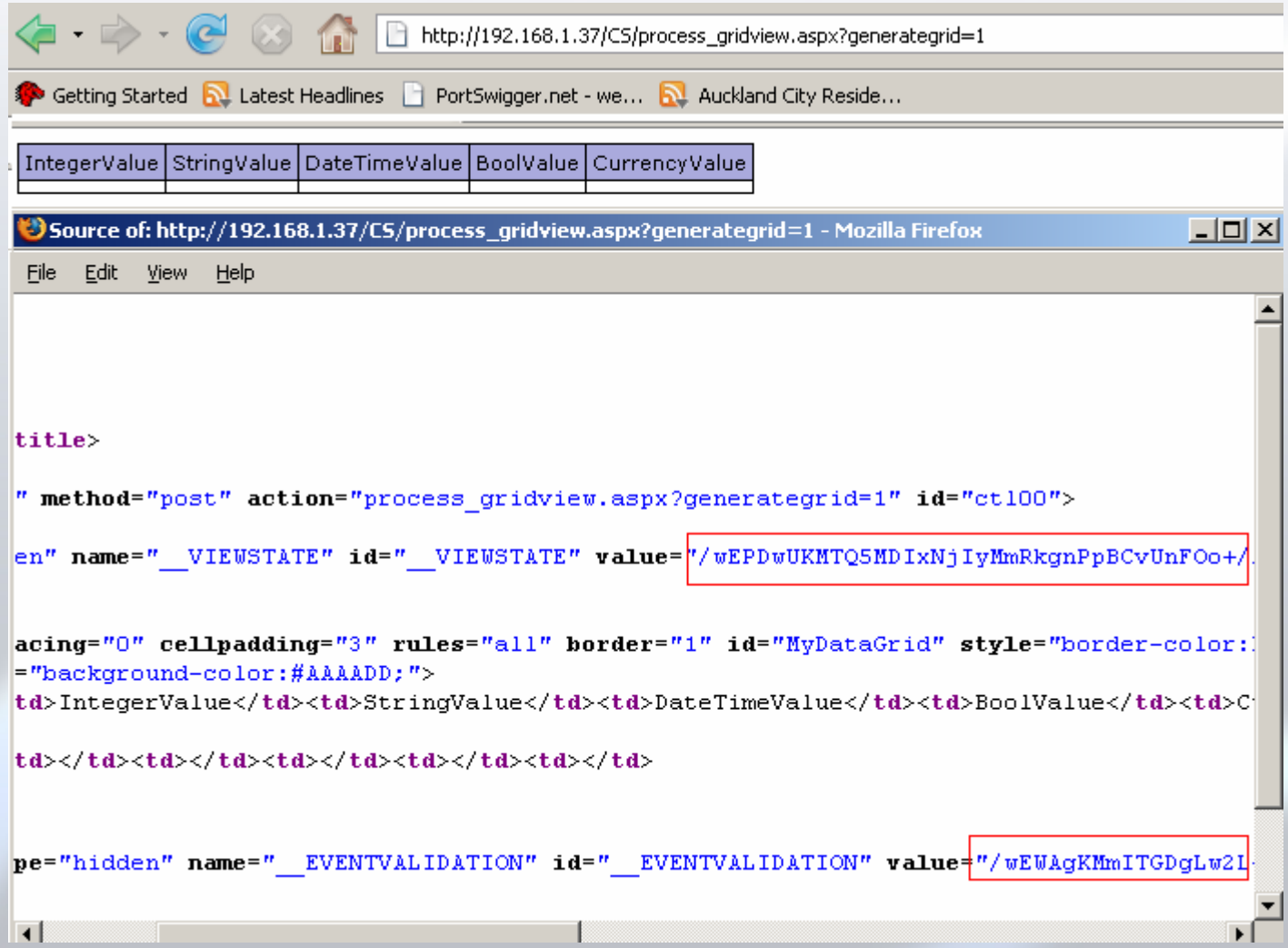
  Process_gridview.aspx

  - Responsible for updating a grid control with current sales figures.
  - Accepts both GET and POST requests.
    - GET /admin/process_gridview.aspx (“Please Login”)
    - GET /admin/process_gridview.aspx?generategrid=1
      - Ajax function, renders an empty grid.
    - We want to POST “command=showsalesresults”
    - Need a Viewstate/Eventvalidation!
- Remember.
  - Eventvalidation is unique to each page.
  - Eventvalidation hash must be valid but Viewstate contents are irrelevant.

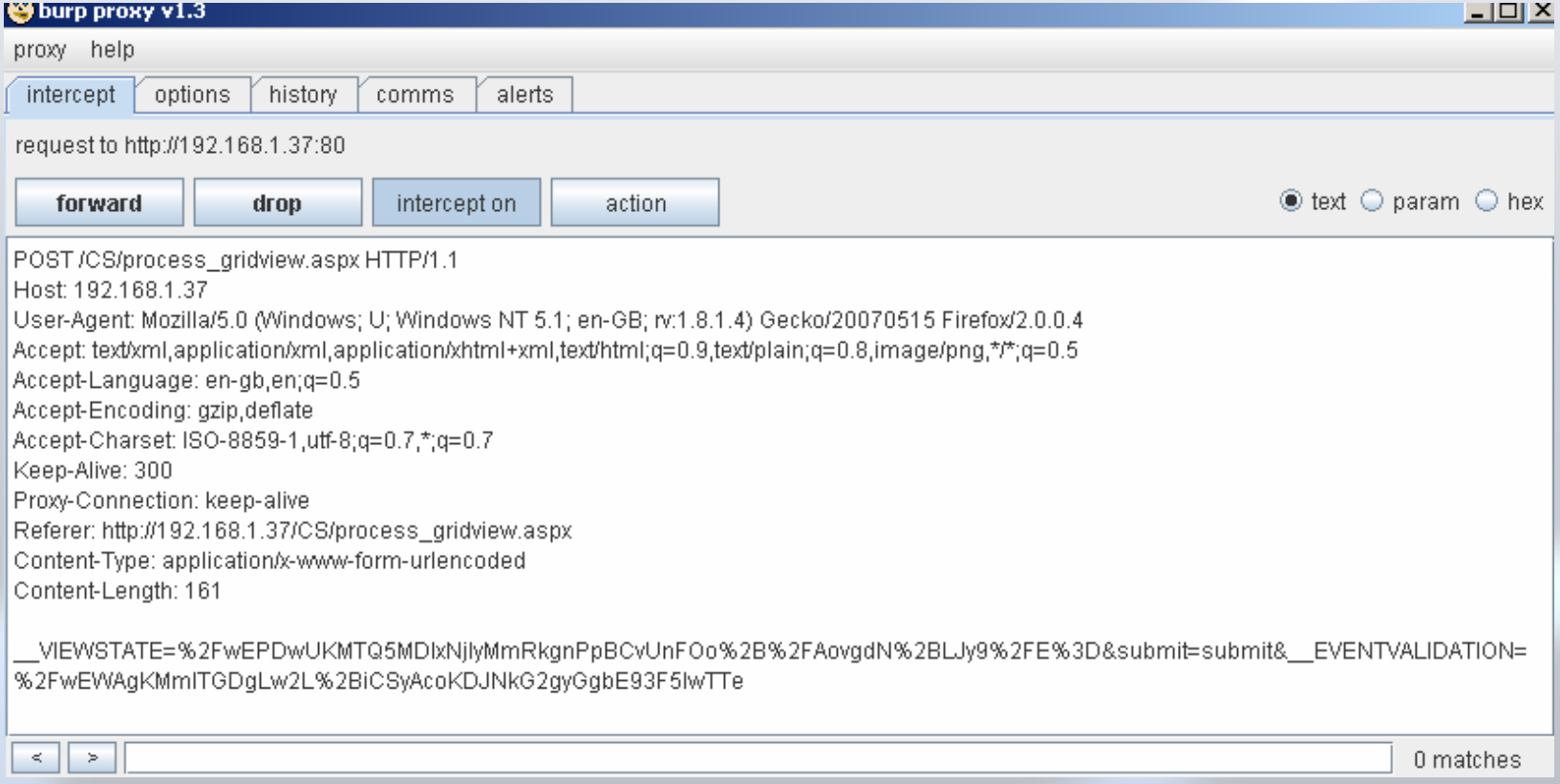
- !This.IsPostBack.

```
if(!this.IsPostBack) {
        // someone requested a GET of the page
        Response.Write("You cannot access this page directly, please login first.");

}

if(this.IsPostBack) {
        // POST data, with valid eventvalidation/viewstate, we must know each other
        Response.Write("What do you want to do today?");

}
```

- We cannot POST.
  - But we can GET.
    - AJAX function generategrid=1 output contains a new form with an empty grid control.
    - Containing a valid __VIEWSTATE and __EVENTVALIDATION.
    - Never gets sent back to the server (No submit button)
    - Viewstate is empty and contains nothing!



- URL Encode Viewstate/Eventvalidaiton
- Replay to the server.
  - This.IsPostBack = True

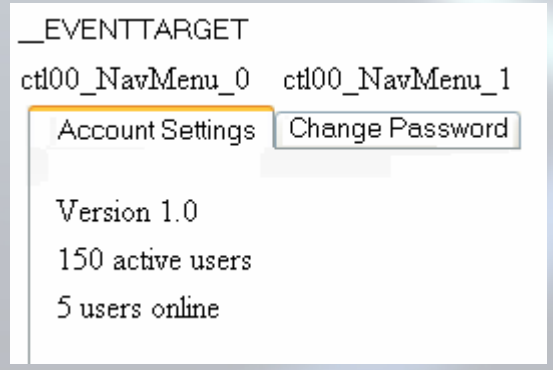


- POST command=showsalesresults

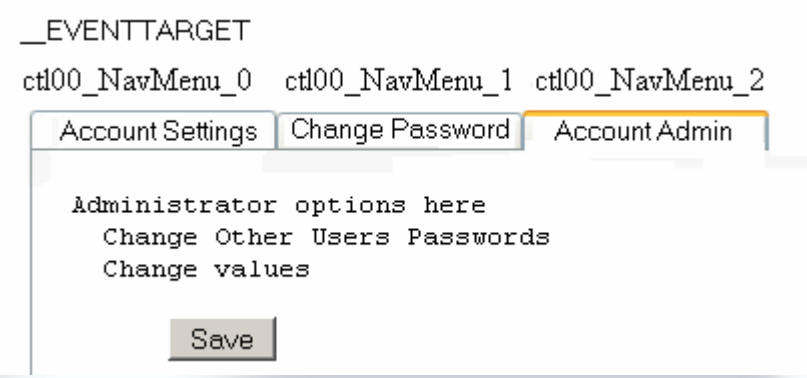
- Exploiting.
  - Exploitation requires the target to implicitly trust all PostBack requests.
  - Postback information carries as much trust as credentials!
  - Found in pages which do not directly accept requests themselves.
    - Server.Transfer from another page.
  - Need to render a server-side form via a GET request.
    - Page toggles (?rendernav=1, ?displayad=true)
    - GET based AJAX functions.
  - Find another way to generate a valid Viewstate/Eventvalidation value for a page.

- Control Accessibility Vulnerabilities
  - .NET forms can contain multiple uniquely accessible controls.
    - Used by client side Postback requests.
    - Each control has a specific control ID.
    - CLR event handlers process events for controls (OnClick).
  - Controls are accessed through the __EVENTTARGET variable.
  - Directly post to controls name.
    - &ButtonX=Submit
  - Only visible controls are accessible.

- The Scenario
  - AccountOptions.aspx
  - Page allowing authenticated users to change account settings.
    - Content divided between two tabs.
      - Account Settings and Change password.
    - Clicking each tab generates a PostBack request.
      - POST __EVENTTARGET ctl00_NavMenu_0
      - OnClick event handlers generate the tab content.
      - 2 control ID's: Ctl00_NavMenu_0, ctl00_NavMenu_1



- First Rule of Penetration Testing: Change Everything
  - What about ctl00_NavMenu_2?
    - __EVENTTARGET = ctl00_NavMenu_2



  - Access administrative content event handler.
  - Developer hid the control by not displaying the Admin tab.
  - All the frame controls were accessible.
  - One was not 'clickable'
  - You just had to know the right control ID value.

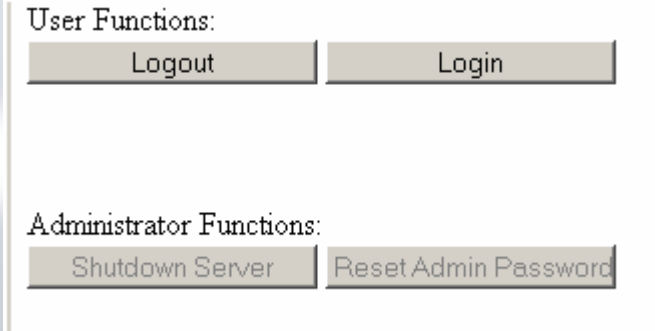
- Disabled Controls.
  - What's wrong with this logic?

```
if ((string)Session[“IsAdministrator"] == "yes"))
    Control1.Enabled = true;
} else {
    Control1.Enabled = false;
}
```

  - If your not an Administrator, then you don’t get access to the control.
    - Its disabled.

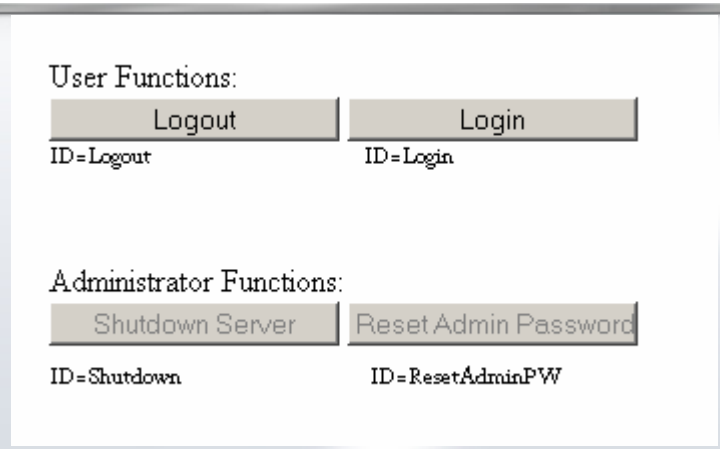
- Maybe not.
  - You cannot click a disabled control, but you can access it.
  - User | Administrator privilege separation.

```
if ((string)Session["IsAdmin"] == "yes") {
        Shutdown.Enabled = true;
        ResetAdminPW.Enabled = true;
} else {
        Shutdown.Enabled = false;
        ResetAdminPW.Enabled = false;
}
```

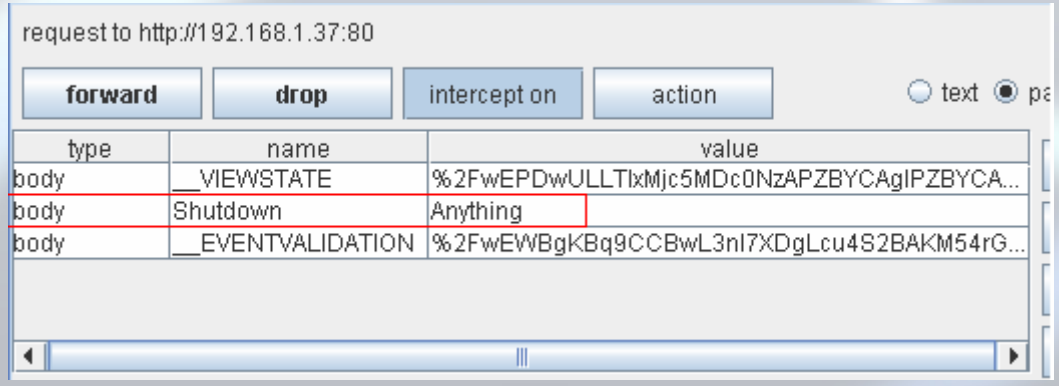


  - **Security by User Interface:**
    - Removing the graphical pathway to a control.
    - You cant 'click' a disabled control.
      - Found in text boxes, gridviews, custom controls.
    - Disabled buttons? Please, is this 1994?

- How to Click a Disabled Button



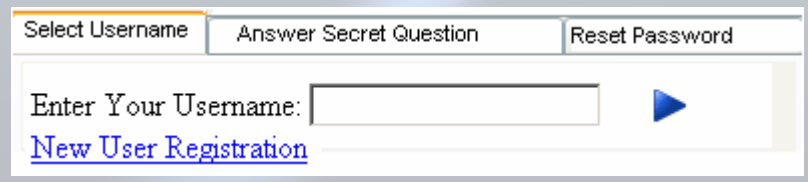
  - Click an enabled button, create a Postback.
  - Capture the request in a web proxy (Web Scarab, Burp Proxy)
  - Add a new POST variable named the ID of the control
    - It can have any value



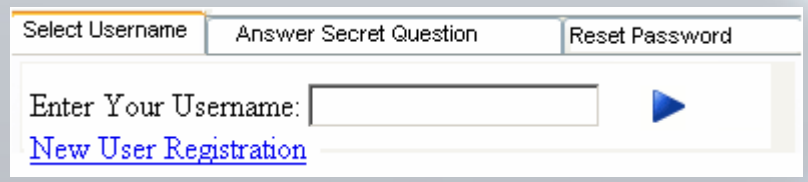
  - Change __EVENTTARGET if its in use.

- Control accessibility is based on control VISIBLITY! (.Visible)
  - Only .Visible = true controls can be accessed!
- 'Why Do Developers Keep Making This Dumb Mistake'?
  - Disable: To make unable or unfit; weaken or destroy the capability of; cripple; incapacitate.
  - Visible: That can be seen; perceptible to the eye.
- If I Wanted To Remove A Control From A Page Would I:
  - Incapacitate it, or hide it from sight?
  - Security teaches us to incapacitate and disable, not hide.
- Disable is a stronger term!
  - Disabled controls are inaccessible in other languages.
  - Developers trust the framework. "But its disabled.."

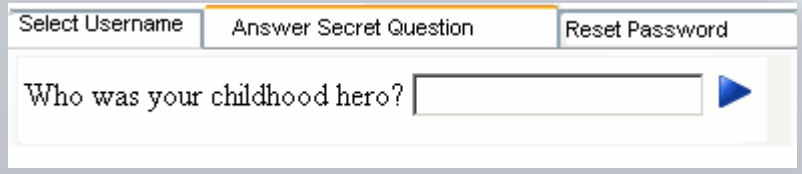
- Control Sequence Attacks
  - PostBack & Viewstate maintain state across multiple pages.
    - Multiple requests to multiple unique pages.
    - You cannot spoof a POST request to an arbitrary page.
  - What about control sequence attacks?
    - Multiple controls on one page.
    - Each control acting as a separate page?
- Scenario: ForgottenPassword.aspx
  - One page with three label controls accessible via a TabStrip.
    - 1: Select Username
    - 2: Answer Secret Question
    - 3: Reset Your Password



- Step 1:
  - Select Username control: Ctl00_tabstrip_00
  - Has a link to a "New User Registration" page.
  - Takes user input NameTextBox.Text
  - Sets a local Viewstate variable of whatever you enter.
    - ViewState.Add("Username",NameTextBox.Text)
  - Posts back information to second ASP:Panel control.
    - "Answer Secret Question"

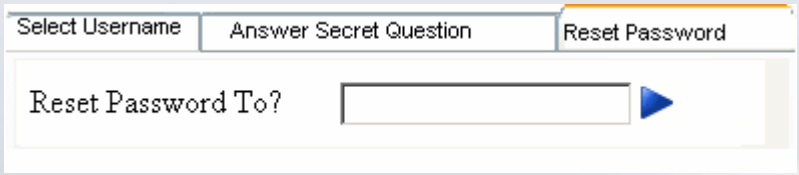


- Step 2:
  - Answer Secret Question: ctl00_TabStrip_01
    - Secret question is returned from Viewstate[“username”]
  - Form allows user input QuestionAnswer.text input box.
    - Postback answer to self and validates input.
  - Secret Question Answer Correct??
    - Sets another Viewstate variable.
      - ViewState.Add(“KnowsAnswer",”1”)
      - User knows the secret question answer.



    - Postback to third control ctl00_TabStrip_2.

- Step 3:
  - Reset Password: ctl00_TabStrip_02
    - Validates the previous 2 steps occurred.

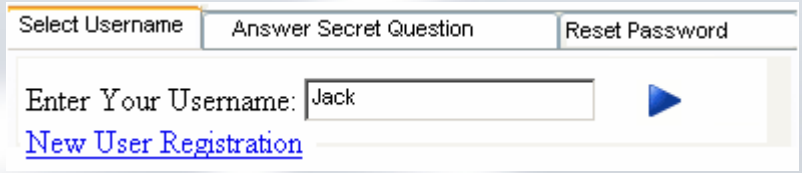



```
if (ViewState["username"] == null) {
        // send user back to first tab
}

if(ViewState["KnowsAnswer"] == null) {
        // send user back to seccond tab
}

// So you know the answer, and you have a username
// What do you want the password to be?
ChangeUserPassword(ViewState["username"], Password.text);
```

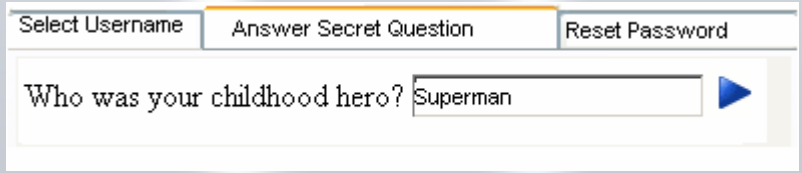
    - You must have two Viewstate variables set.
      - Username
      - KnowsAnswer
  - This application is highly exploitable, can you see how?

- Exploiting.
  - Goal: Change the "Administrator" password.
  - No EVENTVALIDATION in use?
    - Decode, modify, recode Viewstate, we win.
    - Very unlikely situation.
  - Viewstate Encryption or Validation?
    - Must use application logic to manipulate Viewstate.
    - Each tab is a separate, enabled and visible control.
    - We can post to each control at any time.
  - Controls validate the previous step occurred.
    - Skipping forward tabs does not work.

- First we register a new user “Jack”
  - His childhood hero is Superman, and his password is “password”
- Forgotten Password.
- Viewstate username = Jack



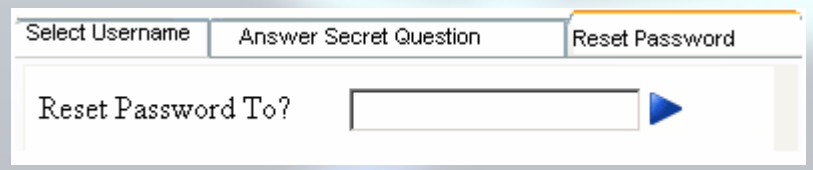
- Viewstate knowsAnswer = 1



- We end up here.

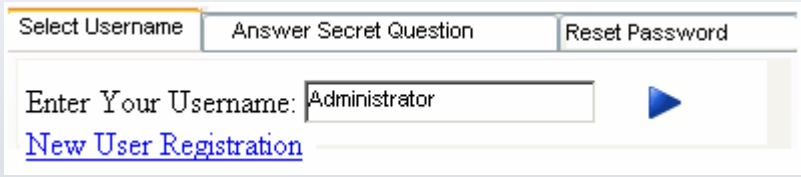
  Click arrow and modify request.

  __EVENTTARGET = ctl00_TabStrip_00



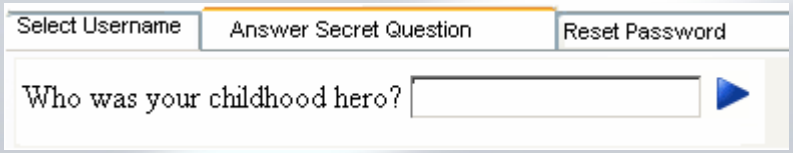
- Actually my username is Administrator

  Click blue arrow.



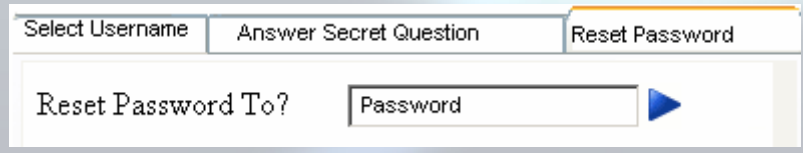
- Skip this stage, Viewstate 'knowsAnswer' is equal to 1 already!

  Post, modify __EVENTTARGET to ctl00_TabStrip_02



- Reset my (Administrator) password to Password please.

```
if(viewstate["username"] == null) {
        //send user back to first tab
}


if(viewstate["KnowsAnswer"] == null) {
//send user back to seccond tab
}

// So you know the answer, and have a username
ChangeUserPassword(Administrator, password)
```



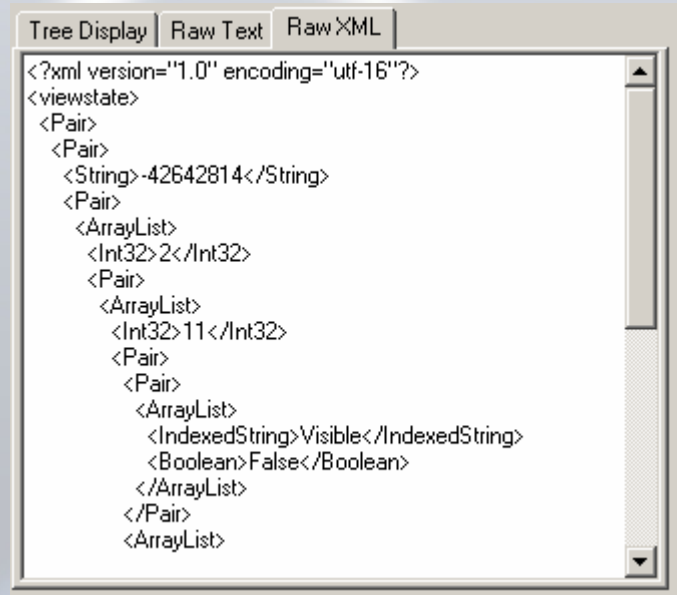
- Control Sequence Attacks
  - Multiple controls on one page, acting as separate pages.
  - Each step in the process is a separate control.
    - ASP:Panel
- Controls can be accessed out of sequence.
  - Access any control ID
  - Skip forwards, backwards.
  - Same page for all steps.
    - We can post to any step and any control!
    - Any visible control is accessible.
- Eventvalidation would stop us skipping forward pages.
  - Cannot generate valid Postback information arbitrarily.

- The Golden Rule Of Web Security
  - "Do not keep anything sensitive inside the document root"
  - Web.config is **most important and sensitive file in .NET.**
  - A normal file inside the document root.
    - Containing usernames, passwords, connection strings.
    - MachineKey and DecryptionKey.
  - Hack a .NET App? Get the Web.config.
    - Decrypt and encrypt Viewstate
    - Regenerate Eventvalidation
  - Completely compromise application state.
  - One file disclosure vulnerability.

Hacking the CLR

- Attacking the Viewstate
  - Viewstate is kept in a serialized base64 encoded format.
    - "1 in 5 audited applications encrypted Viewstate".
  - __VIEWSTATE variable contains application state.
    - Application variables and control attributes.
  - Viewstate contents can be decoded using a ViewState Decoder
    - http://www.pluralsight.com/tools.aspx
  - Can rarely modify Viewstate contents.
    - __EVENTVALIDATION must be explicitly disabled
    - It never is!
  - Typically we can only observe the Viewstate and its contents.

- Determine the Presence of an Invisible Control.
  - __VIEWSTATE maintains current control visibility.
  - Postback validation uses the Viewstate to determine visible controls.
  - Control's not rendered on page are kept inside the Viewstate.
  - Decoded Viewstate from a form with a hidden asp:Panel
  - Contains the controls visibility state.
    - "Visible = False"
  - Page contains an invisible control.



- Display Values of Pre-Populated Invisible Disabled Controls
  - Take an input box “passwrd”, give it a value “AdminPassword”
  - User is not an admin? Hide the box. (Invisible & Disabled)

```
<%@ Page Language="C#" AutoEventWireup="true"%>
<script runat="server">

     protected void Page_Load(object sender, EventArgs e)
    {
        passwrd.Value = "AdminPassword";

        if ((string)Session["IsAdmin"] == "yes")
        {
                passwrd.Visible = true;
                passwrd.Disabled = false;
        } else {
                passwrd.Visible = false;
                passwrd.Disabled = true;
        }
}
```

```
<String>1713148355</String>
<Pair>
 <ArrayList>
  <Int32>3</Int32>
  <Pair>
   <ArrayList>
    <Int32>1</Int32>
    <Pair>
     <ArrayList>
      <IndexedString>value</IndexedString>
      <String>AdminPassword</String>
      <IndexedString>disabled</IndexedString>
      <String>disabled</String>
      <IndexedString>Visible</IndexedString>
      <Boolean>False</Boolean>
     </ArrayList>
    </Pair>
   </ArrayList>
  </Pair>
```

- Passwrd.value is Kept Inside the Viewstate.
  - Even though the control is **disabled**, and **invisible!**
- Viewstate is working correctly.
  - .Value is an attribute of a control.

- More Invisible Disabled Pre-Populated Controls.
  - Gridviews : Show_sales_figures.aspx
    - A Gridview control is pre-populated with values on Page_Load
  - Checks the remote users authorisation.
    - Not in sales department?
    - Disable, hide the MyDataGrid ASP:Panel 'ThePanel'
    - Gridview is inside MyDataGrid and also disabled/invisible.
  - The complete Gridview contents are kept in the Viewstate.

```
Sub Page_Load(sender As Object, e As EventArgs)

    MyDataGrid.DataSource = CreateDataSource()
    MyDataGrid.DataBind
    if Loggedin = "yes" then
            Response.Write("Welcome")
     else
            ThePanel.Visible=false
            ThePanel.Enabled=false
    end if

End Sub
```

```
    <ArrayList>
      <IndexedString>Text</IndexedString>
      <String>Item 6</String>
    </ArrayList>
  </Pair>
</Pair>
<Int32>2</Int32>
<Pair>
  <Pair>
    <ArrayList>
      <IndexedString>Text</IndexedString>
      <String>11/06/2007 2:33:00
```

- NULL Bytes
  - Standard C POSIX systems terminate all strings with a NULL byte (\x00, %00).
    - \x31\x32\x33\x00\x11\x11.
    - Strings are terminated at the first NULL byte.
    - \x11\x11 does not exist.
  - Certain sections of the .NET namespace allow NULL byte input.
    - NULL bytes are just considered data.
    - .NET strings are not terminated by a NULL byte.
  - Interoperability issues when .NET uses native C API calls.
    - .NET string is not terminated at the first NULL
    - Windows C API call terminates string at first NULL byte

- String Termination Attacks
  - Vulnerable methods allow NULL bytes in user supplied input.
  - User supplied data is used in a standard C function call.
    - Somewhere within the .NET method.

    Object.Method(String.Concat(Request.QueryString["a"],"XX"));

    Request.QueryString["a"] = "12345%00"

    value = Object.Method("12345<NULL>XX")

  - String goes into the method: 12345<NULL>XX
  - String comes out (variable value) 12345

  - "XX" was removed from the end of the string.

- MyFictitiousFileUploader
  - Allows developers to easily add file upload functionality.
    - Malicious characters prohibited (../)
    - NULL bytes allowed.
    - .uploaded is appended to all filenames uploaded.

```
MyFileUploader.FileName(filename.Text & ".uploaded");
MyFileUploader.ProcessFile();
```

  - Evil.aspx will become not so evil Evil.aspx.uploaded
    - We supply the filename Evil.aspx%00
    - ProcessFile() uses a custom string validation routine.
    - Which uses a direct strncpy call.
    - String termination vulnerability.
    - Filename of Evil.aspx%00 becomes Evil.aspx
    - Not .uploaded.

- Zero Day!
  - NULL byte injection attacks inside the .NET framework!
  - Multiple different attack vectors.
  - "Released at Syscan '07"
    - Patches were originally planned for May.
      - Then June..
        - Now July!
  - **Responsible** vulnerability disclosure.
    - No patch, no disclosure.
  - Advisories coming soon.
    - Interested in .NET hacking and .NET vulnerabilities?
    - http://www.security-assessment.com/

- Session Fixation
  - Forcing a user to authenticate to a known session ID value, then stealing the users session.
  - .NET Session ID's are vulnerable to session fixation.
  - Exploitable in cookieless session configuration.
    - Session ID is kept in the URI, not in the cookie.
  - http://host.com/(0f3cxnzjskr2ltaiabj2oqp2)/Login.aspx
    - 0f3cxnzjskr2ltaiabj2oqp2 = Encoded random number
    - Session ID is not part of an algorithm or sequence of numbers.
  - .NET does not validate the legitimacy of the session ID.
    - Any 24 character string works.
    - testtesttesttesttesttest is a valid session ID.

- (0f3cxnzjskr2ltaiabj2oqp2)

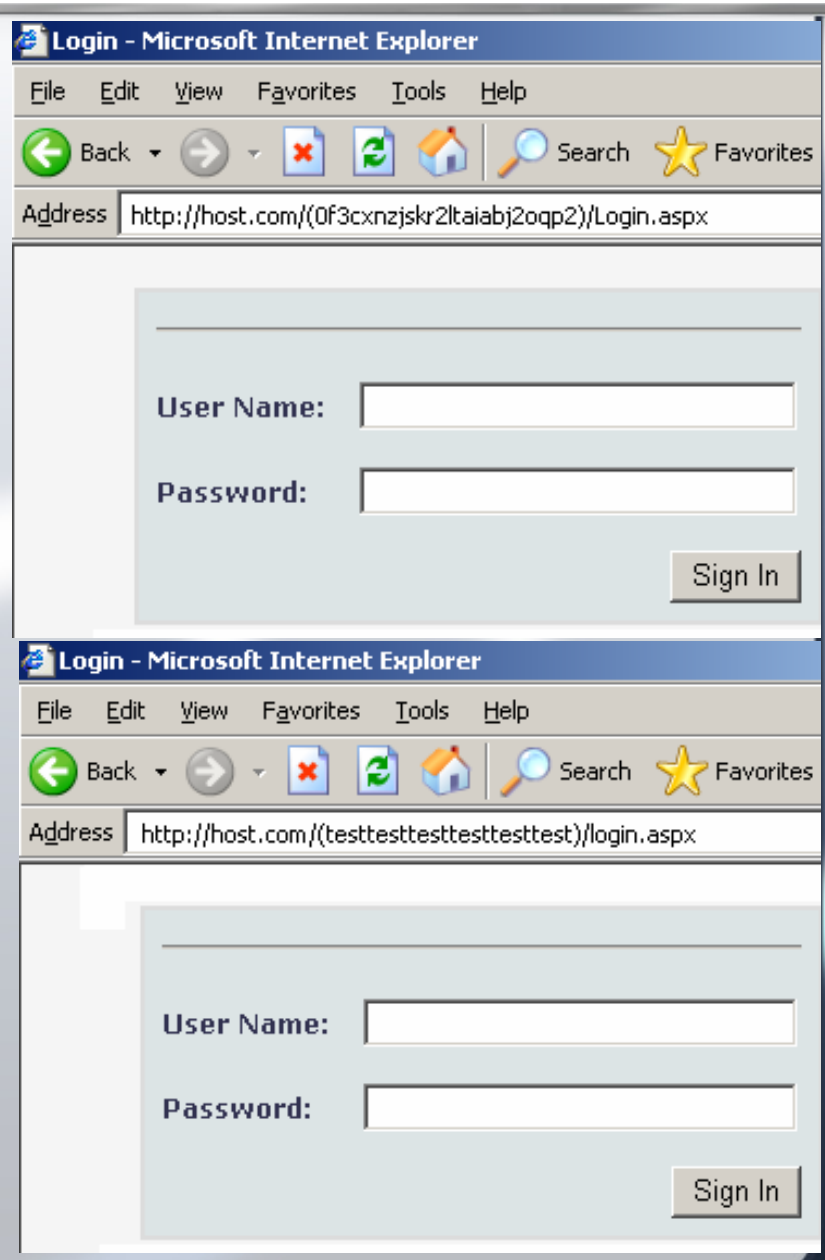
  "Please Login"

- (testtesttesttesttesttesttest)

  "Please Login"

  No difference between the two.

- Phishing Attacks.
  - Spam login pages with hard-coded session ID values.
  - Monitor the pages until a user logs in and the session is validated.
  - Account compromised.

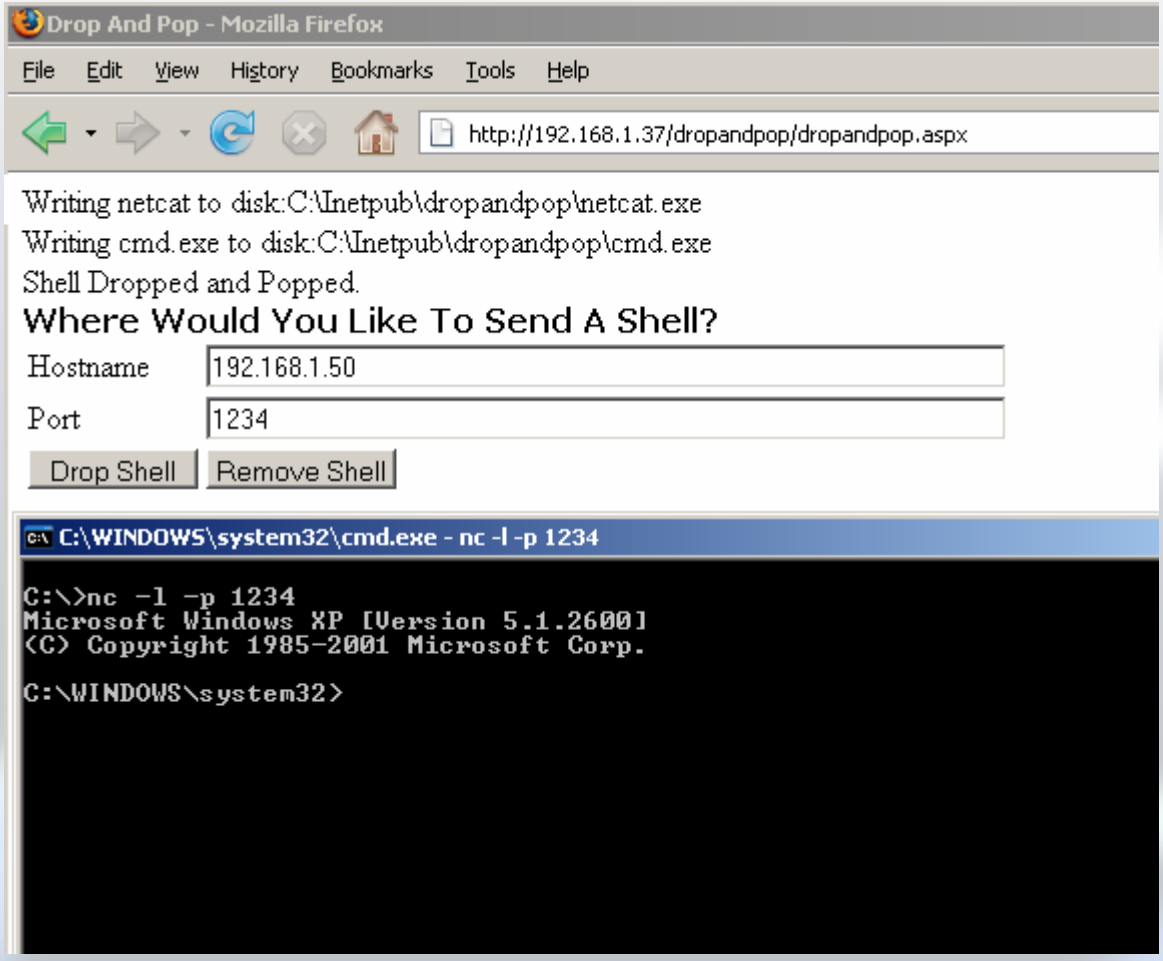


.NET Reverse Shell Dropper

- Drop And Pop
  - .NET Reserve Shell "Dropper and Popper"
  - Produces a .NET reverse shell.
  - 575kb of C#!
    - Embedded version of netcat.exe and cmd.exe.
    - Files packed with a custom version of 'Dragon Armor'.
    - Drops files to disk and spawns reverse shell.
    - .NET user prohibited access to local cmd.exe
  - Very useful in file upload attacks.
    - Graphical remote shells are nice
    - But some times I want the "Little Black Box".

http://ha.cked.net/dropandpop.zip

.NET Reverse Shell Dropper



http://ha.cked.net/dropandpop.zip

Conclusion

Conclusion

- "There are Always Vulnerabilities"
  - Parts of the .NET framework allow developers to take shortcuts
  - Not all developers are created equal.
  - Shortcuts make meet deadlines!
- Next Generation Vulnerabilities Focus on the Presentation Layer.
  - Vulnerabilities are getting simpler!
  - .NET Framework protects against easily identifiable exploits
    - XSS, input manipulation.
  - Control accessibility is not as regimented as page accessibility.
- **CLR Can Never FULLY Protect Against Human Stupidity.**
  - "Only two things are infinite, the universe and human stupidity "

Questions?

Email Me:

paul.craig <at> security-assessment.com