ACME, Inc

Web Application Audit Report

July 16, 2008

545 Boylston Street
Boston, MA 02116
Tel: (617) 247-1717
Fax: (617) 507-6488

Table of Contents

Customer Information					
Company Name:	Acme, INC				
Contact Name:	Michael Bolton		Title:	Security Manager	
Telephone:	555-867-5309		E-mail:	mjordan@acme.com	
Business Address:					
City:	Beverly Hills	State/Province:	LA	ZIP:	**90210**
URL:	http://acmeinc-not-a-real-domain.com				

Penetration Test Information					
Company Name:	Rapid7, LLC.				
Contact Name:			Title:		
Telephone:			E-mail:		
Business Address:	545 Boylston St				
City:	Boston	State/Province:	MA	ZIP:	02116
URL:	http://www.rapid7.com				

1.0 EXECUTIVE SUMMARY

On January 31, 2008, ACME contracted Rapid7, LLC to perform a web application assessment against services.ACME.com. The assessment was designed to test the security of the ACME.COM portal from an external attacker's standpoint.

Rapid7 performed the application assessment to the OWASP Top 10 methodology's standards. The assessment tests broadly for the top ten problems within web applications. Rapid7 was able to identify several key issues with the SERVICES.ACME.COM portal.

Major findings from the application assessment:

- "Internal" applications available without authorization
    - TimeReports
    - Downloads
- Verbose Error Messages
- Various forms of Information Leakage

Rapid7 began the assessment by spidering and mapping the application. This provided Rapid7 consultants with a layout of the application, and a set of functionality to test. An administrative account was used to perform this mapping.

Next, an unprivileged account was used to test the application. Rapid7 attempted to access functionality that would normally be provided to a privileged account. It was discovered that the TimeReports application on (SERVICES2) could be accessed by an unauthenticated attacker. Similarly, it was discovered that an unauthenticated attacker could access the ACME downloads. It is recommended that ACME conduct further testing (on all published applications) to ensure that applications properly enforce access control.

Additionally, Rapid7 discovered information leakage in the form of verbose ASP.NET and application login messages. It would be prudent for ACME to remove these verbose errors, perhaps replacing them with custom error messages. This procedure is widely published for most web servers, but will depend on the specific web server.

Further testing of the portal would be beneficial, mainly in the area of access control. It appears that the service2.ACME.com testing portal is particularly vulnerable to broken authorization problems.

It is recommended ACME:

- Ensure applications properly enforce access control.
- Remove verbose errors from applications and application servers.
- Ensure applications do not "leak" important information such as versions, account information, and sensitive data.

2.0 BUSINESS DESCRIPTION

Founded in 2006, ACME Inc. has production and research facilities across the globe.

3.0 REPORTING AND METHODOLOGY

This section of the report details the methodology used by Rapid7. It presents the process, timeline, and tools behind the penetration test. Additionally, this section details the report's structure and workflow.

3.1 Reporting

Structure

The report is organized into the following sections:

1.0 Executive Overview – A high-level overview of the findings from the assessment. This section gives the reader a view of the major issues discovered during the engagement.

2.0 Business Description – A description of the business being tested.

3.0 Reporting and Methodology - This section explains the methodology used during the assessment. Rapid7 aims to make its methodology both practical and auditable.

4.0 Application Scope – An overview of the application under test, accounts used, and other factors affecting the assessment.

5.0 Findings and Recommendations – An in-depth discussion of key issues found during the engagement.

6.0 OWASP Assessment – Results of the in-depth OWASP application analysis in conjunction with a list of steps performed.

Understanding the Results

Rather than report each missing patch as a vulnerability, this report describes risks and findings. A finding is a logical grouping of one or more security issue(s) having a common cause and/or a common resolution. In addition to identifying the underlying cause(s), each finding also contains hyperlinked references to resources and provides detailed remediation information.

A provided findings matrix summarizes the overall findings and can be used as a workflow plan that can be tracked within the security organization. This plan is intended to assist the remediation team in prioritizing and tracking the remediation effort. Each finding has been categorized according to its relative risk level and also contains a rating as to the amount of work and resources required in order to address the finding.

It is important to reiterate that this report represents a “snapshot” of the security posture of the environment at the point in time when these tests were conducted.

3.2 Assessment Methodology

Process

The methodology used by Rapid7 for this assessment is based on the OWASP Testing Guidelines. The OWASP standards are a comprehensive method to judge the security posture of a web application. Additionally, considerations are made for business needs and deployment configuration.

Functionally, the application assessment is divided into the following stages:

- Manual Web Application Reconnaissance
- Web Application Scan
- Verification of Scan Results
- Manual / Automated Fuzzing
- Verification of OWASP Checkpoints (see section 6.0)
- Analysis of Results
- Reporting

Tools Used

- OWASP WebScarab Application Proxy
- Burp Suite Application Proxy
- Fiddler Application Proxy
- Mozilla Firefox
  - Tamper Data
  - LiveHTTPHeaders
  - HackBar
  - FoxyProxy
  - XSSMe
  - SQLInjectMe
  - Firecookie
  - AddnEdit Cookies
  - Modify Headers
  - Poster
  - WebDeveloperToolbar
- Internet Explorer
  - IE Debug Bar
  - IE WebDev Toolbar
  - IE Watch
  - IE Tester

4.0 INTERNET FOOTPRINT

The confirmed scope for the application audit was:

- Services.ACME.com

Rapid7 was informed that this site consisted of a portal (DNN) that contained other bespoke applications. Rapid7 was given one administrative username at the beginning of the assessment, and requested another non-administrative username later in the assessment.

The provided users were:

Test1 (administrative) – later moved to a non-administrative at the request of Rapid7

Ttest20 (administrative) – requested later in the assessment

5.0 FINDINGS AND RECOMMENDATIONS

Rapid7 has identified a number of areas where security could be improved, and recommendations have been provided for consideration. This section of the report describes the details of Rapid7's observations, the impact associated with the vulnerabilities identified, and recommendations for resolving these vulnerabilities. To assist in prioritizing these findings, Rapid7 has categorized the observations with risk rankings based on the DREAD model. A scoring criteria and key for the DREAD model is provided below.

Each vulnerability or finding is assigned a composite Risk Score, calculated by adding each of the DREAD components producing a number between 5 and 50.

5.1 DREAD Scoring Criteria (Key)

Damage Criteria		Damage Description	Critical (Score: 10)	High (Score: 7)	Medium (Score : 4)	Low (Score :1)
D	Damage Potential	The level of damage and exposure that could be cased if a vulnerability were exploited	An attacker can gain full access to the system; execute commands as root/administrator	An attacker can gain non-privileged user access; leaking extremely sensitive information	Sensitive information leak; Denial of Service	Leaking trivial information
R	Reproducibility	The level of difficulty in reproducing an attack	The attack can be reproduced every time and does not require a timing window.	The attack can be reproduced most of the time.	The attack can be reproduced, but only with a timing window.	The attack is very difficult to reproduce, even with knowledge of the security hole.
E	Exploitability	The ease to which the attack could be launched	No programming skills are needed; automated exploit tools exist	A novice hacker/programmer could execute the attack in a short time.	A skilled programmer could create the attack, and a novice could repeat the steps.	The attack required a skilled person and in-depth knowledge every time to exploit.
A	Affected Users	The volume of users and assets that are affected in a successful attack scenario	All users, default configuration, key customers	Most users; common configuration	Some users; non-standard configuration	Very small percentage of users; obscure features; affects anonymous users
D	Discoverability	The level of difficulty involved in enumerating the vulnerability	Vulnerability can be found using automated scanning tools.	Published information explains the attack. The vulnerability is found in the most com-monly used feature.	The vulnerability is in a seldom-used part of the product, and few users would come across it.	The vulnerability is obscure and it is unlikely that it would be discovered.

5.2 DREAD Composite Risk Categories (Key)

RISK RATING	DREAD SCORE	RISK DESCRIPTION
CRITICAL	40-50	A critical finding or vulnerability should be considered immediately for review and resolution. Exploitation of critical vulnerabilities is relatively easy and can lead directly to an attacker gaining privileged access (root or administrator) to the system. Findings with this risk rating, if not quickly addressed, may pose risks that could negatively impact business operations or business continuity.
SEVERE	25-39	A severe finding or vulnerability should be considered for review and resolution within a short time frame. These vulnerabilities can lead to an attacker gaining non-privileged access (standard user) to a system, or the vulnerability can be leveraged to gain elevated level of access.
MODERATE	11-24	Moderate risk finding or vulnerabilities should be considered once the high critical and severe risks have been addressed. These vulnerabilities may leak sensitive data that an attacker can use to assist in the exploitation of other vulnerabilities. Moderate findings do not pose a substantial threat to business operations.

5.3 Remediation Effort Level (Key)

EFFORT RATING	EFFORT DESCRIPTION
HIGH	Significant multi-resource effort that may span over a considerable amount of time. Required a significant network architecture change or the purchase of additional security products
MODERATE	One to several days requiring moderate amounts of resources.
LOW	Less than a day requiring only a minimal amount of resources.

5.4 Findings Matrix

This table summarizes the findings documented in this report. The findings are ordered based on a weighed score of the severity of the risk and the effort of remediation.

FINDING	DREAD SCORE	REMEDIATION EFFORT
CRITICAL RISK FINDINGS (40 – 50)		
(none)		
SEVERE RISK FINDINGS (25 – 39)		
Applications Available without Authorization	35	Low
Verbose Error Messages	32	Low
Downloadable Applications Available without Authorization	31	Low
Account Information Leakage	26	Moderate
MODERATE RISK FINDINGS (11- 24)		
WSDL File Available	11	Low
LOW RISK FINDINGS (1 - 10)		
(none)		

5.5 Critical Findings and Vulnerabilities

Rating 40 -50

A critical finding or vulnerability should be considered immediately for review and resolution. Exploitation of critical vulnerabilities is relatively easy and can lead directly to an attacker gaining privileged access (root or administrator) to the system. Findings with this risk rating, if not quickly addressed, may pose risks that could negatively impact business operations or business continuity.

(none)

5.6 Severe Findings and Vulnerabilities

Rating 25-39

A severe finding or vulnerability should be considered for review and resolution within a short time frame. These vulnerabilities can lead to an attacker gaining non-privileged access (standard user) to a system, or the vulnerability can be leveraged to gain an elevated level of access.

Finding: Applications Available without Authorization

DREAD Score Summary						
Damage Potential	Reproduc-ibility	Exploit-ability	Affected Users	Discover-ability	Total	Risk Rating
7	10	7	4	7	35	**Severe**

Proof

The following site can be accessed without authentication:

- https://services2.ACME.com/timereports/timereports.aspx

This form can then be used to access reports such as:

- https://services2.ACME.com/timereports/ReportOutput/2018099122.html

This form can then be used to query sensitive information from the database.

Additionally, a user (test1) with no "internal site" authorization can access sites within the portal (both services and services2) such as:

```
/WebReports
/Agency/CommissionInfo
```

More testing is needed to confirm the extent of an unauthorized user's access.

Discussion

These reports contain sensitive information, including users, and projects that each user worked on. This could be used by an attacker to gain insight into the organization.

Recommendation

Ensure that the application is only available with authorization.

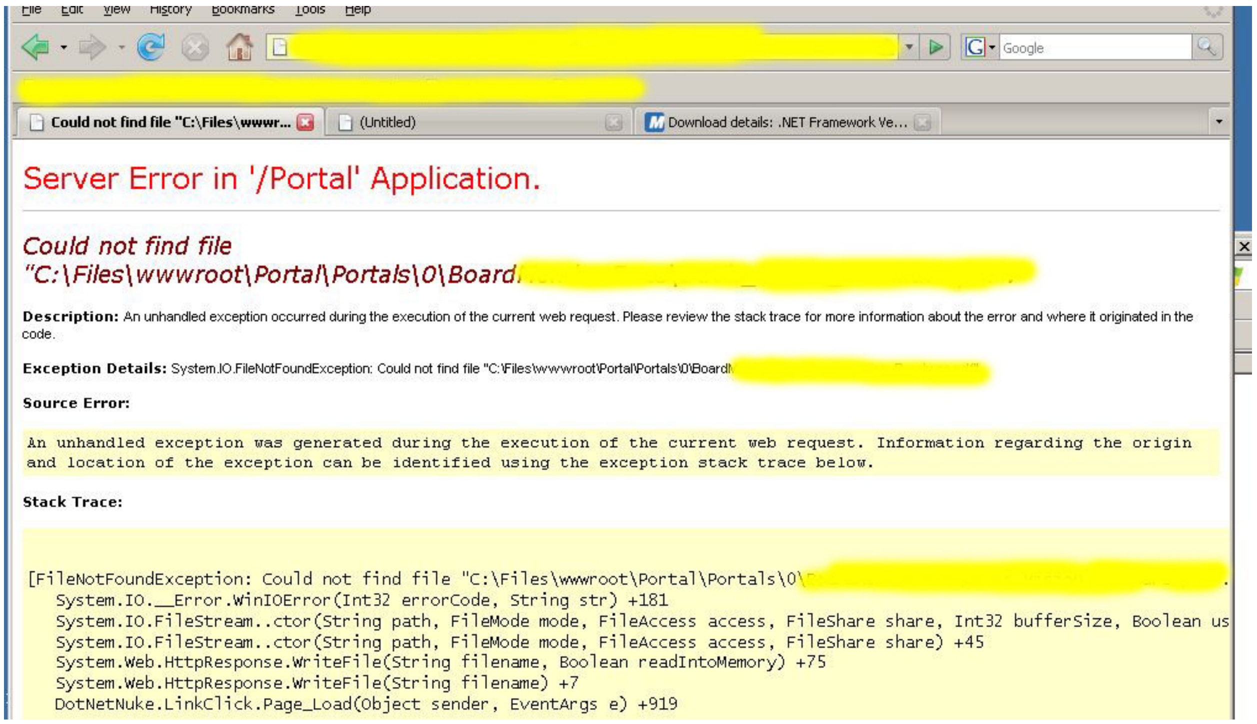
Finding: Verbose Error Messages

DREAD Score Summary						
Damage Potential	Reproduc-ibility	Exploit-ability	Affected Users	Discover-ability	Total	Risk Rating
7	10	1	7	7	32	**Severe**

Proof

Verbose error messages were found on both SERVICES.ACME.COM and SERVICES2.ACME.COM

For example:


File Edit View History Bookmarks Tools Help
[redacted]
Google
[redacted]
Could not find file "C:\Files\wwwr...
(Untitled)
Download details: .NET Framework Ve...

Server Error in '/Portal' Application.

Could not find file "C:\Files\wwwroot\Portal\Portals\0\Board[redacted]

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.IO.FileNotFoundException: Could not find file "C:\Files\wwwroot\Portal\Portals\0\Board\[redacted]

Source Error:

```
An unhandled exception was generated during the execution of the current web request. Information regarding the origin
and location of the exception can be identified using the exception stack trace below.
```

Stack Trace:

```
[FileNotFoundException: Could not find file "C:\Files\wwwroot\Portal\Portals\0\[redacted]
   System.IO.__Error.WinIOError(Int32 errorCode, String str) +181
   System.IO.FileStream..ctor(String path, FileMode mode, FileAccess access, FileShare share, Int32 bufferSize, Boolean us
   System.IO.FileStream..ctor(String path, FileMode mode, FileAccess access, FileShare share) +45
   System.Web.HttpResponse.WriteFile(String filename, Boolean readIntoMemory) +75
   System.Web.HttpResponse.WriteFile(String filename) +7
   DotNetNuke.LinkClick.Page_Load(Object sender, EventArgs e) +919
```

Here you see an example of a verbose error message giving the full path for the application, as well as configuration information.

Discussion

Motivated attackers like to see error messages as they might leak information that leads to further attacks, or may leak privacy related information. Web application error handling is rarely robust enough to survive a penetration test.

Applications should always fail safe. If an application fails to an unknown state, it is likely that an attacker may be able to exploit this indeterminate state to access unauthorized functionality, or worse create, modify or destroy data.

Recommendation

Ensure that your application has a “safe mode” which it can return to if something truly unexpected occurs. If all else fails, log the user out and close the browser window

Production code should not be capable of producing debug messages. If it does, debug mode should be triggered by editing a file or configuration option on the server. In particular, debug should not be enabled by an option in the application itself

If the framework or language has a structured exception handler (ie try {} catch {}), it should be used in preference to functional error handling

If the application uses functional error handling, its use must be comprehensive and thorough.

Detailed error messages, such as stack traces or leaking privacy related information, should never be presented to the user. Instead a generic error message should be used. This includes HTTP status response codes (ie 404 or 500 Internal Server error).

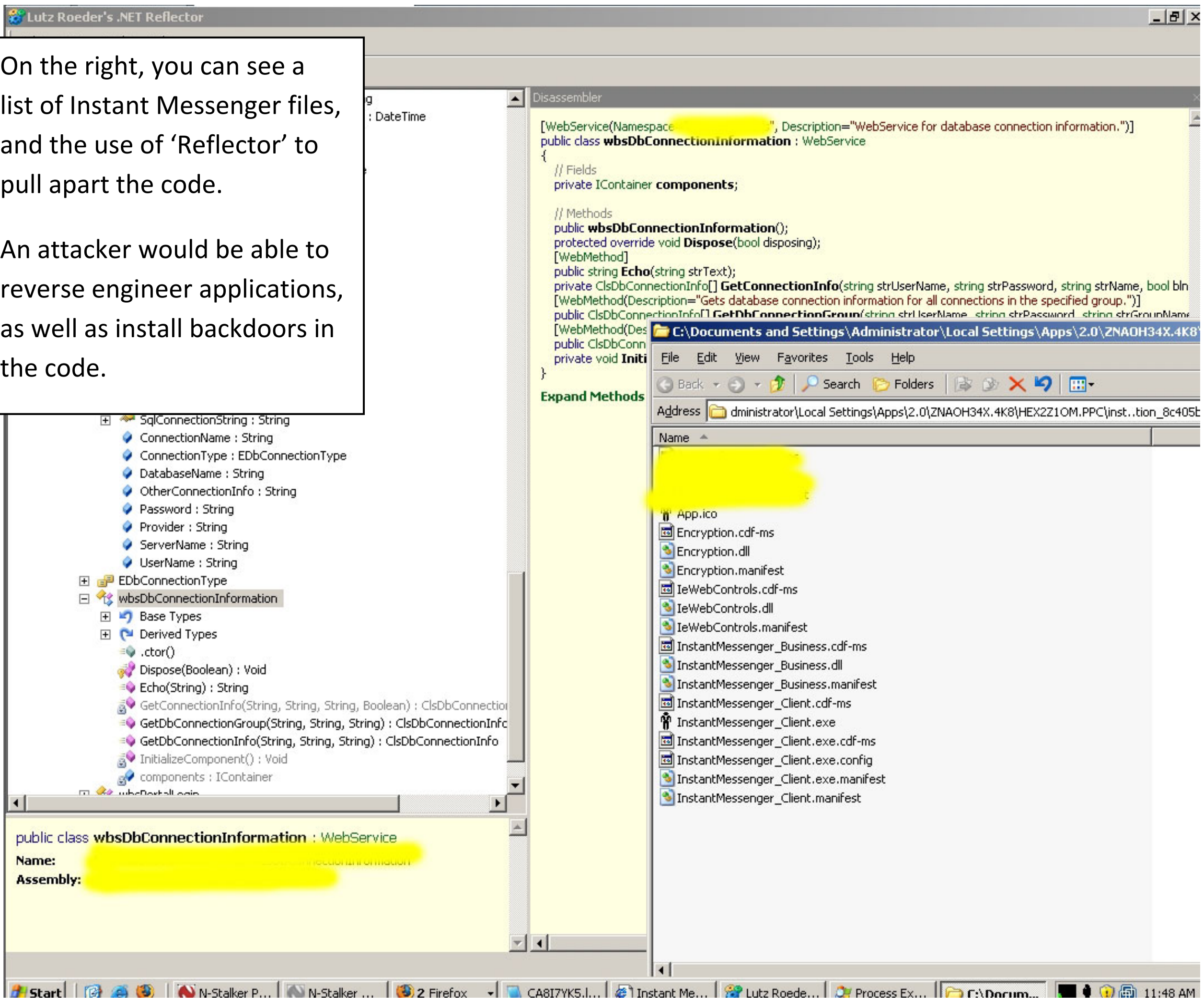
Finding: Downloadable Applications Available without Authorization

DREAD Score Summary						
Damage Potential	**Reproduc-ibility**	**Exploit-ability**	**Affected Users**	**Discover-ability**	**Total**	**Risk Rating**
4	10	7	4	7	31	**Severe**

Proof

By specifying the full path to a download, an attacker could download without authorization:

- https://services2.ACME.com/UpdaterApplication/InstantMessenger2/
- https://services2.ACME.com/Portal/admin/Portal/LinkClick.aspx?tabid=1&table=Links&field=ItemID&id=36&link=%2fdownloads%2fInstall-DST-XP.txt.exe



On the right, you can see a list of Instant Messenger files, and the use of 'Reflector' to pull apart the code.

An attacker would be able to reverse engineer applications, as well as install backdoors in the code.

```
[WebService(Namespace[redacted]", Description="WebService for database connection information.")]
public class wbsDbConnectionInformation : WebService
{
    // Fields
    private IContainer components;

    // Methods
    public wbsDbConnectionInformation();
    protected override void Dispose(bool disposing);
    [WebMethod]
    public string Echo(string strText);
    private ClsDbConnectionInfo[] GetConnectionInfo(string strUserName, string strPassword, string strName, bool bln
    [WebMethod(Description="Gets database connection information for all connections in the specified group.")]
    public ClsDbConnectionInfo[] GetDbConnectionGroup(string strUserName, string strPassword, string strGroupName
    [WebMethod(Des
    public ClsDbConn
    private void Initi
}
Expand Methods
```

```
SqlConnectionString : String
ConnectionName : String
ConnectionType : EDbConnectionType
DatabaseName : String
OtherConnectionInfo : String
Password : String
Provider : String
ServerName : String
UserName : String
EDbConnectionType
wbsDbConnectionInformation
    Base Types
    Derived Types
    .ctor()
    Dispose(Boolean) : Void
    Echo(String) : String
    GetConnectionInfo(String, String, String, Boolean) : ClsDbConnectio
    GetDbConnectionGroup(String, String, String) : ClsDbConnectionInfo
    GetDbConnectionInfo(String, String, String) : ClsDbConnectionInfo
    InitializeComponent() : Void
    components : IContainer
```

```
public class wbsDbConnectionInformation : WebService
Name: [redacted]
Assembly:
```

C:\Documents and Settings\Administrator\Local Settings\Apps\2.0\ZNAOH34X.4K8

Address: dministrator\Local Settings\Apps\2.0\ZNAOH34X.4K8\HEX2Z1OM.PPC\inst..tion_8c405b

```
Name
[redacted]
App.ico
Encryption.cdf-ms
Encryption.dll
Encryption.manifest
IeWebControls.cdf-ms
IeWebControls.dll
IeWebControls.manifest
InstantMessenger_Business.cdf-ms
InstantMessenger_Business.dll
InstantMessenger_Business.manifest
InstantMessenger_Client.cdf-ms
InstantMessenger_Client.exe
InstantMessenger_Client.exe.cdf-ms
InstantMessenger_Client.exe.config
InstantMessenger_Client.exe.manifest
InstantMessenger_Client.manifest
```

Lutz Roeder's .NET Reflector
Disassembler
File Edit View Favorites Tools Help
Back Search Folders
Start N-Stalker P... N-Stalker ... 2 Firefox CA8I7YK5.I... Instant Me... Lutz Roede... Process Ex... C:\Docum... 11:48 AM

Discussion

This is a risky configuration. Downloads can contain sensitive information too. (VPN Client, Chat Program)
Rapid7 was able to download the Chat and VPN applications and deconstruct them. Given more time, an attacker could learn much about the ACME environment.

Recommendation

- Attacker may be able to reverse engineer applications to gain credentials or other sensitive information.
- If behavior is unwanted, ensure authorization is required to download internal applications.

Finding: Account Information Leakage

DREAD Score Summary						
Damage Potential	Reproduc-ibility	Exploit-ability	Affected Users	Discover-ability	Total	Risk Rating
4	7	4	7	4	26	**Severe**

Proof

[Screenshot removed from Sample report]

Above, you see a valid account test. An attacker would now know that an account ‘test1’ exists. Below, you see an invalid account test. The username ‘richard’ does not exist.

[Screenshot removed from Sample report]

An attacker could write a simple brute force tool to enumerate accounts.

Discussion

This functionality, while useful, could allow an attacker to enumerate accounts. This would be useful in a brute force, or phishing-style attack. Best practice suggests **not** giving valid / invalid account hints.

Recommendation

Change terminology to suggest that if the account exists, an email will be sent.
Possible messages:

- "Check your email for the password reminder"
- This account will be sent a reminder. Please check your email."

5.7 Moderate Findings and Vulnerabilities

Rating 11-24

Moderate risk findings or vulnerabilities should be considered once the high critical and severe risks have been addressed. These vulnerabilities may leak sensitive data that an attacker can use to assist in the exploitation of other vulnerabilities. Moderate findings do not pose a substantial threat to business operations.

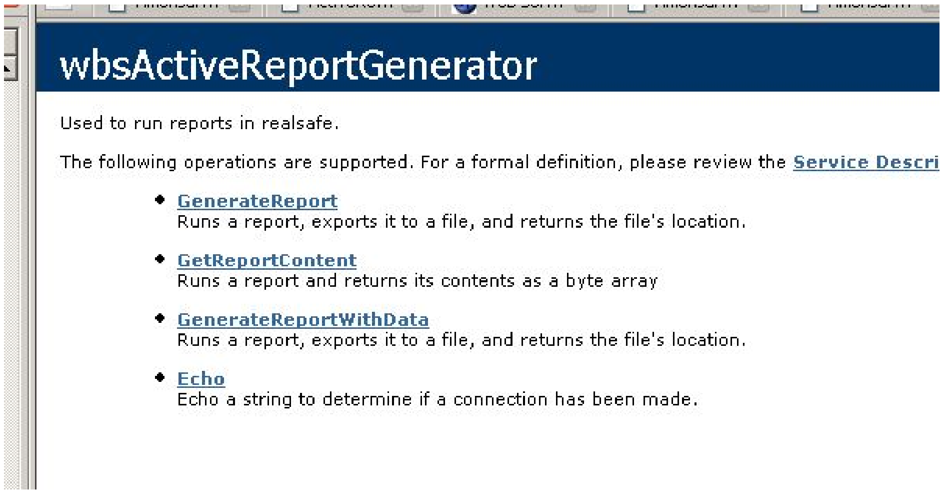
Finding: WSDL File Found

DREAD Score Summary						
Damage Potential	Reproduc-ibility	Exploit-ability	Affected Users	Discover-ability	Total	Risk Rating
4	4	1	1	1	11	**Moderate**

Proof

- https://services2.ACME.com/WebReports/WebServices/wbsActiveReportGenerator.asmx

Here you see the WSDL file as it appears in a browser.



wbsActiveReportGenerator

Used to run reports in realsafe.

The following operations are supported. For a formal definition, please review the Service Descri

- GenerateReport
  Runs a report, exports it to a file, and returns the file's location.
- GetReportContent
  Runs a report and returns its contents as a byte array
- GenerateReportWithData
  Runs a report, exports it to a file, and returns the file's location.
- Echo
  Echo a string to determine if a connection has been made.

Discussion

This finding is to make ACME aware of the published WSDL files. Other web services likely have similar files, as they are created by default when a web service is published.

Recommendation

If behavior is unwanted, remove WSDL file. See **http://whitepapers.techrepublic.com.com/abstract.aspx?docid=136930** for more information.

6.0: OWASP TOP 10 ASSESSMENT

A1 - Cross Site Scripting

Findings

No significant findings.

Steps

Theory

Look through the application for input vectors, check each potential vector for XSS.

Automated Steps

- Scan application with NeXpose
- Scan application with OWASP SQLiX
- Scan application with N-Stalker
- Spider website with webScarab
- Spider webiste with Burp Suite
- Pull down a mirror of the website using wget

Manual Steps

- Check for hidden form fields using webScarab
- Use webScarab to fuzz potential vectors
- Use TamperData to view application interaction
- Examine application inputs for potential to store XSS commands
- Examine javascript for potentially-dangerous (DOM-Based XSS) calls that:
    - Are controlled by an attacker
        - document.URL
        - document.URLEncoded
        - document.location (and many of its properties)
        - document.referrer
        - window.location
    - Write raw HTML, e.g.:
        - document.write(…)
        - document.writeln(…)
        - document.body.innerHtml=…
            - Directly modifying the DOM (including DHTML events), e.g.:
        - document.forms[0].action=… (and various other collections)
        - document.attachEvent(…)

    - document.create…(…)
    - document.execCommand(…)
    - document.body. … (accessing the DOM through the body object)
    - window.attachEvent(…)
  - Replacing the document URL, e.g.:
    - document.location=… (and assigning to location's href, host and hostname)
    - document.location.hostname=…
    - document.location.replace(…)
    - document.location.assign(…)
    - document.URL=…
    - window.navigate(…)
  - Opening/modifying a window, e.g.:
    - document.open(…)
    - window.open(…)
    - window.location.href=… (and assigning to location's href, host and hostname)
  - Directly executing script, e.g.:
    - eval(…)
    - window.execScript(…)
    - window.setInterval(…)
    - window.setTimeout(…)

References

- See: http://www.owasp.org/index.php/Top_10_2007-A1 OWASP Top 10 (XSS)
- See: http://www.owasp.org/index.php/Testing_for_Cross_site_scripting OWASP Testing Guide (XSS)
- See: http://www.webappsec.org/projects/articles/071105.shtml DOM-Based XSS (XSS of the Third Kind)

A2 - Injection Flaws

Findings

Potential here
https://services.ACME.com/Portal/admin/Portal/LinkClick.aspx?tabid=25&table=%27&field=ItemID&id=6&link=https%3a%2f%2fservices2.ACME.com%2ftimereports%2ftimereports.aspx

No other vectors found.

Steps

Theory

Analyze the application's inputs, looking for vectors for injection. Use application context to infer how input might be used (on the back end).

Automated

- Scan application with Nexpose?
- Scan application with N-Stalker
- Fuzz application using OWASP SQLiX

Manual

- Spider application using webScarab and Burp Suite
- Fuzz potential vectors using webScarab or Burp Suite
    - SQL Injection
    - LDAP
    - XPath
    - XSLT
    - HTML
    - XML
    - OS Command

References

- See: http://www.owasp.org/index.php/Top_10_2007-A2
- See: http://www.owasp.org/index.php/Testing_for_Data_Validation
- See: http://www.owasp.org/index.php/OWASP_Testing_Guide_Appendix_C:_Fuzz_Vectors#SQL_Injection

A3 - Malicious File Execution

Findings

No significant findings.

Steps

Theory

Analyze the application, looking for the use of external object references, such as URLs or file system references.

Automated

Manual

- Spider application using WebScarab or BurpSuite
    - Look for suspicious parameters (filename, include, file, language, etc)
- If you have source code, search source for 'Include' Commands

References

- See: http://www.owasp.org/index.php/Top_10_2007-A3

A4 - Insecure Direct Object Reference

Findings

No significant findings.

Steps

Theory

Examine the application for references to internal objects (database references, internal files, includes, etc)

Automated

Manual

- Most of the work will be manual, as it is context-driven.

References

- See: http://www.owasp.org/index.php/Top_10_2007-A4
- See: http://www.owasp.org/index.php/Testing_for_business_logic
- See: http://www.owasp.org/index.php/Testing_for_Directory_Traversal

A5 - Cross Site Request Forgery

Findings

No vectors found, though the application was not tested extensively for CSRF.

Encrypted viewstate is being used.

Dot Net Nuke provides solid (exterior) security around the applications.

Steps

Theory

Any application that doesn't do authorization checks for actions is likely vulnerable to CSRF. Verify that pages that perform actions for the user are protected.

Automated

- None Known

Manual

- If code is available, examine for authorization verification mechanisms
- Check to see if the application is CSRF Protections
    - Encrypted Viewstate
    - Short Timeouts
    - CAPTCHA (or something requiring human input)
    - Confirmation Page
    - using POST instead of GET
    - custom, random tokens in every URL
    - re-authentication for Sensitive URL's

References

- See: http://www.owasp.org/index.php/Top_10_2007-A5
- See: http://www.owasp.org/index.php/Testing_for_CSRF
- See: http://www.owasp.org/index.php/Cross-Site_Request_Forgery

A6 - Information Leakage and Improper Error Handling

Findings

Multiple information leaks, including account information, error information, sensitive data, and version information.

This is the major finding of the assessment! – Much potential here, just not enough time to fully exploit the system within the timeframe of the engagement.

Steps

Theory

Examine the application in question, looking for error messages, or information that shouldn't be available to an attacker. This could include comments, verbose / system error messages, or other forms of information leakage.

Automated

Manual

- Browse the application, looking for information leakage in context
- Check the application for custom errors (custom 404, custom 500)
- Check to see if exception handling is centralized.

References

- See: http://www.owasp.org/index.php/Top_10_2007-A6
- See: http://www.owasp.org/index.php/Error_Handling
- See: http://www.webappsec.org/projects/threat/classes/information_leakage.shtml
- See: http://www.owasp.org/index.php/Testing_for_Error_Code

A7 - Broken Authentication and Session Management

Findings

Dot Net Nuke provides solid security for portal.

Testing was not done on individual applications due to the limited timeframe of the engagement

Steps

Theory

Look through the application for opportunities to subvert the authentication and authorization functionality. The goal is to verify that the application properly authenticates users and properly protects identities and their associated credentials. Check for weakness in accounts, passwords, session tokens, authentication code, authorization code, etc.

Automated

- Brute force accounts (using Burp Suite?), checking for default / weak accounts

Manual

- Check for multiple authentication / authorization mechanisms.
- Analyze strength of session tokens using Burp Suite
- Check for password reset functionality. Can it be mis-used?
- Check for password remember functionality. Is it stored client-side? How?
- Check for session timeout, session invalidation (server / client).
- Check for exposed session tokens
- Check for injection on login forms (probably SQL injection)
- Check for parameter manipulation that might afford an authenticated session.

References

- See: http://www.owasp.org/index.php/Top_10_2007-A7
- See: http://www.owasp.org/index.php/Testing_for_authentication
- See: http://www.owasp.org/index.php/Testing_for_Session_Management

A8 - Insecure Cryptographic Storage

Findings

No significant findings.

Steps

Theory

The goal is to verify that the application properly encrypts sensitive information in storage.

Automated

Manual

- Check the application for
    - Not encrypting sensitive data
    - Using home grown algorithms
    - Insecure use of strong algorithms
    - Continued use of proven weak algorithms (MD5, SHA-1, RC3, RC4, etc…)
    - Hard coding keys, and storing keys in unprotected stores
- Make sure data passed in the URL is encrypted or hashed (and can't be used by an eavesdropper)

References

- See: http://www.owasp.org/index.php/Top_10_2007-A8
- See: http://www.owasp.org/index.php/Insecure_Storage
- See: http://www.owasp.org/index.php/How_to_protect_sensitive_data_in_URL%27s

A9 - Insecure Communications

Findings

No significant findings.

Steps

Theory

Check the application to ensure that encryption is used on all sensitive URLs. Also, verify that encyption strength is sufficient for the sensitivity of the data

Automated

- Run SSLDigger on website to verify supported ciphers

Manual

- Verify that all sensitive pages are encrypted with SSL.
- Verify that all calls to the server (including Flash & AJAX calls) are encrypted.

References

- See: http://www.owasp.org/index.php/Top_10_2007-A9
- See: https://www.owasp.org/index.php/Testing_for_SSL-TLS

A10 - Failure to Restrict URL Access

Findings

Major issues here. Applications "inside" the portal are not subject to the same restrictions as the portal's files.

Attackers are able to use the testing server (services2) to access data and applications.

Steps

Theory

Check application for URLS and input parameters that likely exist, based on application context.

Automated

- Run Dirbuster on application to brute force possible URLs

Manual

- Use application context to search for likely resources (/admin, etc).
- Search for:
  - "Hidden" or "special" URLs, rendered only to administrators or privileged users in the presentation layer, but accessible to all users if they know it exists, such as /admin/adduser.php or /approve_Transfer.do. This is particularly prevalent with menu code.
  - Applications often allow access to "hidden" files, such as static XML or system generated reports, trusting security through obscurity to hide them.
  - Code that enforces an access control policy but is out of date or insufficient. For example, imagine /approveTransfer.do was once available to all users, but since SOX controls were brought in, it is only supposed to be available to approvers. A fix might have been to not present it to unauthorized users, but no access control is actually enforced when requesting that page. References

- See: http://www.owasp.org/index.php/Top_10_2007-A10
- See: http://www.owasp.org/index.php/Forced_Browsing
- See: http://www.owasp.org/index.php/Guide_to_Authorization