Fortify Security Report

Feb 1, 2013

113340E

Executive Summary

Issues Overview

On February 01, 2013, a source code review was performed over the MWN code base. 173 files, 55347 lines of code were scanned and reviewed for defects that could lead to potential security vulnerabilities. A total of 500 reviewed findings were uncovered during the analysis.

Issues by Fortify Priority Order	
High	255
Low	214
Critical	30
Medium	1

Recommendations and Conclusions

The Issues Category section provides Fortify recommendations for addressing issues at a generic level. The recommendations for specific fixes can be extrapolated from those generic recommendations by the development group.

Project Summary

Code Base Summary

Code location:
Number of Files: 173
Lines of Code: 55347
Build Label: <No Build Label>

Scan Information

Scan time: 07:11
SCA Engine version: 5.10.1.0043
Machine Name: CL6G301
Username running scan: 113340E

Results Certification

Results Certification Valid

Details:

Results Signature:

SCA Analysis Results has Valid signature

Rules Signature:

There were no custom rules used in this scan

Attack Surface

Attack Surface:
$ATTACK_SURFACE$

Filter Set Summary

Current Enabled Filter Set:
Security Auditor View

Filter Set Details:

Folder Filters:
If [fortify priority order]: contains critical Then set folder to Critical
If [fortify priority order]: contains high Then set folder to High
If [fortify priority order]: contains medium Then set folder to Medium
If [fortify priority order]: contains low Then set folder to Low

Audit Guide Summary

Audit guide not enabled

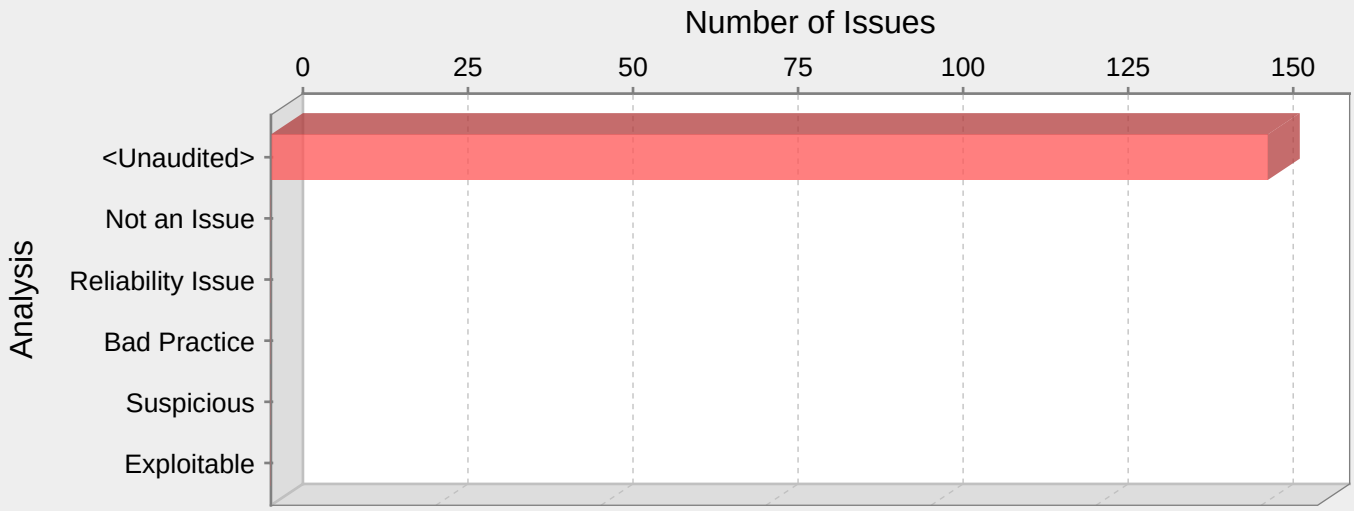
Results Outline

Overall number of results

The scan found 500 issues.

Vulnerability Examples by Category

Category: Insecure Randomness (151 Issues)



Abstract:

Standard pseudo-random number generators cannot withstand cryptographic attacks.

Explanation:

Insecure randomness errors occur when a function that can produce predictable values is used as a source of randomness in security-sensitive context.

Computers are deterministic machines, and as such are unable to produce true randomness. Pseudo-Random Number Generators (PRNGs) approximate randomness algorithmically, starting with a seed from which subsequent values are calculated.

There are two types of PRNGs: statistical and cryptographic. Statistical PRNGs provide useful statistical properties, but their output is highly predictable and forms an easy to reproduce numeric stream that is unsuitable for use in cases where security depends on generated values being unpredictable. Cryptographic PRNGs address this problem by generating output that is more difficult to predict. For a value to be cryptographically secure, it must be impossible or highly improbable for an attacker to distinguish between it and a truly random value. In general, if a PRNG algorithm is not advertised as being cryptographically secure, then it is probably a statistical PRNG and should not be used in security-sensitive contexts.

Example: The following code uses a statistical PRNG to create a URL for a receipt that remains active for some period of time after a purchase.

```
string GenerateReceiptURL(string baseUrl) {
Random Gen = new Random();
return(baseUrl + Gen.Next().toString() + ".html");
}
```

This code uses the Random.Next() function to generate "unique" identifiers for the receipt pages it generates. Because Random.Next() is a statistical PRNG, it is easy for an attacker to guess the strings it generates. Although the underlying design of the receipt system is also faulty, it would be more secure if it used a random number generator that did not produce predictable receipt identifiers, such as a cryptographic PRNG.

Recommendations:

When unpredictability is critical, as is the case with most security-sensitive uses of randomness, use a cryptographic PRNG. Regardless of the PRNG you choose, always use a value with sufficient entropy to seed the algorithm. (Values such as the current time offer only negligible entropy and should not be used.)

The .NET framework provides a cryptographic PRNG in System.Security.Cryptography.RandomNumberGenerator. As is the case with other algorithm-based classes in System.Security, RandomNumberGenerator provides an implementation-independent wrapper around a particular set of algorithms. When you request an instance of a RandomNumberGenerator object using RandomNumberGenerator.Create(), you can request a specific implementation of the algorithm. If the algorithm is available, then it is given as a RandomNumberGenerator object. If it is unavailable or if you do not specify a particular implementation, then you are given a RandomNumberGenerator implementation selected by the system.

Microsoft provides a single RandomNumberGenerator implementation with the .NET framework named RNGCryptoServiceProvider, which Microsoft describes in the following way:

"To form the seed for the random number generator, a calling application supplies bits it might have-for instance, mouse or keyboard timing input-that are then added to both the stored seed and various system data and user data such as the process ID and thread ID, the system clock, the system time, the system counter, memory status, free disk clusters, the hashed user environment block. This result is SHA-1 hashed, and the output is used to seed an RC4 stream, which is then used as the random stream and used to update the stored seed."

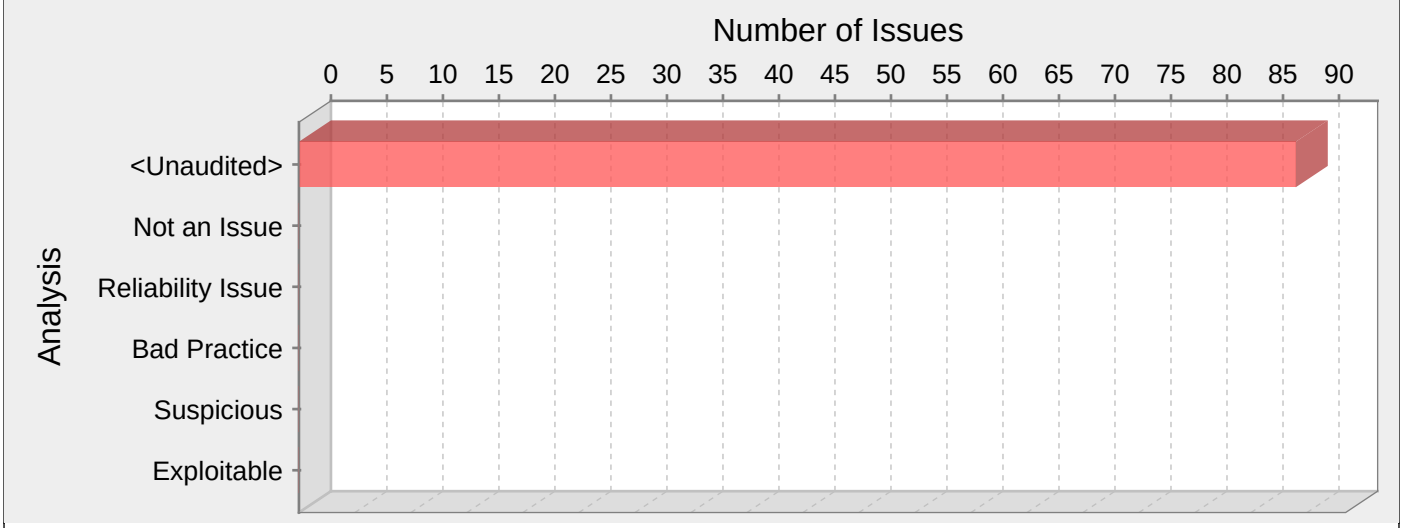
However, the specifics of the Microsoft implementation of the RNGCryptoServiceProvider algorithm are poorly documented, and it is unclear which sources of entropy the implementation uses under what circumstances and therefore what amount of true randomness exists in its output. Although there is speculation on the Web about the Microsoft implementation, there is no evidence to contradict the claim that the algorithm is cryptographically strong and can be used safely in security-sensitive contexts.

BMICalc.aspx.cs, line 295 (Insecure Randomness)

Fortify Priority:		**Folder**	High
Kingdom:	Security Features		
Abstract:	The random number generator implemented by Next() cannot withstand a cryptographic attack.		
Sink:	BMICalc.aspx.cs:295 Next()		

```
292
293             private static string RandomAdvertisement(int randomLength, bool
strongPassword)
294             {
295                 int seed = Random.Next(1, int.MaxValue);
296                 //const string allowedChars = "ABCDEFGHJKLMNOPQRSTUVWXYZ0123456789";
297                 const string allowedChars = "12345";
298                 const string specialCharacters = @"!#$%&'()*+,-./:;<=>?@[\]_";
```

Category: Unreleased Resource: Database (89 Issues)



Abstract:

The program can potentially fail to release a system resource.

Explanation:

The program can potentially fail to release a system resource.

Resource leaks have at least two common causes:

- Error conditions and other exceptional circumstances.

- Confusion over which part of the program is responsible for releasing the resource.

Most unreleased resource issues result in general software reliability problems, but if an attacker can intentionally trigger a resource leak, the attacker might be able to launch a denial of service attack by depleting the resource pool.

Example 1: The following method never closes the file handle it opens. The Finalize() method for StreamReader eventually calls Close(), but there is no guarantee as to how long it will take before the Finalize() method is invoked. In fact, there is no guarantee that Finalize() will ever be invoked. In a busy environment, this can result in the VM using up all of its available file handles.

```
private void processFile(string fName) {
StreamWriter sw = new StreamWriter(fName);
string line;
while ((line = sr.ReadLine()) != null)
processLine(line);
}
```

Example 2: Under normal conditions the following code executes a database query, processes the results returned by the database, and closes the allocated SqlConnection object. But if an exception occurs while executing the SQL or processing the results, the SqlConnection object is not closed. If this happens often enough, the database will run out of available cursors and not be able to execute any more SQL queries.

```
...
SqlConnection conn = new SqlConnection(connString);
SqlCommand cmd = new SqlCommand(queryString);
cmd.Connection = conn;
conn.Open();
SqlDataReader rdr = cmd.ExecuteReader();
HarvestResults(rdr);
conn.Connection.Close();
...
```

Recommendations:

Never rely on Finalize() to reclaim resources. In order for an object's Finalize() method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the VM is low on memory, there is no guarantee that an object's Finalize() method will be invoked in an expedient fashion, if it is ever invoked at all (the language does not guarantee that it will be). When the garbage collector finally does run, it can cause a large number of resources to be reclaimed in a short period of time, which can lead to "bursty" performance and lower overall system throughput. The effect becomes more pronounced as the load on the system increases.

Instead of explicitly closing objects that manage resources, use the C# keyword 'using', which employs the IDisposable interface to perform a cleanup. The following two blocks of code achieve the same result:

The following code uses the finally keyword:

```
StreamReader sr;
try {
sr = new StreamReader(myFileStream);
doWork(sr);
} finally {
if (sr != null) {
sr.Close();
}
}
```

The following code uses the using keyword:

```
using (StreamReader sr = new StreamReader(myFileStream)) {
doWork(sr);
}
```

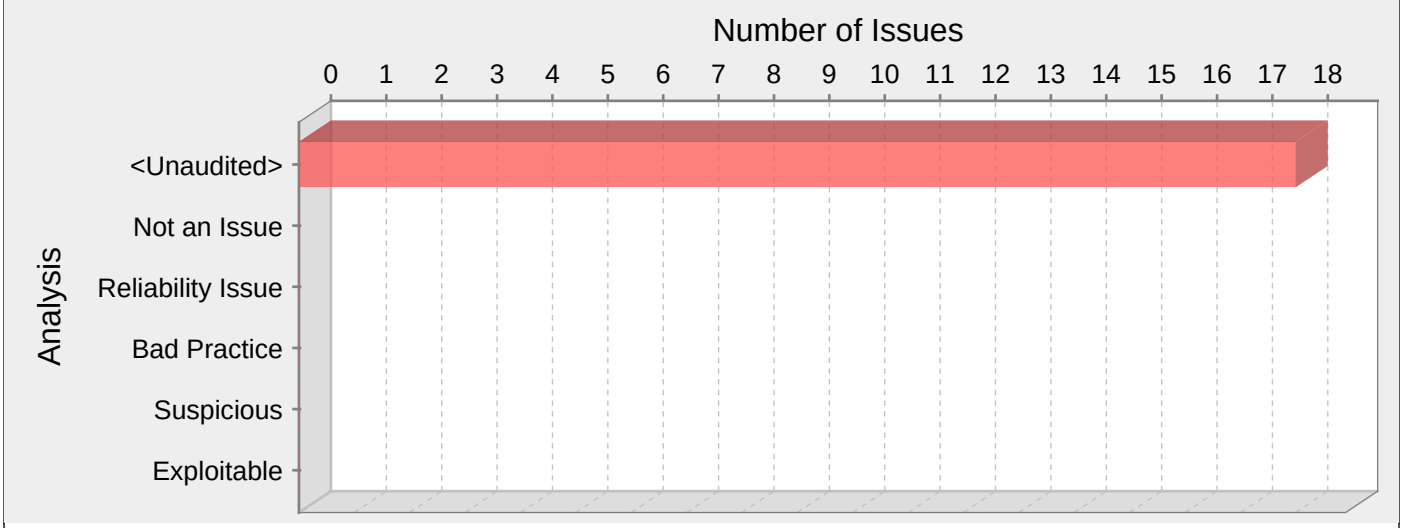
BMICalc.aspx.cs, line 29 (Unreleased Resource: Database)

Fortify Priority:		**Folder**	High
Kingdom:	Code Quality		
Abstract:	The function Page_Load() in BMICalc.aspx.cs sometimes fails to release a system resource allocated by ExecuteReader() on line 29.		

Sink: BMICalc.aspx.cs:29 reader = ExecuteReader()

```
26                          using (MySqlCommand command = new MySqlCommand(query00, conn))
27                          {
28                              command.Parameters.Add(new MySqlParameter("1",
User.Identity.Name));
29                              MySqlDataReader reader = command.ExecuteReader();
30                              while (reader.Read())
31                              {
32                                  ToShowAds = reader["AdsVisible"].ToString();
```

Category: Cross-Site Scripting: Persistent (18 Issues)



Abstract:

Sending unvalidated data to a web browser can result in the browser executing malicious code.

Explanation:

Cross-site scripting (XSS) vulnerabilities occur when:

1. Data enters a web application through an untrusted source. In the case of Persistent (also known as Stored) XSS, the untrusted source is typically a database or other back-end datastore, while in the case of Reflected XSS it is typically a web request.

2. The data is included in dynamic content that is sent to a web user without being validated for malicious code.

The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data like cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

Example 1: The following ASP.NET Web Form queries a database for an employee with a given employee ID and prints the name corresponding with the ID.

```
<script runat="server">
...
string query = "select * from emp where id=" + eid;
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
string name = dt.Rows[0]["Name"];
...
EmployeeName.Text = name;
</script>
```

Where EmployeeName is a form control defined as follows:

```
<form runat="server">
...
<asp:Label id="EmployeeName" runat="server">
...
</form>
```

Example 2: The following ASP.NET code segment is functionally equivalent to Example 1 above, but implements all of the form elements programmatically.

```
protected System.Web.UI.WebControls.Label EmployeeName;
...
string query = "select * from emp where id=" + eid;
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
```

```
sda.Fill(dt);
string name = dt.Rows[0]["Name"];
...
EmployeeName.Text = name;
```

These code examples function correctly when the values of name are well-behaved, but they do nothing to prevent exploits if they are not. This code can appear less dangerous because the value of name is read from a database, whose contents are apparently managed by the application. However, if the value of name originates from user-supplied data, then the database can be a conduit for malicious content. Without proper input validation on all data stored in the database, an attacker can execute malicious commands in the user's web browser. This type of exploit, known as Persistent (or Stored) XSS, is particularly insidious because the indirection caused by the data store makes it more difficult to identify the threat and increases the possibility that the attack will affect multiple users. XSS got its start in this form with web sites that offered a "guestbook" to visitors. Attackers would include JavaScript in their guestbook entries, and all subsequent visitors to the guestbook page would execute the malicious code.

Example 3: The following ASP.NET Web Form reads an employee ID number from an HTTP request and displays it to the user.

```
<script runat="server">
...
EmployeeID.Text = Login.Text;
...
</script>
```

Where Login and EmployeeID are form controls defined as follows:

```
<form runat="server">
<asp:TextBox runat="server" id="Login"/>
...
<asp:Label runat="server" id="EmployeeID"/>
</form>
```

Example 4: The following ASP.NET code segment shows the programmatic way to implement Example 3 above.

```
protected System.Web.UI.WebControls.TextBox Login;
protected System.Web.UI.WebControls.Label EmployeeID;
...
EmployeeID.Text = Login.Text;
```

As in Example 1 and 2, these examples operate correctly if Login contains only standard alphanumeric text. If Login has a value that includes meta-characters or source code, then the code will be executed by the web browser as it displays the HTTP response.

Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use e-mail or social engineering tricks in order to lure victims into clicking a link. When the victims click the link, they unwittingly reflect the malicious content through the vulnerable web application and back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS.

As the examples demonstrate, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response. There are three vectors by which an XSS attack can reach a victim:

- As in Examples 1 and 2, the application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Persistent XSS exploits occur when an attacker injects dangerous content into a data store that is later read and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

- As in Examples 3 and 4, data is read directly from the HTTP request and reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a user to supply dangerous content to a vulnerable web application, which is then reflected back to the user and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to victims. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces victims to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the user, the content is executed and proceeds to transfer private information, such as cookies that may include session information, from the user's machine to the attacker or perform other nefarious activities.

- A source outside the application stores dangerous data in a database or other data store, and the dangerous data is subsequently read back into the application as trusted data and included in dynamic content.

A number of modern web frameworks provide mechanisms for performing validation of user input. ASP.NET Request Validation and WCF are among them. To highlight the unvalidated sources of input, the rulepacks dynamically re-prioritize the issues reported by HP Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. In case of ASP.NET Request Validation, we also provide evidence for when validation is explicitly disabled. We refer to this feature as Context-Sensitive Ranking. To further assist the HP Fortify user with the auditing process, the Fortify Security Research Group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

Recommendations:

The solution to XSS is to ensure that validation occurs in the correct places and checks for the correct properties.

Since XSS vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application. However, because web applications often have complex and intricate code for generating dynamic content, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for XSS.

Web applications must validate their input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for XSS is generally relatively easy. Despite its value, input validation for XSS does not take the place of rigorous output validation. An application may accept input through a shared data store or other trusted source, and that data store may accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means the best way to prevent XSS vulnerabilities is to validate everything that enters the application or leaves the application destined for the user.

The most secure approach to validation for XSS is to create a whitelist of safe characters that are allowed to appear in HTTP content and accept input composed exclusively of characters in the approved set. For example, a valid username might only include alpha-numeric characters or a phone number might only include digits 0-9. However, this solution is often infeasible in web applications because many characters that have special meaning to the browser should still be considered valid input once they are encoded, such as a web design bulletin board that must accept HTML fragments from its users.

A more flexible, but less secure approach is known as blacklisting, which selectively rejects or escapes potentially dangerous characters before using the input. In order to form such a list, you first need to understand the set of characters that hold special meaning for web browsers. Although the HTML standard defines what characters have special meaning, many web browsers try to correct common mistakes in HTML and may treat other characters as special in certain contexts. The CERT(R) Coordination Center at the Software Engineering Institute at Carnegie Mellon University provides the following details about special characters in various contexts [1]:

In the content of a block-level element (in the middle of a paragraph of text):

- "<" is special because it introduces a tag.

- "&" is special because it introduces a character entity.

- ">" is special because some browsers treat it as special, on the assumption that the author of the page intended to include an opening "<", but omitted it in error.

The following principles apply to attribute values:

- In attribute values enclosed with double quotes, the double quotes are special because they mark the end of the attribute value.

- In attribute values enclosed with single quote, the single quotes are special because they mark the end of the attribute value.

- In attribute values without any quotes, white-space characters, such as space and tab, are special.

- "&" is special when used with certain attributes, because it introduces a character entity.

In URLs, for example, a search engine might provide a link within the results page that the user can click to re-run the search. This can be implemented by encoding the search query inside the URL, which introduces additional special characters:

- Space, tab, and new line are special because they mark the end of the URL.

- "&" is special because it either introduces a character entity or separates CGI parameters.

- Non-ASCII characters (that is, everything above 128 in the ISO-8859-1 encoding) are not allowed in URLs, so they are considered to be special in this context.

- The "%" symbol must be filtered from input anywhere parameters encoded with HTTP escape sequences are decoded by server -side code. For example, "%" must be filtered if input such as "%68%65%6C%6C%6F" becomes "hello" when it appears on the web page in question.

Within the body of a <SCRIPT> </SCRIPT>:

- The semicolon, parenthesis, curly braces, and new line should be filtered in situations where text could be inserted directly into a pre-existing script tag.

Server-side scripts:

- Server-side scripts that convert any exclamation characters (!) in input to double-quote characters (") on output might require additional filtering.

Other possibilities:

- If an attacker submits a request in UTF-7, the special character '<' appears as '+ADw-' and may bypass filtering. If the output is included in a page that does not explicitly specify an encoding format, then some browsers try to intelligently identify the encoding based on the content (in this case, UTF-7).

Once you identify the correct points in an application to perform validation for XSS attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. If special characters are not considered valid input to the application, then you can reject any input that contains special characters as invalid. A second option in this situation is to remove special characters with filtering. However, filtering has the side effect of changing any visual representation of the filtered content and may be unacceptable in circumstances where the integrity of the input must be preserved for display.

If input containing special characters must be accepted and displayed accurately, validation must encode any special characters to remove their significance. A complete list of ISO 8859-1 encoded values for special characters is provided as part of the official HTML specification [2].

Many application servers attempt to limit an application's exposure to cross-site scripting vulnerabilities by providing implementations for the functions responsible for setting certain specific HTTP response content that perform validation for the characters essential to a cross-site scripting attack. Do not rely on the server running your application to make it secure. When an application is developed there are no guarantees about what application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will also stay in sync.

Tips:

The HP Fortify Secure Coding Rulepacks treat the database as a source of untrusted data, which can lead to XSS vulnerabilities. If the database is a trusted resource in your environment, use custom filters to filter out dataflow issues that include the DATABASE taint flag or originate from database sources.

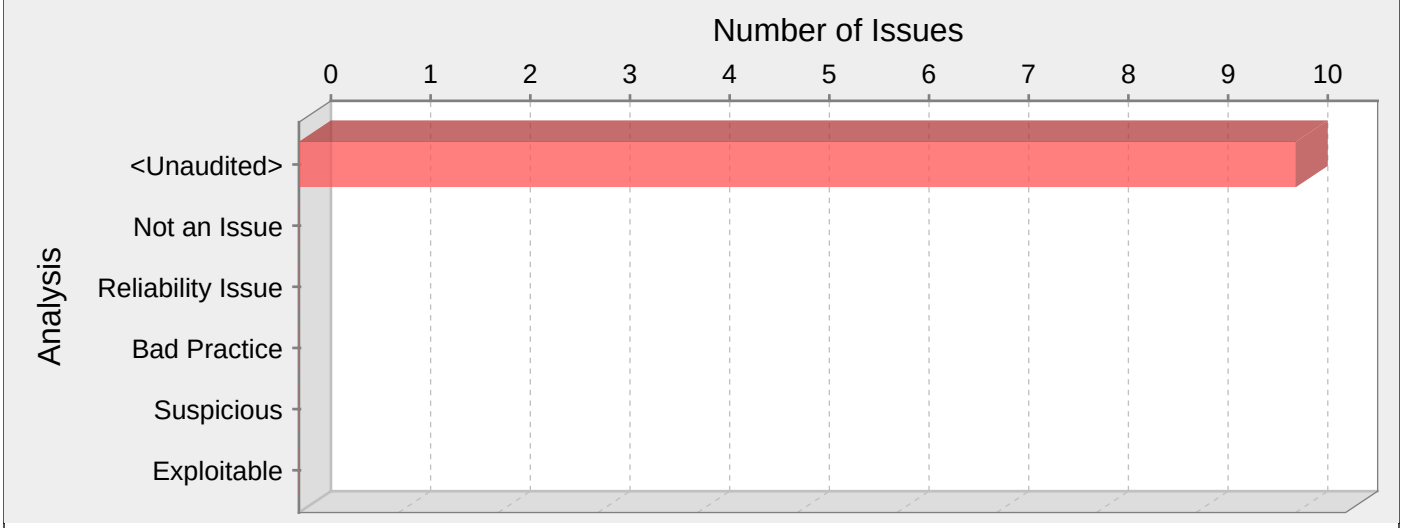
Even though URL encoding untrusted data protects against many XSS attacks, some browsers (specifically, Internet Explorer 6 and 7 and possibly others) automatically decode content at certain locations within the Document Object Model (DOM) prior to passing it to the JavaScript interpreter. To reflect this danger, the rulepacks no longer treat URL encoding routines as sufficient to protect against Cross-Site Scripting. Data values that are URL encoded and subsequently output will cause Fortify to report Cross-Site Scripting: Poor Validation vulnerabilities.

Fortify RTA adds protection against this category.

Conversation.aspx.cs, line 20 (Cross-Site Scripting: Persistent)

Fortify Priority:		**Folder**	Critical
Kingdom:	Input Validation and Representation		
Abstract:	The method Page_Load() in Conversation.aspx.cs sends unvalidated data to a web browser on line 20, which can result in the browser executing malicious code.		
Source:	Tables.cs:82 MySql.Data.MySqlClient.MySqlCommand.ExecuteReader()		
Sink:	Conversation.aspx.cs:20 System.Web.UI.WebControls.BaseDataBoundControl.set_DataSource()		

Category: Password Management: Hardcoded Password (10 Issues)



Abstract:

Hardcoded passwords can compromise system security in a way that cannot be easily remedied.

Explanation:

It is never a good idea to hardcode a password. Not only does hardcoding a password allow all of the project's developers to view the password, it also makes fixing the problem extremely difficult. Once the code is in production, the password cannot be changed without patching the software. If the account protected by the password is compromised, the owners of the system will be forced to choose between security and availability.

Example: The following code uses a hardcoded password to create a network credential:

```
...
NetworkCredential netCred =
new NetworkCredential("scott", "tiger", domain);
...
```

This code will run successfully, but anyone who has access to it will have access to the password. Once the program has shipped, there is no going back from the network credential user "scott" with a password of "tiger" unless the program is patched. A devious employee with access to this information can use it to break into the system. Even worse, if attackers have access to the executable for application, they can disassemble code, which will contain the values of the passwords used.

Recommendations:

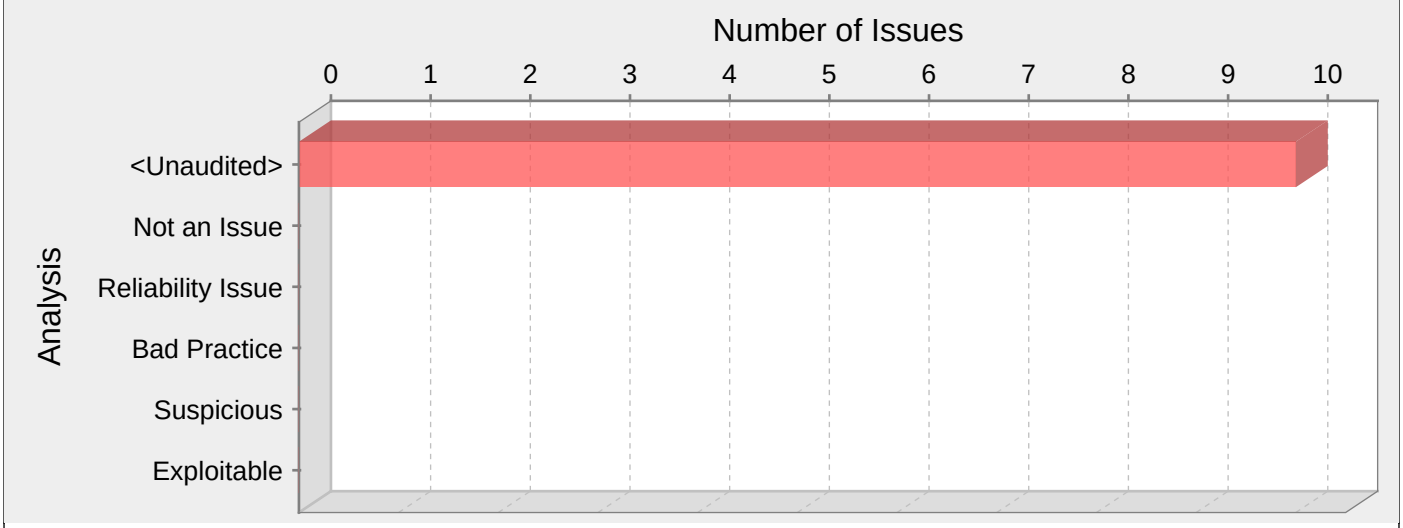
Passwords should never be hardcoded and should generally be obfuscated and managed in an external source. Storing passwords in plaintext anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the password.

Microsoft(R) provides a tool that can be used in conjunction with the Windows Data Protection application programming interface (DPAPI) to protect sensitive application entries in configuration files [1].

ChangePassword.aspx.cs, line 159 (Password Management: Hardcoded Password)

Fortify Priority:		**Folder**	Critical
Kingdom:	Security Features		
Abstract:	Hardcoded passwords can compromise system security in a way that cannot be easily remedied.		
Sink:	ChangePassword.aspx.cs:159 NetworkCredential()		

Category: Cross-Site Scripting: Reflected (10 Issues)



Abstract:

Sending unvalidated data to a web browser can result in the browser executing malicious code.

Explanation:

Cross-site scripting (XSS) vulnerabilities occur when:

1. Data enters a web application through an untrusted source. In the case of Reflected XSS, the untrusted source is typically a web request, while in the case of Persisted (also known as Stored) XSS it is typically a database or other back-end datastore.

2. The data is included in dynamic content that is sent to a web user without being validated for malicious code.

The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data like cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

Example 1: The following ASP.NET Web Form reads an employee ID number from an HTTP request and displays it to the user.

```
<script runat="server">
...
EmployeeID.Text = Login.Text;
...
</script>
```

Where Login and EmployeeID are form controls defined as follows:

```
<form runat="server">
<asp:TextBox runat="server" id="Login"/>
...
<asp:Label runat="server" id="EmployeeID"/>
</form>
```

Example 2: The following ASP.NET code segment shows the programmatic way to implement Example 1 above.

```
protected System.Web.UI.WebControls.TextBox Login;
protected System.Web.UI.WebControls.Label EmployeeID;
...
EmployeeID.Text = Login.Text;
```

The code in these examples operates correctly if Login contains only standard alphanumeric text. If Login has a value that includes meta-characters or source code, then the code will be executed by the web browser as it displays the HTTP response.

Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use e-mail or social engineering tricks in order to lure victims into clicking a link. When the victims click the link, they unwittingly reflect the malicious content through the vulnerable web application and back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS.

Example 3: The following ASP.NET Web Form queries a database for an employee with a given employee ID and prints the name corresponding with the ID.

```
<script runat="server">
...
string query = "select * from emp where id=" + eid;
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
string name = dt.Rows[0]["Name"];
...
EmployeeName.Text = name;
</script>
```

Where EmployeeName is a form control defined as follows:

```
<form runat="server">
...
<asp:Label id="EmployeeName" runat="server">
...
</form>
```

Example 4: The following ASP.NET code segment is functionally equivalent to Example 3 above, but implements all of the form elements programmatically.

```
protected System.Web.UI.WebControls.Label EmployeeName;
...
string query = "select * from emp where id=" + eid;
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
string name = dt.Rows[0]["Name"];
...
EmployeeName.Text = name;
```

As in Examples 1 and 2, these code examples function correctly when the values of name are well-behaved, but they nothing to prevent exploits if the values are not. Again, these can appear less dangerous because the value of name is read from a database, whose contents are apparently managed by the application. However, if the value of name originates from user-supplied data, then the database can be a conduit for malicious content. Without proper input validation on all data stored in the database, an attacker can execute malicious commands in the user's web browser. This type of exploit, known as Persistent (or Stored) XSS, is particularly insidious because the indirection caused by the data store makes it more difficult to identify the threat and increases the possibility that the attack will affect multiple users. XSS got its start in this form with web sites that offered a "guestbook" to visitors. Attackers would include JavaScript in their guestbook entries, and all subsequent visitors to the guestbook page would execute the malicious code.

As the examples demonstrate, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response. There are three vectors by which an XSS attack can reach a victim:

- As in Examples 1 and 2, data is read directly from the HTTP request and reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a user to supply dangerous content to a vulnerable web application, which is then reflected back to the user and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to victims. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces victims to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the user, the content is executed and proceeds to transfer private information, such as cookies that may include session information, from the user's machine to the attacker or perform other nefarious activities.

- As in Examples 3 and 4, the application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Persistent XSS exploits occur when an attacker injects dangerous content into a data store that is later read and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

- A source outside the application stores dangerous data in a database or other data store, and the dangerous data is subsequently read back into the application as trusted data and included in dynamic content.

A number of modern web frameworks provide mechanisms for performing validation of user input. ASP.NET Request Validation and WCF are among them. To highlight the unvalidated sources of input, the rulepacks dynamically re-prioritize the issues reported by HP Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. In case of ASP.NET Request Validation, we also provide evidence for when validation is explicitly disabled. We refer to this feature as Context-Sensitive Ranking. To further assist the HP Fortify user with the auditing process, the Fortify Security Research Group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

Recommendations:

The solution to XSS is to ensure that validation occurs in the correct places and checks for the correct properties.

Since XSS vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application. However, because web applications often have complex and intricate code for generating dynamic content, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for XSS.

Web applications must validate their input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for XSS is generally relatively easy. Despite its value, input validation for XSS does not take the place of rigorous output validation. An application may accept input through a shared data store or other trusted source, and that data store may accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means the best way to prevent XSS vulnerabilities is to validate everything that enters the application or leaves the application destined for the user.

The most secure approach to validation for XSS is to create a whitelist of safe characters that are allowed to appear in HTTP content and accept input composed exclusively of characters in the approved set. For example, a valid username might only include alpha-numeric characters or a phone number might only include digits 0-9. However, this solution is often infeasible in web applications because many characters that have special meaning to the browser should still be considered valid input once they are encoded, such as a web design bulletin board that must accept HTML fragments from its users.

A more flexible, but less secure approach is known as blacklisting, which selectively rejects or escapes potentially dangerous characters before using the input. In order to form such a list, you first need to understand the set of characters that hold special meaning for web browsers. Although the HTML standard defines what characters have special meaning, many web browsers try to correct common mistakes in HTML and may treat other characters as special in certain contexts. The CERT(R) Coordination Center at the Software Engineering Institute at Carnegie Mellon University provides the following details about special characters in various contexts [1]:

In the content of a block-level element (in the middle of a paragraph of text):

- "<" is special because it introduces a tag.

- "&" is special because it introduces a character entity.

- ">" is special because some browsers treat it as special, on the assumption that the author of the page intended to include an opening "<", but omitted it in error.

The following principles apply to attribute values:

- In attribute values enclosed with double quotes, the double quotes are special because they mark the end of the attribute value.

- In attribute values enclosed with single quote, the single quotes are special because they mark the end of the attribute value.

- In attribute values without any quotes, white-space characters, such as space and tab, are special.

- "&" is special when used with certain attributes, because it introduces a character entity.

In URLs, for example, a search engine might provide a link within the results page that the user can click to re-run the search. This can be implemented by encoding the search query inside the URL, which introduces additional special characters:

- Space, tab, and new line are special because they mark the end of the URL.

- "&" is special because it either introduces a character entity or separates CGI parameters.

- Non-ASCII characters (that is, everything above 128 in the ISO-8859-1 encoding) are not allowed in URLs, so they are considered to be special in this context.

- The "%" symbol must be filtered from input anywhere parameters encoded with HTTP escape sequences are decoded by server -side code. For example, "%" must be filtered if input such as "%68%65%6C%6C%6F" becomes "hello" when it appears on the web page in question.

Within the body of a <SCRIPT> </SCRIPT>:

- The semicolon, parenthesis, curly braces, and new line should be filtered in situations where text could be inserted directly into a pre-existing script tag.

Server-side scripts:

- Server-side scripts that convert any exclamation characters (!) in input to double-quote characters (") on output might require additional filtering.

Other possibilities:

- If an attacker submits a request in UTF-7, the special character '<' appears as '+ADw-' and may bypass filtering. If the output is included in a page that does not explicitly specify an encoding format, then some browsers try to intelligently identify the encoding based on the content (in this case, UTF-7).

Once you identify the correct points in an application to perform validation for XSS attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. If special characters are not considered valid input to the application, then you can reject any input that contains special characters as invalid. A second option in this situation is to remove special characters with filtering. However, filtering has the side effect of changing any visual representation of the filtered content and may be unacceptable in circumstances where the integrity of the input must be preserved for display.

If input containing special characters must be accepted and displayed accurately, validation must encode any special characters to remove their significance. A complete list of ISO 8859-1 encoded values for special characters is provided as part of the official HTML specification [2].

Many application servers attempt to limit an application's exposure to cross-site scripting vulnerabilities by providing implementations for the functions responsible for setting certain specific HTTP response content that perform validation for the characters essential to a cross-site scripting attack. Do not rely on the server running your application to make it secure. When an application is developed there are no guarantees about what application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will also stay in sync.

Tips:

The HP Fortify Secure Coding Rulepacks treat the database as a source of untrusted data, which can lead to XSS vulnerabilities. If the database is a trusted resource in your environment, use custom filters to filter out dataflow issues that include the DATABASE taint flag or originate from database sources.

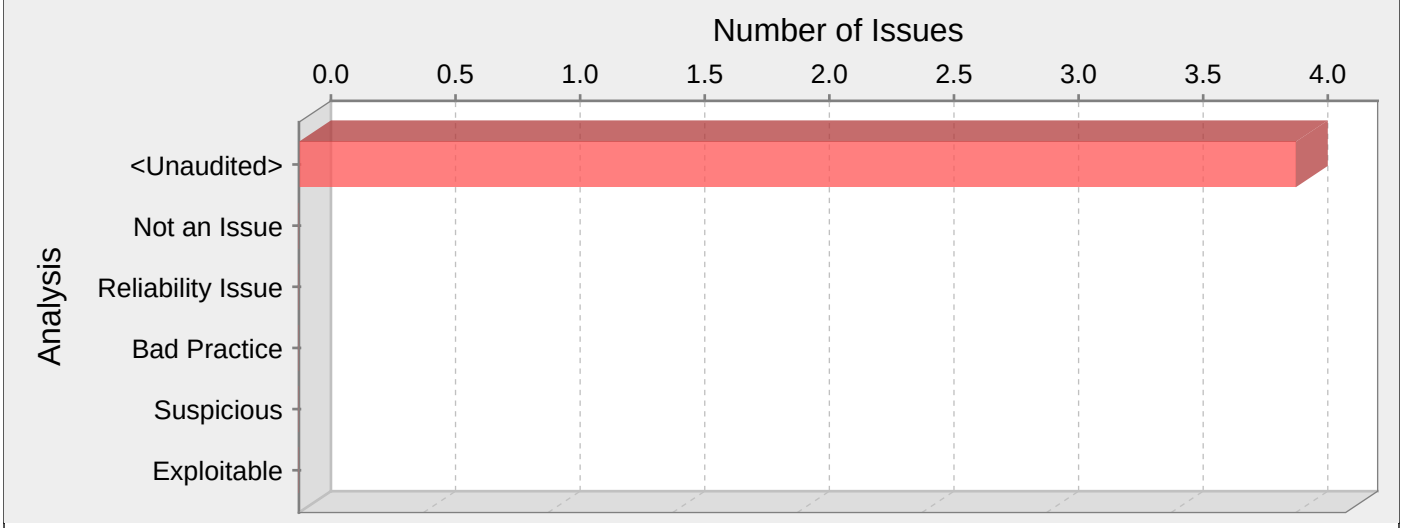
Even though URL encoding untrusted data protects against many XSS attacks, some browsers (specifically, Internet Explorer 6 and 7 and possibly others) automatically decode content at certain locations within the Document Object Model (DOM) prior to passing it to the JavaScript interpreter. To reflect this danger, the rulepacks no longer treat URL encoding routines as sufficient to protect against Cross-Site Scripting. Data values that are URL encoded and subsequently output will cause Fortify to report Cross-Site Scripting: Poor Validation vulnerabilities.

Fortify RTA adds protection against this category.

ForgotEmail.aspx.cs, line 63 (Cross-Site Scripting: Reflected)

Fortify Priority:		**Folder**	High
Kingdom:	Input Validation and Representation		
Abstract:	The method RetrieveEmail_Click() in ForgotEmail.aspx.cs sends unvalidated data to a web browser on line 63, which can result in the browser executing malicious code.		
Source:	ForgotEmail.aspx.cs:30 System.Web.UI.WebControls.TextBox.get_Text()		
Sink:	ForgotEmail.aspx.cs:63 System.Web.UI.WebControls.Label.set_Text()		

Category: Unreleased Resource: Unmanaged Object (4 Issues)



Abstract:

The program fails to dispose of a managed object that utilizes unmanaged system resources.

Explanation:

The program fails to properly dispose of a managed object that uses unmanaged system resources.

Failure to properly dispose of a managed object that uses unmanaged system resources has at least two common causes:

- Error conditions and other exceptional circumstances.

- Confusion over which part of the program is responsible for releasing the resource.

A small subset of managed .NET objects use unmanaged system resources. .NET's Garbage Collector may not free the original managed objects in a predictable way. As such, the application may run out of available memory as the Garbage Collector is unaware of the memory consumed by the unmanaged resources. Most unmanaged resource leak issues result in general software reliability problems, but if an attacker can intentionally trigger an unmanaged resource leak, the attacker might be able to launch a denial of service attack by depleting the unmanaged resource pool.

Example 1: The following method creates a managed Bitmap Object from an incoming stream incomingStream. The Bitmap is manipulated and persisted to the outgoing stream outgoingStream. The Dispose() method of incomingBitmap and outgoingBitmap is never explicitly called.

Normally, one can safely rely upon the Garbage Collector to do this at a safe time for managed objects that do not use unmanaged system resources. The Garbage Collector calls Bitmap.Dispose() when it sees fit. However, the Bitmap object utilizes scarce, unmanaged system resources. The Garbage Collector may fail to call Dispose() before the unmanaged resource pool is depleted.

```
private void processBitmap(Stream incomingStream, Stream outgoingStream, int thumbnailSize)
{
Bitmap incomingBitmap = (Bitmap)System.Drawing.Image.FromStream(incomingStream);

bool validBitmap = validateBitmap(incomingBitmap);
if (!validBitmap)
throw new ValidationException(incomingBitmap);

Bitmap outgoingBitmap = new Bitmap(incomingBitmap, new Size(thumbnailSize, thumbnailSize));
outgoingBitmap.Save(outgoingStream, ImageFormat.Bmp);
}
```

Recommendations:

Never rely on .NET's Garbage Collector to call Dispose() on objects that use unmanaged system resources. In order for an object's Dispose() method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the memory is low, there is no guarantee that an object's Dispose() method will be invoked in an expedient fashion. The garbage collector may not be aware of unmanaged system resource pools that need to be freed up due to depletion. It is possible to run out of memory within a .NET environment due to unmanaged resource depletion.

Instead of explicitly closing objects that use managed resources, use the C# keyword 'using', which employs the IDisposable interface to perform a cleanup. The following two blocks of code achieve the same result:

The following code uses the finally keyword:

```
private void processBitmap(Stream incomingStream, Stream outgoingStream, int thumbnailSize)
{
```

```
Bitmap incomingBitmap = null;
Bitmap outgoingBitmap = null;

try {

incomingBitmap = (Bitmap)System.Drawing.Image.FromStream(incomingStream);

bool validBitmap = validateBitmap(incomingBitmap);
if (!validBitmap)
throw new ValidationException(incomingBitmap);

outgoingBitmap = new Bitmap(incomingBitmap, new Size(thumbnailSize, thumbnailSize));
outgoingBitmap.Save(outgoingStream, ImageFormat.Bmp);

} finally {
if (incomingBitmap != null) {
incomingBitmap.Dispose();
}

if (outgoingBitmap != null) {
outgoingBitmap.Dispose();
}
}
}
```

The following code uses the using keyword:

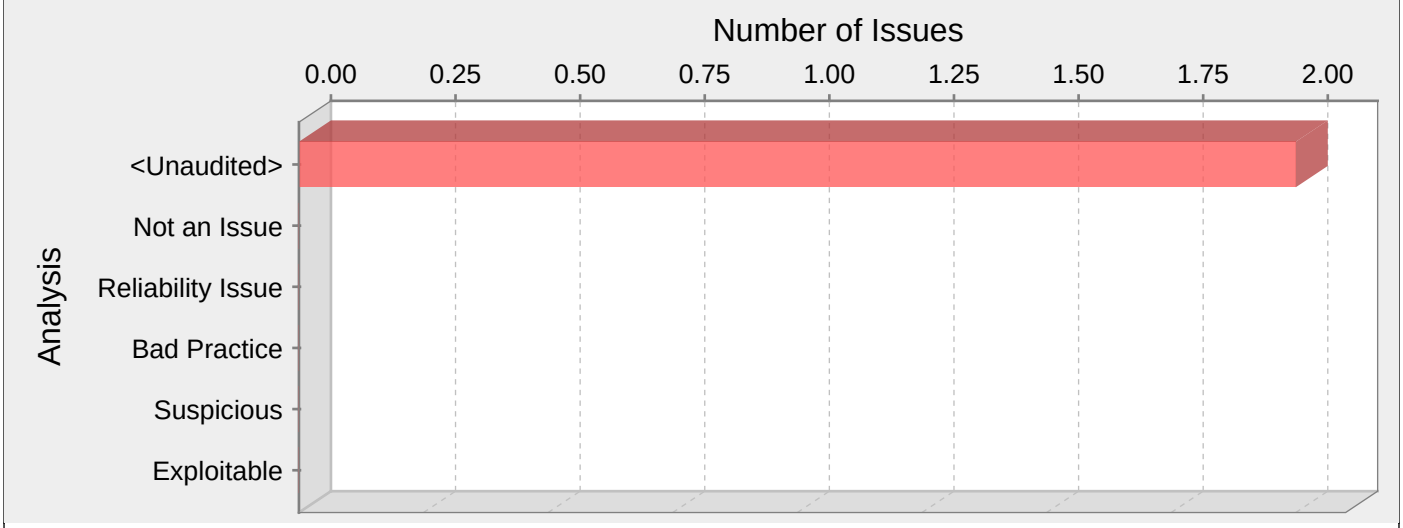
```
private void processBitmap(Stream incomingStream, Stream outgoingStream, int thumbnailSize)
{
using (Bitmap incomingBitmap = (Bitmap)System.Drawing.Image.FromStream(incomingStream))
{
bool validBitmap = validateBitmap(incomingBitmap);
if (!validBitmap)
throw new ValidationException(incomingBitmap);

using (Bitmap outgoingBitmap = new Bitmap(incomingBitmap, new Size(thumbnailSize, thumbnailSize)))
{
outgoingBitmap.Save(outgoingStream, ImageFormat.Bmp);
}
}
}
```

RandomImage.cs, line 74 (Unreleased Resource: Unmanaged Object)

Fortify Priority:		**Folder**	High
Kingdom:	Code Quality		
Abstract:	The function GenerateImage() in RandomImage.cs fails to properly dispose of unmanaged system resources allocated by HatchBrush() on line 74.		
Sink:	RandomImage.cs:74 hatchBrush = new HatchBrush(...)		

Category: Open Redirect (2 Issues)



Abstract:

Allowing unvalidated input to control the URL used in a redirect can aid phishing attacks.

Explanation:

Redirects allow web applications to direct users to different pages within the same application or to external sites. Applications utilize redirects to aid in site navigation and, in some cases, to track how users exit the site. Open redirect vulnerabilities occur when a web application redirects clients to any arbitrary URL that can be controlled by an attacker.

Attackers can utilize open redirects to trick users into visiting a URL to a trusted site and redirecting them to a malicious site. By encoding the URL, an attacker can make it more difficult for end-users to notice the malicious destination of the redirect, even when it is passed as a URL parameter to the trusted site. Open redirects are often abused as part of phishing scams to harvest sensitive end-user data.

Example 1: The following code instructs the user's browser to open a URL parsed from the dest request parameter when a user clicks the link.

```
String redirect = Request["dest"];
Response.Redirect(redirect);
```

If a victim receives an email instructing the user to follow a link to "http://trusted.example.com/ecommerce/redirect.asp?dest=www.wilyhacker.com", the user might click on the link believing they would be transferred to the trusted site. However, when the user clicks the link, the code above will redirect the browser to "http://www.wilyhacker.com".

Many users have been educated to always inspect URLs they receive in emails to make sure the link specifies a trusted site they know. However, if the attacker encoded the destination url as follows:

"http://trusted.example.com/ecommerce/redirect.asp?dest=%77%69%6C%79%68%61%63%6B%65%72%2E%63%6F%6D"

then even a savvy end-user may be fooled into following the link.

Recommendations:

Unvalidated user input should not be allowed to control the destination URL in a redirect. Instead, a level of indirection should be introduced: create a list of legitimate URLs that users are allowed to specify and only allow users to select from the list. With this approach, input provided by users is never used directly to specify a URL for redirects.

Example 2: The following code references an array populated with valid URLs. The link the user clicks passes in the array index that corresponds to the desired URL.

```
String redirect = Request["dest"];
Int32 strDest = System.Convert.ToInt32(redirect);
if((strDest >= 0) && (strDest <= strURLArray.Length -1 ))
{
strFinalURL = strURLArray[strDest];
pageContext.forward(strFinalURL);
}
```

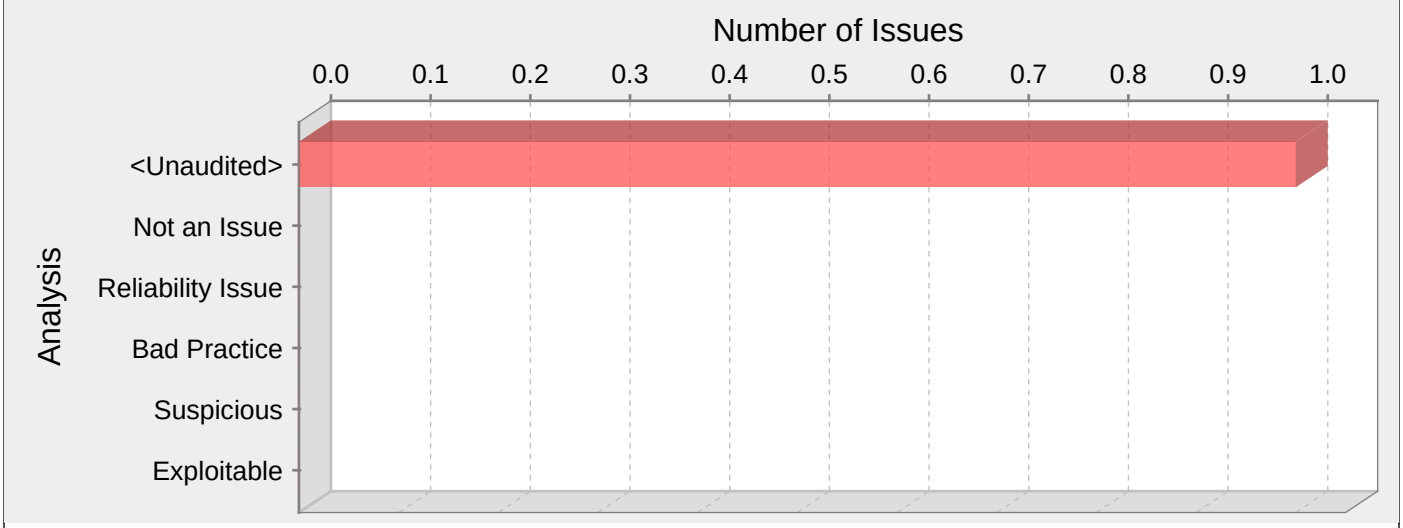
In some situations this approach is impractical because the set of legitimate URLs is too large or too hard to keep track of. In such cases, use a similar approach to restrict the domains that users can be redirected to, which can at least prevent attackers from sending users to malicious external sites.

Tips:

A number of modern web frameworks provide mechanisms for performing validation of user input. ASP.NET Request Validation and WCF are among them. To highlight the unvalidated sources of input, the HP Fortify Secure Coding Rulepacks dynamically re-prioritize the issues reported by HP Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. In case of ASP.NET Request Validation, we also provide evidence for when validation is explicitly disabled. We refer to this feature as Context-Sensitive Ranking. To further assist the HP Fortify user with the auditing process, the Fortify Security Research Group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

Login.aspx.cs, line 393 (Open Redirect)			
Fortify Priority:		**Folder**	Critical
Kingdom:	Input Validation and Representation		
Abstract:	The file Login.aspx.cs passes unvalidated data to an HTTP redirect on line 393. Allowing unvalidated input to control the URL used in a redirect can aid phishing attacks.		
Source:	Login.aspx.cs:393 System.Web.HttpRequest.get_Url()		
Sink:	Login.aspx.cs:393 System.Web.HttpResponse.Redirect()		

Category: Password Management: Password in Configuration File (1 Issues)



Abstract:

Storing a plaintext password in a configuration file could result in a system compromise.

Explanation:

Storing a plaintext password in a configuration file allows anyone who can read the file access to the password-protected resource. Developers sometimes believe that they cannot defend the application from someone who has access to the configuration, but this attitude makes an attacker's job easier. Good password management guidelines require that a password never be stored in plaintext.

Recommendations:

A password should never be stored in plaintext. Instead, the password should be entered by an administrator when the system starts. If that approach is impractical, a less secure but often adequate solution is to obfuscate the password and scatter the de-obfuscation material around the system so that an attacker has to obtain and correctly combine multiple system resources to decipher the password.

Microsoft(R) provides a tool that can be used in conjunction with the Windows Data Protection application programming interface (DPAPI) to protect sensitive application entries in configuration files [1].

Tips:

HP Fortify Static Code Analyzer searches configuration files for common names used for password properties. Audit these issues by verifying that the flagged entry is used as a password and that the password entry contains plaintext.

If the entry in the configuration file is a default password, require that it be changed in addition to requiring that it be obfuscated in the configuration file.

The following ASP .NET configuration file appears to contain an encrypted password with the hexadecimal value 0x174f7a68:

```
<configuration>
...
<appSettings>
<add key="pwd" value="0x174f7a68" />
</appSettings>
...
</configuration>
```

However, to verify that the password is actually encrypted, you must locate where the password is used in the application code and ensure that an appropriate decryption algorithm is used. If such a function is never called, then the creator of the password may have chosen a hexadecimal value to make the password hard to guess, which means the application is still vulnerable because the password is stored in plaintext.

Web.config, line 11 (Password Management: Password in Configuration File)

Fortify Priority:		**Folder**	High
Kingdom:	Environment		
Abstract:	Storing a plaintext password in a configuration file could result in a system compromise.		

Sink: Web.config:11

```
8  <configuration>
9    <connectionStrings>
```

```
10                          <add name="DBConnectionString" connectionString="server=192.168.0.1;User
                          Id=admin;database=mwn;password=123;"
11                               providerName="MySql.Data.MySqlClient" />
12                        </connectionStrings>
13
14                        <system.web>
```

Detailed Project Summary

Files Scanned

Code base location:

Files Scanned:

D:/MWN(Fortify)/MWN/Scripts/WebForms/WebParts.js 27 KB 1/31/2013 9:44:46 AM

D:/MWN(Fortify)/MWN/Account/ChangePassword.aspx 4 KB 1/31/2013 8:51:14 AM

D:/WebApplication1/WebApplication1/Login.aspx.cs 0 bytes 1/1/1970 8:00:00 AM

C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_newspage.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:11 PM

C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_receivemessage.aspx.cdcab7d2.0.cs 34 KB 2/1/2013 1:27:10 PM

C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_forgotemail.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:15 PM

D:/WebApplication1/WebApplication1/CImage.aspx.cs 0 bytes 1/1/1970 8:00:00 AM

D:/MWN(Fortify)/MWN/ChangePassword.aspx.designer.cs 5 KB 2/1/2013 1:11:50 PM

C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_resendverify.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:13 PM

D:/WebApplication1/WebApplication1/SendMessage.aspx.cs 0 bytes 1/1/1970 8:00:00 AM

D:/MWN(Fortify)/MWN/Scripts/WebForms/Menu.js 33 KB 1/31/2013 9:44:46 AM

D:/MWN(Fortify)/MWN/Scripts/elegant-press.js 111 KB 1/31/2013 9:44:48 AM

D:/WebApplication1/WebApplication1/VerifyAccount.aspx.cs 0 bytes 1/1/1970 8:00:00 AM

D:/WebApplication1/WebApplication1/Account/Login.aspx.cs 0 bytes 1/1/1970 8:00:00 AM

D:/WebApplication1/WebApplication1/Global.asax.cs 0 bytes 1/1/1970 8:00:00 AM

D:/MWN(Fortify)/MWN/RandomImage.cs 5 KB 1/31/2013 11:03:42 AM

D:/WebApplication1/WebApplication1/Tables.cs 0 bytes 1/1/1970 8:00:00 AM

D:/MWN(Fortify)/MWN/Account/Login.aspx 3 KB 1/31/2013 8:51:14 AM

D:/MWN(Fortify)/MWN/Footer.html 2 KB 2/1/2013 1:14:26 PM

D:/MWN(Fortify)/MWN/CImage.aspx.cs 2 KB 1/31/2013 11:03:40 AM

D:/MWN(Fortify)/MWN/Scripts/jquery-1.7.1.intellisense.js 150 KB 1/31/2013 9:44:44 AM

D:/WebApplication1/WebApplication1/ResendVerify.aspx.designer.cs 0 bytes 1/1/1970 8:00:00 AM

D:/MWN(Fortify)/MWN/Account/Register.aspx 6 KB 1/31/2013 8:51:14 AM

D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxNetwork.js 13 KB 1/31/2013 9:44:48 AM

D:/MWN(Fortify)/MWN/HealthTopics.aspx 10 KB 2/1/2013 1:16:28 PM

D:/MWN(Fortify)/MWN/ReceiveMessage.aspx.designer.cs 2 KB 2/1/2013 1:19:30 PM

D:/MWN(Fortify)/MWN/Web.Release.config 1 KB 1/31/2013 8:51:16 AM

D:/WebApplication1/WebApplication1/Account/Register.aspx.cs 0 bytes 1/1/1970 8:00:00 AM

D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxApplicationServices.js 9 KB 1/31/2013 9:44:46 AM

D:/MWN(Fortify)/MWN/Web.config 2 KB 1/31/2013 8:54:14 AM

D:/MWN(Fortify)/MWN/BMICalc.aspx.cs 13 KB 2/1/2013 1:11:34 PM

D:/MWN(Fortify)/MWN/Scripts/WebForms/WebForms.js 22 KB 1/31/2013 9:44:46 AM

D:/MWN(Fortify)/MWN/BMICalc.aspx 9 KB 2/1/2013 1:10:18 PM

D:/MWN(Fortify)/MWN/Login.aspx 9 KB 2/1/2013 1:26:06 PM

D:/MWN(Fortify)/MWN/Account/Login.aspx.cs 461 bytes 1/31/2013 11:03:40 AM

D:/WebApplication1/WebApplication1/ForgotEmail.aspx.cs 0 bytes 1/1/1970 8:00:00 AM

C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_register.aspx.dae9cef9.1.cs 1 KB 2/1/2013 1:27:10 PM

C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_site.master.cdcab7d2.0.cs 30 KB 2/1/2013 1:27:08 PM

D:/MWN(Fortify)/MWN/HomePage.aspx 12 KB 2/1/2013 1:16:46 PM

D:/MWN(Fortify)/MWN/Account/Register.aspx.cs 903 bytes 1/31/2013 11:03:40 AM

C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_sendmessage.aspx.cdcab7d2.0.cs 42 KB 2/1/2013 1:27:12 PM
D:/MWN(Fortify)/MWN/Scripts/jquery-1.4.1.js 165 KB 1/31/2013 9:44:48 AM
D:/WebApplication1/WebApplication1/Conversation.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_bmicalc.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:13 PM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxCore.js 30 KB 1/31/2013 9:44:46 AM
D:/MWN(Fortify)/MWN/Global.asax.cs 1 KB 1/31/2013 11:03:42 AM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxSerialization.js 6 KB 1/31/2013 9:44:48 AM
D:/MWN(Fortify)/MWN/ForgotUsername.aspx.cs 16 KB 2/1/2013 1:15:48 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_register.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:11 PM
D:/MWN(Fortify)/MWN/HealthTopics.aspx.cs 8 KB 2/1/2013 1:16:36 PM
D:/MWN(Fortify)/MWN/Account/ChangePassword.aspx.cs 341 bytes 1/31/2013 11:03:40 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_conversation.aspx.cdcab7d2.0.cs 30 KB 2/1/2013 1:27:15 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_healthtopics.aspx.cdcab7d2.0.cs 22 KB 2/1/2013 1:27:13 PM
D:/MWN(Fortify)/MWN/Connector/DBConnector.cs 525 bytes 1/31/2013 11:03:48 AM
D:/WebApplication1/WebApplication1/ForgotPassword.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
D:/WebApplication1/WebApplication1/HealthTopics.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
D:/MWN(Fortify)/MWN/Account/Web.config 330 bytes 1/31/2013 8:51:14 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_resendverify.aspx.cdcab7d2.0.cs 43 KB 2/1/2013 1:27:13 PM
D:/MWN(Fortify)/MWN/NewsPage.aspx.cs 8 KB 2/1/2013 1:19:00 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_site.master.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:08 PM
D:/MWN(Fortify)/MWN/ManageProfile.aspx.cs 14 KB 2/1/2013 1:18:50 PM
D:/MWN(Fortify)/MWN/Scripts/WebForms/WebUIValidation.js 26 KB 1/31/2013 9:44:46 AM
D:/MWN(Fortify)/MWN/Scripts/jquery-1.7.1.js 251 KB 1/31/2013 9:44:44 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_changepasswordsuccess.aspx.dae9cef9.0.cs 9 KB 2/1/2013 1:27:09 PM
D:/MWN(Fortify)/MWN/ResendVerify.aspx 8 KB 2/1/2013 1:20:02 PM
D:/MWN(Fortify)/MWN/Header.html 1 KB 1/31/2013 9:20:02 AM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxWebForms.js 39 KB 1/31/2013 9:44:48 AM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxGlobalization.js 20 KB 1/31/2013 9:44:46 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_receivemessage.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:10 PM
D:/MWN(Fortify)/MWN/Tables.cs 5 KB 1/31/2013 11:03:48 AM
D:/WebApplication1/WebApplication1/ForgotPassword.aspx.designer.cs 0 bytes 1/1/1970 8:00:00 AM
D:/WebApplication1/WebApplication1/ChangePassword.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
D:/MWN(Fortify)/MWN/Scripts/WebForms/TreeView.js 9 KB 1/31/2013 9:44:46 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_register.aspx.dae9cef9.0.cs 53 KB 2/1/2013 1:27:10 PM
D:/MWN(Fortify)/MWN/Scripts/modernizr-2.5.3.js 48 KB 1/31/2013 9:44:46 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_cimage.aspx.cdcab7d2.0.cs 10 KB 2/1/2013 1:27:16 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_login.aspx.dae9cef9.0.cs 36 KB 2/1/2013 1:27:09 PM
D:/WebApplication1/WebApplication1/Connector/DBConnector.cs 0 bytes 1/1/1970 8:00:00 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET

Files/mwn/8fe50806/141778c7/App_Web_login.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:10 PM
D:/MWN(Fortify)/MWN/Scripts/WebForms/SmartNav.js 11 KB 1/31/2013 9:44:46 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_forgotusername.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:14 PM
D:/MWN(Fortify)/MWN/NewsPage.aspx 8 KB 2/1/2013 1:18:00 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_verifyaccount.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:12 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_homepage.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:12 PM
D:/WebApplication1/WebApplication1/Comments.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MenuStandards.js 27 KB 1/31/2013 9:44:46 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_login.aspx.cdcab7d2.0.cs 53 KB 2/1/2013 1:27:10 PM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxComponentModel.js 24 KB 1/31/2013 9:44:46 AM
D:/MWN(Fortify)/MWN/VerifyAccount.aspx 7 KB 2/1/2013 1:20:36 PM
D:/WebApplication1/WebApplication1/ReceiveMessage.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_changepasswordsuccess.aspx.dae9cef9.1.cs 1 KB 2/1/2013 1:27:09 PM
D:/MWN(Fortify)/MWN/Scripts/jquery-1.7.1.min.js 92 KB 1/31/2013 9:44:46 AM
D:/WebApplication1/WebApplication1/HomePage.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_healthtopics.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:13 PM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjax.js 100 KB 1/31/2013 9:44:46 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_global.asax.1.cs 1 KB 2/1/2013 1:27:07 PM
D:/MWN(Fortify)/MWN/ResendVerify.aspx.cs 18 KB 2/1/2013 1:20:12 PM
D:/WebApplication1/WebApplication1/Account/ChangePassword.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
D:/MWN(Fortify)/MWN/ChangePassword.aspx 8 KB 2/1/2013 1:11:50 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_comments.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:15 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_comments.aspx.cdcab7d2.0.cs 24 KB 2/1/2013 1:27:15 PM
D:/MWN(Fortify)/MWN/Web.Debug.config 1 KB 1/31/2013 8:51:16 AM
D:/MWN(Fortify)/MWN/Scripts/jquery-ui-1.8.20.min.js 198 KB 1/31/2013 9:44:46 AM
D:/MWN(Fortify)/MWN/Scripts/jquery-1.4.1.min.js 70 KB 1/31/2013 9:44:42 AM
D:/MWN(Fortify)/MWN/ForgotPassword.aspx.cs 19 KB 2/1/2013 1:15:30 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_sendmessage.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:12 PM
D:/MWN(Fortify)/MWN/Scripts/jquery-ui-1.8.20.js 362 KB 1/31/2013 9:44:46 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_login.aspx.dae9cef9.1.cs 1 KB 2/1/2013 1:27:09 PM
D:/MWN(Fortify)/MWN/Register.aspx 10 KB 2/1/2013 1:26:58 PM
D:/MWN(Fortify)/MWN/Conversation.aspx.cs 9 KB 2/1/2013 1:12:56 PM
D:/MWN(Fortify)/MWN/Account/ChangePasswordSuccess.aspx 532 bytes 1/31/2013 8:51:14 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_cimage.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:16 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_changepassword.aspx.dae9cef9.0.cs 45 KB 2/1/2013 1:27:09 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_forgotpassword.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:14 PM
D:/WebApplication1/WebApplication1/Register.aspx.designer.cs 0 bytes 1/1/1970 8:00:00 AM

D:/WebApplication1/WebApplication1/Register.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_manageprofile.aspx.cdcab7d2.0.cs 48 KB 2/1/2013 1:27:14 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_bmicalc.aspx.cdcab7d2.0.cs 60 KB 2/1/2013 1:27:13 PM
D:/MWN(Fortify)/MWN/SendMessage.aspx.designer.cs 5 KB 2/1/2013 1:20:20 PM
D:/MWN(Fortify)/MWN/Account/ChangePasswordSuccess.aspx.cs 348 bytes 1/31/2013 11:03:44 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_verifyaccount.aspx.cdcab7d2.0.cs 35 KB 2/1/2013 1:27:12 PM
D:/MWN(Fortify)/MWN/js/main.js 3 KB 2/1/2013 1:23:00 PM
D:/MWN(Fortify)/MWN/Site.Master 2 KB 1/31/2013 8:51:16 AM
D:/MWN(Fortify)/MWN/Scripts/WebForms/Focus.js 3 KB 1/31/2013 9:44:46 AM
D:/MWN(Fortify)/MWN/SendMessage.aspx 7 KB 2/1/2013 1:20:20 PM
D:/MWN(Fortify)/MWN/Scripts/WebForms/GridView.js 1 KB 1/31/2013 9:44:46 AM
D:/MWN(Fortify)/MWN/CImage.aspx 434 bytes 1/31/2013 11:06:14 AM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxTimer.js 3 KB 1/31/2013 9:44:48 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_changepassword.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:13 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_forgotemail.aspx.cdcab7d2.0.cs 45 KB 2/1/2013 1:27:15 PM
D:/MWN(Fortify)/MWN/ForgotPassword.aspx 7 KB 2/1/2013 1:15:24 PM
D:/WebApplication1/WebApplication1/ForgotUsername.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_changepassword.aspx.dae9cef9.1.cs 1 KB 2/1/2013 1:27:09 PM
D:/WebApplication1/WebApplication1/NewsPage.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
D:/MWN(Fortify)/MWN/ForgotEmail.aspx 8 KB 2/1/2013 1:15:00 PM
D:/MWN(Fortify)/MWN/Site.Master.cs 335 bytes 1/31/2013 11:03:44 AM
D:/MWN(Fortify)/MWN/ChangePassword.aspx.cs 19 KB 2/1/2013 1:12:00 PM
D:/MWN(Fortify)/MWN/Scripts/_references.js 268 bytes 1/31/2013 9:44:48 AM
D:/MWN(Fortify)/MWN/js/jquery.pikachoose.full.js 36 KB 2/1/2013 1:23:00 PM
D:/MWN(Fortify)/MWN/Login.aspx.cs 25 KB 2/1/2013 1:23:50 PM
D:/WebApplication1/WebApplication1/NewsPage.aspx.designer.cs 0 bytes 1/1/1970 8:00:00 AM
D:/MWN(Fortify)/MWN/Register.aspx.cs 19 KB 2/1/2013 1:19:56 PM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxWebServices.js 5 KB 1/31/2013 9:44:48 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_forgotpassword.aspx.cdcab7d2.0.cs 45 KB 2/1/2013 1:27:14 PM
D:/WebApplication1/WebApplication1/BMICalc.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
D:/MWN(Fortify)/MWN/Scripts/WebForms/DetailsView.js 1 KB 1/31/2013 9:44:46 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_manageprofile.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:14 PM
D:/WebApplication1/WebApplication1/Site.Master.cs 0 bytes 1/1/1970 8:00:00 AM
D:/MWN(Fortify)/MWN/Conversation.aspx 6 KB 2/1/2013 1:12:50 PM
D:/MWN(Fortify)/MWN/SendMessage.aspx.cs 12 KB 2/1/2013 1:20:26 PM
D:/MWN(Fortify)/MWN/ForgotEmail.aspx.cs 14 KB 2/1/2013 1:14:48 PM
D:/MWN(Fortify)/MWN/js/jquery.pikachoose.js 30 KB 2/1/2013 1:23:00 PM
D:/WebApplication1/WebApplication1/RandomImage.cs 0 bytes 1/1/1970 8:00:00 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET
Files/mwn/8fe50806/141778c7/App_Web_newspage.aspx.cdcab7d2.0.cs 26 KB 2/1/2013 1:27:11 PM
D:/MWN(Fortify)/MWN/Scripts/jquery-1.4.1-vsdoc.js 237 KB 1/31/2013 9:44:44 AM
D:/MWN(Fortify)/MWN/ReceiveMessage.aspx 6 KB 2/1/2013 1:19:30 PM
D:/WebApplication1/WebApplication1/Account/ChangePasswordSuccess.aspx.cs 0 bytes 1/1/1970 8:00:00 AM

C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_forgotusername.aspx.cdcab7d2.0.cs 46 KB 2/1/2013 1:27:14 PM
D:/MWN(Fortify)/MWN/VerifyAccount.aspx.cs 16 KB 2/1/2013 1:20:44 PM
D:/MWN(Fortify)/MWN/ReceiveMessage.aspx.cs 10 KB 2/1/2013 1:19:40 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_register.aspx.cdcab7d2.0.cs 69 KB 2/1/2013 1:27:11 PM
D:/WebApplication1/WebApplication1/ResendVerify.aspx.cs 0 bytes 1/1/1970 8:00:00 AM
D:/MWN(Fortify)/MWN/ForgotUsername.aspx 8 KB 2/1/2013 1:26:46 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_homepage.aspx.cdcab7d2.0.cs 24 KB 2/1/2013 1:27:12 PM
D:/MWN(Fortify)/MWN/ManageProfile.aspx 8 KB 2/1/2013 1:18:40 PM
D:/MWN(Fortify)/MWN/HomePage.aspx.cs 8 KB 2/1/2013 1:16:54 PM
D:/MWN(Fortify)/MWN/Register.aspx.designer.cs 8 KB 2/1/2013 1:19:50 PM
D:/MWN(Fortify)/MWN/Comments.aspx 8 KB 2/1/2013 1:12:26 PM
D:/MWN(Fortify)/MWN/Comments.aspx.cs 8 KB 2/1/2013 1:12:34 PM
D:/MWN(Fortify)/MWN/Scripts/WebForms/MSAjax/MicrosoftAjaxHistory.js 7 KB 1/31/2013 9:44:48 AM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_global.asax.0.cs 4 KB 2/1/2013 1:27:07 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_conversation.aspx.cdcab7d2.1.cs 1 KB 2/1/2013 1:27:15 PM
C:/Windows/Microsoft.NET/Framework/v4.0.30319/Temporary ASP.NET Files/mwn/8fe50806/141778c7/App_Web_changepassword.aspx.cdcab7d2.0.cs 41 KB 2/1/2013 1:27:13 PM

Reference Elements

Classpath:

Libdirs:

c:\program files (x86)\mysql\connector net 6.5.4\assemblies\v4.0
c:\program files (x86)\reference assemblies\microsoft\framework\.netframework\v4.0

Rulepacks

Valid Rulepacks:

Name: Fortify Secure Coding Rules, Core, ABAP
Version: 2012.1.0.0010
ID: 5A663288-B1F7-416A-AEED-E5F00DC3596F
SKU: RUL13095

Name: Fortify Secure Coding Rules, Core, ActionScript 3.0
Version: 2012.1.0.0010
ID: 92127AA2-E666-4F28-B1C1-C0F6A939A089
SKU: RUL13094

Name: Fortify Secure Coding Rules, Core, Android
Version: 2012.1.0.0010
ID: FF9890E6-D119-4EE8-A591-83DCF4CA6952

SKU: RUL13093

Name: Fortify Secure Coding Rules, Core, Annotations
Version: 2012.1.0.0010
ID: 14EE50EB-FA1C-4AE8-8B59-39F952E21E3B
SKU: RUL13078

Name: Fortify Secure Coding Rules, Core, ColdFusion
Version: 2012.1.0.0010
ID: EEA7C678-058E-462A-8A59-AF925F7B7164
SKU: RUL13024

Name: Fortify Secure Coding Rules, Core, C/C++
Version: 2012.1.0.0010
ID: 711E0652-7494-42BE-94B1-DB3799418C7E
SKU: RUL13001

Name: Fortify Secure Coding Rules, Core, .NET
Version: 2012.1.0.0010
ID: D57210E5-E762-4112-97DD-019E61D32D0E
SKU: RUL13002

Name: Fortify Secure Coding Rules, Core, Java
Version: 2012.1.0.0010
ID: 06A6CC97-8C3F-4E73-9093-3E74C64A2AAF
SKU: RUL13003

Name: Fortify Secure Coding Rules, Core, JavaScript
Version: 2012.1.0.0010
ID: BD292C4E-4216-4DB8-96C7-9B607BFD9584
SKU: RUL13059

Name: Fortify Secure Coding Rules, Core, PHP
Version: 2012.1.0.0010
ID: 343CBB32-087C-4A4E-8BD8-273B5F876069
SKU: RUL13058

Name: Fortify Secure Coding Rules, Core, Python
Version: 2012.1.0.0010
ID: FD15CBE4-E059-4CBB-914E-546BDCEB422B
SKU: RUL13083

Name: Fortify Secure Coding Rules, Core, SQL
Version: 2012.1.0.0010
ID: 6494160B-E1DB-41F5-9840-2B1609EE7649
SKU: RUL13004

Name: Fortify Secure Coding Rules, Core, Classic ASP, VBScript, and VB6
Version: 2012.1.0.0010
ID: 1D426B6F-8D33-4AD6-BBCE-237ABAFAB924
SKU: RUL13060

Name: Fortify Secure Coding Rules, Extended, Configuration
Version: 2012.1.0.0010
ID: CD6959FC-0C37-45BE-9637-BAA43C3A4D56
SKU: RUL13005

Name: Fortify Secure Coding Rules, Extended, Content
Version: 2012.1.0.0010
ID: 9C48678C-09B6-474D-B86D-97EE94D38F17
SKU: RUL13067

Name: Fortify Secure Coding Rules, Extended, C/C++
Version: 2012.1.0.0010
ID: BD4641AD-A6FF-4401-A8F4-6873272F2748
SKU: RUL13006

Name: Fortify Secure Coding Rules, Extended, .NET
Version: 2012.1.0.0010
ID: 557BCC56-CD42-43A7-B4FE-CDD00D58577E
SKU: RUL13027

Name: Fortify Secure Coding Rules, Extended, Java
Version: 2012.1.0.0010
ID: AAAC0B10-79E7-4FE5-9921-F4903A79D317
SKU: RUL13007

Name: Fortify Secure Coding Rules, Extended, JSP
Version: 2012.1.0.0010
ID: 00403342-15D0-48C9-8E67-4B1CFBDEFCD2
SKU: RUL13026

Name: Fortify Secure Coding Rules, Extended, SQL
Version: 2012.1.0.0010
ID: 4BC5B2FA-C209-4DBC-9C3E-1D3EEFAF135A
SKU: RUL13025

Properties

java.vendor=Sun Microsystems Inc.
sun.java.launcher=SUN_STANDARD
sun.management.compiler=HotSpot Tiered Compilers
os.name=Windows 7
sun.boot.class.path=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\lib\resources.jar;C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\lib\rt.jar;C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\lib\sunrsasign.jar;C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\lib\jsse.jar;C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\lib\jce.jar;C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\lib\charsets.jar;C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\lib\modules\jdk.boot.jar;C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\classes
sun.desktop=windows
com.sun.management.jmxremote=true
java.vm.specification.vendor=Sun Microsystems Inc.

java.runtime.version=1.6.0_24-b07
com.fortify.sca.env.classpath=.;C:\Program Files (x86)\Java\jre7\lib\ext\QTJava.zip
user.name=113340E
dotnet.install.dir=C:\Windows\Microsoft.NET\Framework
dotnet.v30.referenceAssemblies=C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\v3.0
user.language=en
com.fortify.sca.cpfe.options=--remove_unneeded_entities --suppress_vtbl -tused
sun.boot.library.path=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\bin
com.fortify.sca.env.exesearchpath=D:\MWN(Fortify);C:\Program Files\Common Files\Microsoft Shared\Windows Live;C:\Program Files (x86)\Common Files\Microsoft Shared\Windows Live;C:\Program Files\Fortify Software\Fortify 360 v3.1.0\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Windows Live\Shared;C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\DTS\Binn\;C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Binn\VSShell\Common7\IDE\;C:\Program Files (x86)\Microsoft SQL Server\100\DTS\Binn\;C:\Program Files (x86)\Microsoft Visual Studio 9.0\Common7\IDE\PrivateAssemblies\;c:\Program Files (x86)\Microsoft ASP.NET\ASP.NET Web Pages\v1.0\;C:\Program Files\TortoiseSVN\bin;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Users\phuacs\AppData\Roaming\npm;C:\Program Files (x86)\nodejs\;C:\Program Files\Intel\DMIX;C:\Program Files\Microsoft Dynamics AX\50\BusinessConnector\Bin\;C:\Program Files (x86)\Microsoft Dynamics AX\50\Client\Bin\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Program Files (x86)\QuickTime\QTSystem\;C:\Program Files (x86)\Parasoft\Test for Visual Studio\9.1\plugins\com.parasoft.xtest.libs.vstudio\os\win32\x86;C:\Program Files (x86)\Parasoft\Test for Visual Studio\9.1\plugins\com.parasoft.xtest.runtime.vstudio.core;C:\Program Files (x86)\Parasoft\Test for Visual Studio\9.1\plugins\com.parasoft.xtest.runtime.vstudio.core;C:\Program Files (x86)\Parasoft\dotTEST\9.1\plugins\com.parasoft.xtest.libs.vstudio.dotnet\engine\bin
java.version=1.6.0_24
user.timezone=Asia/Singapore
sun.arch.data.model=32
java.endorsed.dirs=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\lib\endorsed
java.rmi.server.randomIDs=true
sun.cpu.isalist=pentium_pro+mmx pentium_pro pentium+mmx pentium i486 i386 i86
sun.jnu.encoding=Cp1252
file.encoding.pkg=sun.io
file.separator=\
java.specification.name=Java Platform API Specification
dotnet.sdk.v20.install.dir=C:\Program Files (x86)\Microsoft Visual Studio 8\SDK\v2.0
java.class.version=50.0
com.fortify.TotalPhysicalMemory=4277530624
user.country=US
java.home=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre
java.vm.info=mixed mode
stdout.isatty=false
os.version=6.1
vs.90.dotnet.install.dir=c:\Program Files (x86)\Microsoft Visual Studio 9.0\Common7\IDE
path.separator=;
java.vm.version=19.1-b02
user.variant=
java.awt.printerjob=sun.awt.windows.WPrinterJob
sun.io.unicode.encoding=UnicodeLittle
awt.toolkit=sun.awt.windows.WToolkit
com.fortify.sca.Tank=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\Core\config\tank.dat#551372#1#C:\Program Files\Fortify Software\Fortify 360 v3.1.0\Core\config\tank.a03512#4319105#1#C:\Program Files\Fortify Software\Fortify 360

v3.1.0\Core\config\tank.b03512#2204758#1#
com.fortify.sca.JVMArgs=-Dcom.sun.management.jmxremote=true -XX:SoftRefLRUPolicyMSPerMB=100 -Xss1M -Xmx600M -Xms300M -server
com.fortify.InstallRoot=C:\Program Files\Fortify Software\Fortify 360 v3.1.0
user.home=C:\Users\phuacs.SIT.000
java.specification.vendor=Sun Microsystems Inc.
java.library.path=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\bin;.;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Program Files\Common Files\Microsoft Shared\Windows Live;C:\Program Files (x86)\Common Files\Microsoft Shared\Windows Live;C:\Program Files\Fortify Software\Fortify 360 v3.1.0\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Windows Live\Shared;C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\DTS\Binn\;C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Binn\VSShell\Common7\IDE\;C:\Program Files (x86)\Microsoft SQL Server\100\DTS\Binn\;C:\Program Files (x86)\Microsoft Visual Studio 9.0\Common7\IDE\PrivateAssemblies\;c:\Program Files (x86)\Microsoft ASP.NET\ASP.NET Web Pages\v1.0\;C:\Program Files\TortoiseSVN\bin;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Users\phuacs\AppData\Roaming\npm;C:\Program Files (x86)\nodejs\;C:\Program Files\Intel\DMIX;C:\Program Files\Microsoft Dynamics AX\50\BusinessConnector\Bin\;C:\Program Files (x86)\Microsoft Dynamics AX\50\Client\Bin\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Program Files (x86)\QuickTime\QTSystem\;C:\Program Files (x86)\Parasoft\Test for Visual Studio\9.1\plugins\com.parasoft.xtest.libs.vstudio\os\win32\x86;C:\Program Files (x86)\Parasoft\Test for Visual Studio\9.1\plugins\com.parasoft.xtest.runtime.vstudio.core;C:\Program Files (x86)\Parasoft\Test for Visual Studio\9.1\plugins\com.parasoft.xtest.runtime.vstudio.core;C:\Program Files (x86)\Parasoft\dotTEST\9.1\plugins\com.parasoft.xtest.libs.vstudio.dotnet\engine\bin
java.vendor.url=http://java.sun.com/
java.vm.vendor=Sun Microsystems Inc.
java.runtime.name=Java(TM) SE Runtime Environment
java.class.path=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\Core\lib\exe\sca-exe.jar
vs.90.dotnet.clr.version=v2.0.50727
dotnet.v35.referenceAssemblies=C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\v3.5
java.vm.specification.name=Java Virtual Machine Specification
vs.100.dotnet.install.dir=C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE
java.vm.specification.version=1.0
sun.cpu.endian=little
sun.os.patch.level=Service Pack 1
com.fortify.sca.PID=3884
java.io.tmpdir=C:\Users\113340E\AppData\Local\Temp\
java.vendor.url.bug=http://java.sun.com/cgi-bin/bugreport.cgi
stderr.isatty=false
os.arch=x86
java.awt.graphicsenv=sun.awt.Win32GraphicsEnvironment
java.ext.dirs=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\jre\lib\ext;C:\Windows\Sun\Java\lib\ext
user.dir=D:\MWN(Fortify)
line.separator=

java.vm.name=Java HotSpot(TM) Server VM
vs.100.dotnet.clr.version=v4.0.30319
file.encoding=Cp1252
win32.LocalAppdata=C:\Users\113340E\AppData\Local
dotnet.sdk.v3x.install.dir=C:\Program Files\Microsoft SDKs\Windows\v6.0A
java.specification.version=1.6

com.fortify.AuthenticationKey=C:\Users\113340E\AppData\Local/Fortify/config/tools
max.file.path.length=255
com.fortify.SCAExecutablePath=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\bin\sourceanalyzer.exe
com.fortify.InstallationUserName=113340E
com.fortify.Core=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\Core
com.fortify.VS.RequireASPPrecompilation=true
com.fortify.WorkingDirectory=C:\Users\113340E\AppData\Local/Fortify
com.fortify.locale=en
com.fortify.sca.compilers.ant=com.fortify.sca.util.compilers.AntAdapter
com.fortify.sca.DefaultRulesDir=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\Core\config\rules
com.fortify.sca.compilers.ld=com.fortify.sca.util.compilers.LdCompiler
com.fortify.sca.fileextensions.ctp=PHP
com.fortify.sca.fileextensions.py=PYTHON
com.fortify.sca.LowSeverityCutoff=1.0
com.fortify.sca.fileextensions.ctl=VB6
com.fortify.sca.SuppressLowSeverity=true
com.fortify.sca.fileextensions.bas=VB6
com.fortify.sca.fileextensions.cs=CSHARP
com.fortify.sca.SolverTimeout=15
com.fortify.sca.fileextensions.asp=ASP
com.fortify.sca.fileextensions.pks=PLSQL
com.fortify.sca.fileextensions.sql=SQL
com.fortify.sca.MachineOutputMode=
com.fortify.sca.fileextensions.pkh=PLSQL
com.fortify.sca.fileextensions.cfm=CFML
com.fortify.sca.fileextensions.pkb=PLSQL
com.fortify.sca.fileextensions.dll=MSIL
com.fortify.sca.fileextensions.cfc=CFML
com.fortify.sca.DaemonCompilers=com.fortify.sca.util.compilers.GppCompiler,com.fortify.sca.util.compilers.GccCompiler,com.fortify.sca.util.compilers.AppleGppCompiler,com.fortify.sca.util.compilers.AppleGccCompiler,com.fortify.sca.util.compilers.MicrosoftCompiler,com.fortify.sca.util.compilers.MicrosoftLinker,com.fortify.sca.util.compilers.LdCompiler,com.fortify.sca.util.compilers.ArUtil,com.fortify.sca.util.compilers.SunCCompiler,com.fortify.sca.util.compilers.SunCppCompiler,com.fortify.sca.util.compilers.IntelCompiler,com.fortify.sca.util.compilers.ExternalCppAdapter
com.fortify.sca.compilers.fortify=com.fortify.sca.util.compilers.FortifyCompiler
com.fortify.sca.compilers.link=com.fortify.sca.util.compilers.MicrosoftLinker
com.fortify.sca.PrintPerformanceDataAfterScan=false
com.fortify.sca.compilers.armcc=com.fortify.sca.util.compilers.ArmCcCompiler
com.fortify.sca.fileextensions.mdl=MSIL
com.fortify.sca.compilers.devenv=com.fortify.sca.util.compilers.DevenvNetAdapter
com.fortify.sca.AllocationWebServiceURL=https://per-use.fortify.com/services/GasAllocationService
com.fortify.sca.DeadCodeFilter=true
com.fortify.sca.analyzer.controlflow.EnableMachineFiltering=false
com.fortify.sca.compilers.tcc=com.fortify.sca.util.compilers.ArmCcCompiler
com.fortify.sca.fileextensions.java=JAVA
com.fortify.sca.IndirectCallGraphBuilders=com.fortify.sca.analyzer.callgraph.VirtualCGBuilder,com.fortify.sca.analyzer.callgraph.J2EEIndirectCGBuilder,com.fortify.sca.analyzer.callgraph.JNICGBuilder,com.fortify.sca.analyzer.callgraph.StoredProcedureResolver,com.fortify.sca.analyzer.callgraph.JavaWSCGBuilder,com.fortify.sca.analyzer.callgraph.StrutsCGBuilder,com.fortify.sca.analyzer.callgraph.DotNetWSCGBuilder,com.fortify.sca.analyzer.callgraph.SqlServerSPResolver,com.fortify.sca.analyzer.callgraph.ASPCGBuilder,com.fortify.sca.analyzer.callgraph.ScriptedCGBuilder,com.fortify.sca.analyzer.callgraph.NewJspCustomTagCGBuilder
com.fortify.sca.compilers.g++-*=com.fortify.sca.util.compilers.GppCompiler

com.fortify.sca.DisplayProgress=true
com.fortify.sca.jsp.UseNativeParser=true
com.fortify.sca.fileextensions.vbscript=VBSCRIPT
com.fortify.sca.fileextensions.cpx=XML
com.fortify.sca.compilers.gcc4*=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.fileextensions.faces=JSPX
com.fortify.sca.fileextensions.properties=JAVA_PROPERTIES
com.fortify.sca.Renderer=fpr
com.fortify.sca.fileextensions.mod=MSIL
com.fortify.sca.JdkVersion=1.4
com.fortify.sca.fileextensions.wsdd=XML
com.fortify.sca.fileextensions.php=PHP
com.fortify.sca.fileextensions.config=XML
com.fortify.sca.fileextensions.jsff=JSPX
com.fortify.sca.DefaultFileTypes=java,jsp,jspx,tag,tagx,sql,cfm,php,ctp,pks,pkh,pkb,xml,config,properties,dll,exe,inc,asp,vbscript
,js,ini,bas,cls,vbs,frm,ctl,html,htm,xsd,wsdd,xmi,py,cfml,cfc,abap,xhtml,cpx,xcfg,jsff
com.fortify.sca.ResultsFile=C:\Users\113340E\AppData\Local\Fortify\VS2010-3.1\MWN\Scan.fpr
com.fortify.sca.compilers.gcc3*=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.analyzer.controlflow.EnableTimeOut=true
com.fortify.sca.compilers.msdev=com.fortify.sca.util.compilers.DevenvAdapter
com.fortify.sca.compilers.touchless=com.fortify.sca.util.compilers.FortifyCompiler
com.fortify.sca.fileextensions.xsd=XML
com.fortify.sca.BuildID=MWN
com.fortify.sca.FVDLStylesheet=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\Core/resources/sca/fvdl2html.xsl
com.fortify.sca.DisableGlobals=false
com.fortify.sca.fileextensions.abap=ABAP
com.fortify.sca.fileextensions.ini=JAVA_PROPERTIES
com.fortify.sca.compilers.icpc=com.fortify.sca.util.compilers.IntelCompiler
com.fortify.sca.fileextensions.htm=HTML
com.fortify.sca.compilers.jam=com.fortify.sca.util.compilers.TouchlessCompiler
com.fortify.sca.compilers.gcc2*=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.fileextensions.xhtml=JSPX
com.fortify.sca.fileextensions.ABAP=ABAP
com.fortify.sca.CollectPerformanceData=true
com.fortify.sca.FVDLDisableProgramData=false
com.fortify.sca.fileextensions.js=JAVASCRIPT
com.fortify.sca.fileextensions.jspx=JSPX
com.fortify.sca.fileextensions.jsp=JSP
com.fortify.sca.CustomRulesDir=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\Core\config\customrules
com.fortify.sca.compilers.g++=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.fileextensions.xcfg=XML
com.fortify.sca.DeadCodeIgnoreTrivialPredicates=true
com.fortify.sca.compilers.javac=com.fortify.sca.util.compilers.JavacCompiler
com.fortify.sca.fileextensions.tagx=JSP
com.fortify.sca.compilers.c++=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.UnicodeInputFile=true
com.fortify.sca.fileextensions.vb=VB
com.fortify.sca.DefaultAnalyzers=semantic:dataflow:controlflow:nullptr:configuration:content:structural:buffer
com.fortify.sca.AntCompilerClass=com.fortify.dev.ant.SCACompiler
com.fortify.sca.fileextensions.cls=VB6
com.fortify.sca.fileextensions.frm=VB6

com.fortify.sca.fileextensions.exe=MSIL
com.fortify.sca.compilers.g++4*=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.analyzer.controlflow.EnableRefRuleOptimization=false
com.fortify.sca.compilers.armcpp=com.fortify.sca.util.compilers.ArmCppCompiler
com.fortify.sca.analyzer.controlflow.EnableLivenessOptimization=false
com.fortify.sca.compilers.cl=com.fortify.sca.util.compilers.MicrosoftCompiler
com.fortify.sca.compilers.cc=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.cpfe.file.option=--gen_c_file_name
com.fortify.sca.ProjectRoot=C:\Users\113340E\AppData\Local/Fortify
com.fortify.sca.FVDLDisableSnippets=false
com.fortify.sca.compilers.make=com.fortify.sca.util.compilers.TouchlessCompiler
com.fortify.sca.compilers.g++3*=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.fileextensions.vbs=VB6
com.fortify.sca.SqlLanguage=TSQL
com.fortify.sca.DisableFunctionPointers=false
com.fortify.sca.NoNestedOutTagOutput=org.apache.taglibs.standard.tag.rt.core.RemoveTag,org.apache.taglibs.standard.tag.rt.core.SetTag
com.fortify.sca.compilers.tcpp=com.fortify.sca.util.compilers.ArmCppCompiler
com.fortify.sca.DefaultJarsDirs=default_jars
com.fortify.sca.BundleControlflowIssues=true
com.fortify.sca.cpfe.options=--remove_unneeded_entities --suppress_vtbl -tused
com.fortify.sca.compilers.icc=com.fortify.sca.util.compilers.IntelCompiler
com.fortify.sca.fileextensions.html=HTML
com.fortify.sca.compilers.g++2*=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.compilers.ar=com.fortify.sca.util.compilers.ArUtil
com.fortify.sca.VsVersion=10.0
com.fortify.sca.cpfe.command=C:\Program Files\Fortify Software\Fortify 360 v3.1.0\Core/private-bin/sca/cpfe.exe
com.fortify.sca.compilers.gcc=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.compilers.gcc-*=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.fileextensions.xml=XML
com.fortify.sca.fileextensions.cfml=CFML
com.fortify.sca.compilers.gmake=com.fortify.sca.util.compilers.TouchlessCompiler
com.fortify.sca.fileextensions.xmi=XML
com.fortify.sca.DisableDeadCodeElimination=false
com.fortify.sca.fileextensions.tag=JSP
com.fortify.sca.compilers.nmake=com.fortify.sca.util.compilers.TouchlessCompiler
com.fortify.sca.FVDLDisableDescriptions=false
com.fortify.sca.FVDLAllowUnifiedVulnerability=true
com.fortify.sca.compilers.clearmake=com.fortify.sca.util.compilers.TouchlessCompiler

Commandline Arguments

-scan
-vsversion
10.0
-b
MWN
-machine-output
-f
C:\Users\113340E\AppData\Local\Fortify\VS2010-3.1\MWN\Scan.fpr
-format

fpr

Warnings

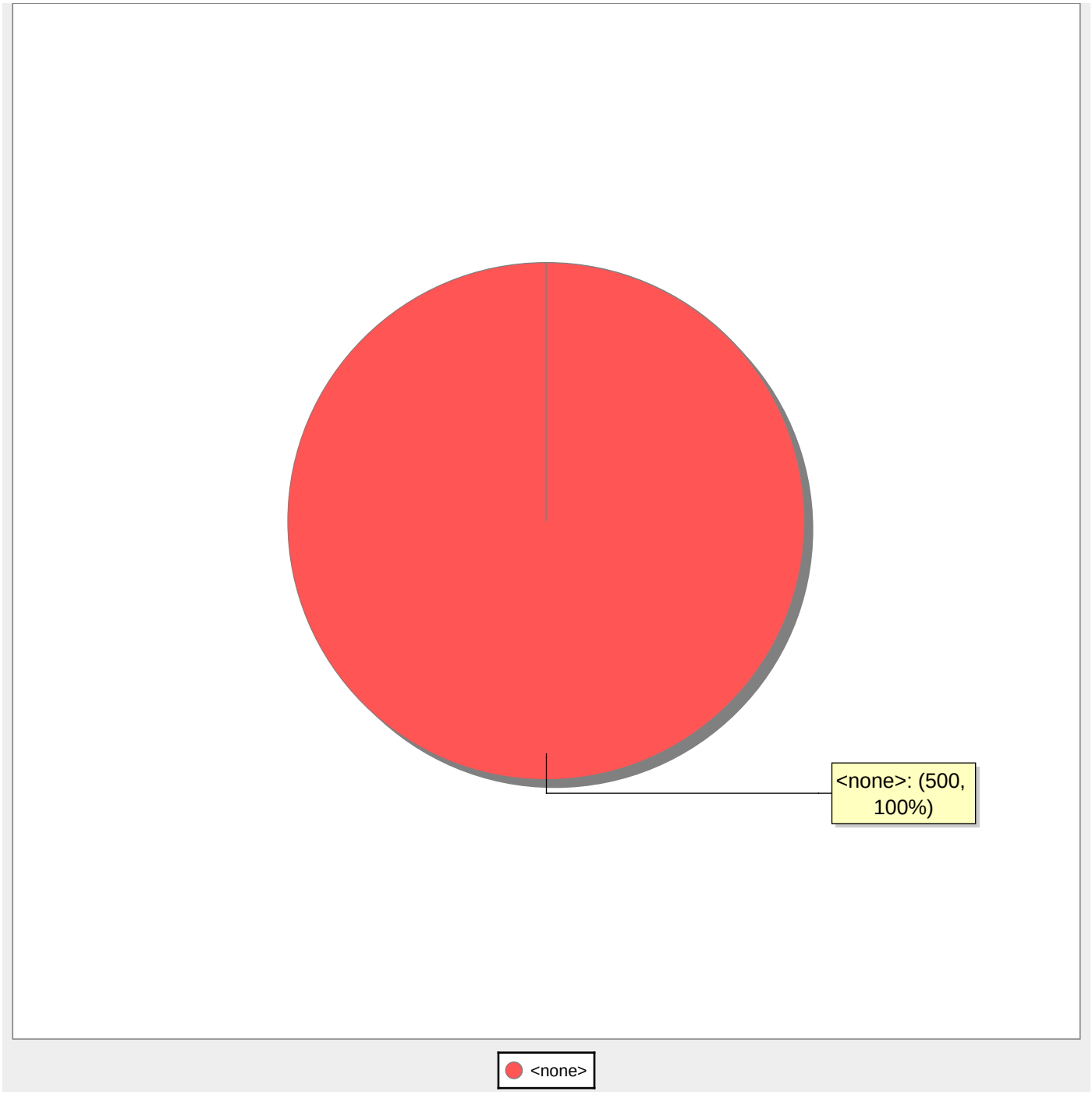
[6001] No files were excluded as the file patterns: [C:\Users\113340E\AppData\Local\Fortify\VS2010-3.1\output\MWN\wuisucdv.ogy***.vshost.exe, C:\Users\113340E\AppData\Local\Fortify\VS2010-3.1\output\MWN\wuisucdv.ogy***XmlSerializers.dll, C:\Users\113340E\AppData\Local\Fortify\VS2010-3.1\output\MWN\wuisucdv.ogy**\Properties.Resources.Designer*dll] specified for -exclude option did not match any files.

Issue Count by Category

Issues by Category	
Insecure Randomness	151
Command Injection	107
Unreleased Resource: Database	89
Dead Code: Unused Method	34
Password Management: Password in Comment	30
JavaScript Hijacking: Vulnerable Framework	28
Cross-Site Scripting: Persistent	18
Password Management: Hardcoded Password	10
Cross-Site Scripting: Reflected	10
Cross-Site Request Forgery	9
Missing Check against Null	4
Unreleased Resource: Unmanaged Object	4
Trust Boundary Violation	2
Open Redirect	2
ASP.NET Misconfiguration: Debug Information	1
Password Management: Password in Configuration File	1

Issue Breakdown by Analysis

Issues by Analysis



New Issues

Issues by New Issue

The following issues have been discovered since the last scan.