ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	5
Medium	2
Low	4
Informational	0

Alert Detail

High (Medium)	Path Traversal
Description	The Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory. An attacker may manipulate a URL in such a way that the web site will execute or reveal the contents of arbitrary files anywhere on the web server. Any device that exposes an HTTP-based interface is potentially vulnerable to Path Traversal. Most web sites restrict user access to a specific portion of the file-system, typically called the "web document root" or "CGI root" directory. These directories contain the files intended for user access and the executable necessary to drive web application functionality. To access files or execute commands anywhere on the file-system, Path Traversal attacks will utilize the ability of special-characters sequences. The most basic Path Traversal attack uses the "../" special-character sequence to alter the resource location requested in the URL. Although most popular web servers will prevent this technique from escaping the web document root, alternate encodings of the "../" sequence may help bypass the security filters. These method variations include valid and invalid Unicode-encoding ("..%u2216" or "..%c0%af") of the forward slash character, backslash characters ("..\") on Windows-based servers, URL encoded characters "%2e%2e%2f"), and double URL encoding ("..%255c") of the backslash character. Even if the web server properly restricts Path Traversal attempts in the URL path, a web application itself may still be vulnerable due to improper handling of user-supplied input. This is a common problem of web applications that use template mechanisms or load static text from files. In variations of the attack, the original URL parameter value is substituted with the file name of one of the web application's dynamic scripts. Consequently, the results can reveal source code because the file is interpreted as text instead of an executable script. These techniques often employ additional special characters such as the dot (".") to reveal the listing of the current working directory, or "%00" NULL characters in order to bypass rudimentary file extension checks.
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?AgreementID=%5CArticleCitator&ProvCode=Generally&aaprovid=76304&cat=arbitrationrules&docid=449&id=201&sort=&subcat=&toc=content
Parameter	AgreementID
Attack	\ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?AgreementID=907&ProvCode=ArticleCitator&aaprovid=76304&cat=arbitrationrules&docid=449&id=201&sort=&subcat=&toc=content
Parameter	ProvCode
Attack	ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?AgreementID=907&ProvCode=Generally&aaprovid=%2FArticleCitator&cat=arbitrationrules&docid=449&id=201&sort=&subcat=&toc=content
Parameter	aaprovid
Attack	/ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?AgreementID=907&ProvCode=Generally&aaprovid=76304&cat=arbitrationrules&docid=449&id=%2FArticleCitator&sort=&subcat=&toc=content
Parameter	id
Attack	/ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?AgreementID=907&ProvCode=Generally&aaprovid=76304&cat=arbitrationrules&docid=449&id=201&sort=&subcat=%2FArticleCitator&toc=content
Parameter	subcat
Attack	/ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?AgreementID=907&ProvCode=Generally&aaprovid=76304&cat=arbitrationrules&docid=449&id=201&sort=&subcat=&toc=%2FArticleCitator
Parameter	toc
Attack	/ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?cat=arbitrationrules&docidclose=449&id=201&subcat=ArticleCitator&toc=content
Parameter	subcat
Attack	ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=671452&disputeID=%5CJurisprudenceCitators&id=11&sort=&type=
Parameter	disputeID
Attack	\JurisprudenceCitators
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=671452&disputeID=671452&id=11&sort=JurisprudenceCitators&type=
Parameter	sort
	JurisprudenceCitators

Attack	JurisprudenceCitators
URL	http://dev.investorstatelawguide.com/ResearchTools/FullTextSearch?id=%5CFullTextSearch&tab=r&toc=content
Parameter	id
Attack	\FullTextSearch
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?cat=arbitrationrules&docid=449&id=201&subcat=&toc=%2FArticleCitator
Parameter	toc
Attack	/ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?toc=content&id=%2FArticleCitator&cat=arbitrationrules&subcat=
Parameter	id
Attack	/ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/TermsPhrases?letter=%5CTermsPhrases&searchType=T&toc=termsPhrases
Parameter	letter
Attack	\TermsPhrases
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?AgreementID=907&ProvCode=Generally&aaprovid=%5CArticleCitator&cat=arbitrationrules&docid=449&id=201&sort=&subcat=&toc=content
Parameter	aaprovid
Attack	\ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?cat=arbitrationrules&docidclose=449&id=%5CArticleCitator&subcat=&toc=content
Parameter	id
Attack	\ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?cat=arbitrationrules&docidclose=449&id=201&subcat=%2FArticleCitator&toc=content
Parameter	subcat
Attack	/ArticleCitator
URL	http://dev.investorstatelawguide.com/ResearchTools/FullTextSearch?id=FullTextSearch&tab=r&toc=content
Parameter	id
Attack	FullTextSearch
URL	http://dev.investorstatelawguide.com/ResearchTools/AgreementNavigator?id=AgreementNavigator&tab=r&toc=content
Parameter	id
Attack	AgreementNavigator
URL	http://dev.investorstatelawguide.com/ResearchTools/AgreementNavigator?id=262&tab=AgreementNavigator&toc=content
Parameter	tab
Attack	AgreementNavigator
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?toc=content&id=201&cat=arbitrationrules&subcat=ArticleCitator
Parameter	subcat
Attack	ArticleCitator
Instances	31
Solution	Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." For filenames, use stringent whitelists that limit the character set to be used. If feasible, only allow a single "." character in the filename to avoid weaknesses, and exclude directory separators such as "/". Use a whitelist of allowable file extensions. Warning: if you attempt to cleanse your data, then do so that the end result is not in the form that can be dangerous. A sanitizing mechanism can remove characters such as '.' and ';' which may be required for some exploits. An attacker can try to fool the sanitizing mechanism into "cleaning" data into a dangerous form. Suppose the attacker injects a '.' inside a filename (e.g. "sensi.tiveFile") and the sanitizing mechanism removes the character resulting in the valid filename, "sensitiveFile". If the input data are now assumed to be safe, then the file may be compromised.

	Inputs should be decoded and canonicalized to the application's current internal representation before being validated. Make sure that your application does not decode the same input twice. Such errors could be used to bypass whitelist schemes by introducing dangerous inputs after they have been checked. Use a built-in path canonicalization function (such as realpath() in C) that produces the canonical version of the pathname, which effectively removes ".." sequences and symbolic links. Run your code using the lowest privileges that are required to accomplish the necessary tasks. If possible, create isolated accounts with limited privileges that are only used for a single task. That way, a successful attack will not immediately give the attacker access to the rest of the software or its environment. For example, database applications rarely need to run as the database administrator, especially in day-to-day operations. When the set of acceptable objects, such as filenames or URLs, is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames or URLs, and reject all other inputs. Run your code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict which files can be accessed in a particular directory or which commands can be executed by your software. OS-level examples include the Unix chroot jail, AppArmor, and SELinux. In general, managed code may provide some protection. For example, java.io.FilePermission in the Java SecurityManager allows you to specify restrictions on file operations. This may not be a feasible solution, and it only limits the impact to the operating system; the rest of your application may still be subject to compromise.
Reference	http://projects.webappsec.org/Path-Traversal http://cwe.mitre.org/data/definitions/22.html
CWE Id	22
WASC Id	33

High (Medium)	**Cross Site Scripting (Reflected)**
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privil the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (c theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks esser compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execu code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containin web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accep HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malic link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send al arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often incl message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site malicious link sent via email), just simply view the web page containing the code.
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?AgreementID=%27+src%3Dhttp%3A%2F%2Fbadsite.com&ProvCode=Generally&aaprovid=76304&cat=arbitrationrules&docid=449&id=201&sort=&subcat=&toc=cc
Parameter	AgreementID
Attack	' src=http://badsite.com
Evidence	' src=http://badsite.com
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?AgreementID=907&ProvCode=Generally&aaprovid=76304&cat=%27+onMouseOver%3D%27alert%281%29%3B&docid=449&id=201&sort=&subcat=&toc=content
Parameter	cat
Attack	' onMouseOver='alert(1);
Evidence	' onMouseOver='alert(1);
URL	http://dev.investorstatelawguide.com/ResearchTools/IframreportGenerator?agreementID=javascript%3Aalert%281%29%3B
Parameter	agreementID
Attack	javascript:alert(1);
Evidence	javascript:alert(1);
URL	http://dev.investorstatelawguide.com/User/Welcome
Parameter	Search
Attack	" onMouseOver="alert(1);
Evidence	" onMouseOver="alert(1);
URL	http://dev.investorstatelawguide.com/Treaties/Index?toc=mainAnnot&cat=%22+onMouseOver%3D%22alert%281%29%3B&tab=
Parameter	cat
Attack	" onMouseOver="alert(1);
Evidence	" onMouseOver="alert(1);
Instances	5

Solution	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding mo and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanume characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, th modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser ma choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as spe opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more r versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. R any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inpu syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or move elsewhere.
Reference	http://projects.webappsec.org/Cross-Site-Scripting http://cwe.mitre.org/data/definitions/79.html
CWE Id	79
WASC Id	8

High (Medium)	**SQL Injection**
Description	SQL injection may be possible.
URL	http://dev.investorstatelawguide.com/Home/BindDropClient?id=40&ddlName=M&_=1490020535714+AND+1%3D1+--+
Parameter	_
Attack	1490020535714 AND 1=1 --
Instances	1
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Other information	The page results were successfully manipulated using the boolean conditions [1490020535714 AND 1=1 --] and [1490020535714 AND 1=2 --] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
CWE Id	89

CWE Id	89
WASC Id	19

High (Medium)	Remote OS Command Injection
Description	Attack technique used for unauthorized execution of operating system commands. This attack is possible when an application accepts untrusted input to build operating system commands in an insecure manner involving improper data sanitization, and/or improper calling of external programs.
URL	http://dev.investorstatelawguide.com/Home/EditNewNotePadUser?Continue=Y%7Ctimeout+%2FT+%7B0%7D
Parameter	Continue
Attack	Y|timeout /T {0}
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?toc=content&id=201&cat=arbitrationrules%3Bstart-sleep+-s+%7B0%7D&subcat=
Parameter	cat
Attack	arbitrationrules;start-sleep -s {0}
Instances	2
Solution	If at all possible, use library calls rather than external processes to recreate the desired functionality. Run your code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict which files can be accessed in a particular directory or which commands can be executed by your software. OS-level examples include the Unix chroot jail, AppArmor, and SELinux. In general, managed code may provide some protection. For example, java.io.FilePermission in the Java SecurityManager allows you to specify restrictions on file operations. This may not be a feasible solution, and it only limits the impact to the operating system; the rest of your application may still be subject to compromise. For any data that will be used to generate a command to be executed, keep as much of that data out of external control as possible. For example, in web applications, this may require storing the command locally in the session's state instead of sending it out to the client in a hidden form field. Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, consider using the ESAPI Encoding control or a similar tool, library, or framework. These will help the programmer encode outputs in a manner less prone to error. If you need to use dynamically-generated query strings or commands in spite of the risk, properly quote arguments and escape any special characters within those arguments. The most conservative approach is to escape or filter all characters that do not pass an extremely strict whitelist (such as everything that is not alphanumeric or white space). If some special characters are still needed, such as white space, wrap each argument in quotes after the escaping/filtering step. Be careful of argument injection. If the program to be executed allows arguments to be specified within an input file or from standard input, then consider using that mode to pass arguments instead of the command line. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Some languages offer multiple functions that can be used to invoke commands. Where possible, identify any function that invokes a command shell using a single string, and replace it with a function that requires individual arguments. These functions typically perform appropriate quoting and filtering of arguments. For example, in C, the system() function accepts a string that contains the entire command to be executed, whereas execl(), execve(), and others require an array of strings, one for each argument. In Windows, CreateProcess() only accepts one command at a time. In Perl, if system() is provided with an array of arguments, then it will quote each of the arguments. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." When constructing OS command strings, use stringent whitelists that limit the character set based on the expected value of the parameter in the request. This will indirectly limit the scope of an attack, but this technique is less important than proper output encoding and escaping. Note that proper output encoding, escaping, and quoting is the most effective solution for preventing OS command injection, although input validation may provide some defense-in-depth. This is because it effectively limits what will appear in output. Input validation will not always prevent OS command injection, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, when invoking a mail program, you might need to allow the subject field to contain otherwise-dangerous inputs like ";" and ">" characters, which would need to be escaped or otherwise handled. In this case, stripping the character might reduce the risk of OS command injection, but it would produce incorrect behavior because the subject field would not be recorded as the user intended. This might seem to be a minor inconvenience, but it could be more important when the program relies on well-structured subject lines in order to pass messages to other components. Even if you make a mistake in your validation (such as forgetting one out of 100 input fields), appropriate encoding is still likely to protect you from injection-based attacks. As long as it is not done in isolation, input validation is still a useful technique, since it may significantly reduce your attack surface, allow you to detect some attacks, and provide other security benefits that proper encoding does not address.
Reference	http://cwe.mitre.org/data/definitions/78.html https://www.owasp.org/index.php/Command_Injection
CWE Id	78
WASC Id	31

High (Medium)	Cross Site Scripting (Persistent)
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site

	they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.
URL	http://dev.investorstatelawguide.com/Home/BindDropClient?id=40&ddlName=M&_=1490020535714
Parameter	AddExistingMatterNew
Attack	</option><script>alert(1);</script><option>
Instances	1
Solution	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.
Other information	Source URL: http://dev.investorstatelawguide.com/Home/AddNewNotePadUser?Continue=Y
Reference	http://projects.webappsec.org/Cross-Site-Scripting http://cwe.mitre.org/data/definitions/79.html
CWE Id	79
WASC Id	8

Medium (Medium)	**Application Error Disclosure**
Description	This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?cat=arbitrationrules&id=201&subcat&toc=content
Evidence	HTTP 500 Internal server error
URL	http://dev.investorstatelawguide.com/Home/Register?Length=0
Evidence	HTTP 500 Internal server error
URL	http://dev.investorstatelawguide.com/Home/Undermaintenance
	HTTP 500 Internal server error

Evidence	HTTP 500 Internal server error
URL	http://dev.investorstatelawguide.com/Home/Feedback
Evidence	HTTP 500 Internal server error
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?id=11&toc=content&type
Evidence	HTTP 500 Internal server error
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?cat=arbitrationrules&docid=449&id=201&subcat&toc=content
Evidence	HTTP 500 Internal server error
Instances	6
Solution	Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.
Reference	
CWE Id	200
WASC Id	13

Medium (Medium)	Format String Error
Description	A Format String error occurs when the submitted data of an input string is evaluated as a command by the application.
URL	http://dev.investorstatelawguide.com/Home/FetchExistingComments?id=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%
Parameter	id
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/AddNewNotePadUser?Continue=Y
Parameter	__RequestVerificationToken
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/BindDropClient?id=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n
Parameter	id
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login? id=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%
Parameter	id
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?id=201&returnUrl=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25
Parameter	returnUrl
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login? id=201&returnUrl=%2FResearchTools%2FArticleCitator%3Fcat%3Darbitrationrules&subcat=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25
Parameter	subcat
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login? id=201&returnUrl=%2FResearchTools%2FArticleCitator%3Fcat%3Darbitrationrules&subcat=&toc=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25
Parameter	toc
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?returnUrl=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%
Parameter	returnUrl
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?returnUrl=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%
Parameter	returnUrl
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login? returnUrl=%2FResearchTools%2FSubjectNavigator%3Fid%3D50&tab=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%2
Parameter	tab
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s

URL	http://dev.investorstatelawguide.com/Home/Login?returnUrl=%2FResearchTools%2FTermsPhrases%3Fletter%3Da&searchType=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%
Parameter	searchType
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?returnUrl=%2FResearchTools%2FSubjectNavigator%3Fid%3D50&tab=r&toc=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%
Parameter	toc
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?returnUrl=%2FResearchTools%2FTermsPhrases%3Fletter%3Da&searchType=T&toc=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25
Parameter	toc
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?returnUrl=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%
Parameter	returnUrl
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?docid=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25
Parameter	docid
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?returnUrl=%2FResearchTools%2FJurisprudenceCitators%3Fid%3D11&type=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%2
Parameter	type
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?docid=3806&id=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n
Parameter	id
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?returnUrl=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%
Parameter	returnUrl
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Login?docid=3806&id=201&returnUrl=ZAP%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%25s%25n%2
Parameter	returnUrl
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
URL	http://dev.investorstatelawguide.com/Home/Feedback
Parameter	__RequestVerificationToken
Attack	ZAP%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s%n%s
Instances	147
Solution	Rewrite the background program using proper deletion of bad character strings. This will require a recompile of the background executable.
Other information	Potential Format String Error. The script closed the connection on a /%s
Reference	https://www.owasp.org/index.php/Format_string_attack
CWE Id	134
WASC Id	6

Low (Medium)	**Web Browser XSS Protection Not Enabled**
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://dev.investorstatelawguide.com/
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/Home/ClearSession
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/Home/iplogin
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/User/Welcome

Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/Scripts/lz-string.js
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/User/CheckAgreementNavigatorActive
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/NonMembers/NoticetoNotePadusers?Id=262
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/subscribers/LogPageCountForUserSession
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/subscribers/LogTheLogoutSessionTime
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/ResearchTools/SubjectNavigator?toc=content&id=50&tab=r
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?toc=content&id=201&cat=&subcat=
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?toc=content&id=201&cat=arbitrationrules&subcat=
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/Treaties/Index?toc=mainAnnot&cat=&tab=
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/sitemap.xml
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/Home/LogOut
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/NonMembers
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/ResearchTools
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/User
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/nonmember
Parameter	X-XSS-Protection
URL	http://dev.investorstatelawguide.com/nonmember/
Parameter	X-XSS-Protection
Instances	165
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
CWE Id	933
WASC Id	14

WASC Id	14

Low (Medium)	**X-Content-Type-Options Header Missing**
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://dev.investorstatelawguide.com/
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/Home/ClearSession
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/User/Welcome
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/Scripts/slidedoutmenu.js
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/Scripts/slideoutquicklinks.js
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/User/CheckAgreementNavigatorActive
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/NonMembers/NoticetoNotePadusers?Id=262
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/subscribers/LogPageCountForUserSession
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/subscribers/LogTheLogoutSessionTime
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/ResearchTools/SubjectNavigator?toc=content&id=50&tab=r
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/Scripts/tooltipbubble.js
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/JS/Search.js
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?toc=content&id=201&cat=&subcat=
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?toc=content&id=201&cat=arbitrationrules&subcat=
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/Treaties/Index?toc=mainAnnot&cat=&tab=
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/robots.txt
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/nonmember
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/nonmember/
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/nonmember/index
Parameter	X-Content-Type-Options
URL	http://dev.investorstatelawguide.com/nonMember/nonMember?Id=15

Parameter	X-Content-Type-Options
Instances	173
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
CWE Id	16
WASC Id	15

Low (Medium)	**Cookie No HttpOnly Flag**
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://dev.investorstatelawguide.com/Home/LogOut
Parameter	UserId
Evidence	Set-Cookie: UserId
URL	http://dev.investorstatelawguide.com/Home/LogOut
Parameter	Password
Evidence	Set-Cookie: Password
URL	http://dev.investorstatelawguide.com/Home/LogOut
Parameter	ReUserId
Evidence	Set-Cookie: ReUserId
URL	http://dev.investorstatelawguide.com/Home/LogOut
Parameter	RePassword
Evidence	Set-Cookie: RePassword
URL	http://dev.investorstatelawguide.com/User/Welcome
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?toc=content&id=201&cat=arbitrationrules&subcat=
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/Home/Login?Logout=Yes&autologin=n
Parameter	UserId
Evidence	Set-Cookie: UserId
URL	http://dev.investorstatelawguide.com/Home/Login?Logout=Yes&autologin=n
Parameter	Password
Evidence	Set-Cookie: Password
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?id=11
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/ArticleCitator?cat=arbitrationrules&docid=449&id=201&subcat&toc=content
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=555266&disputeID=555266&id=11&sort&type
Parameter	

	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=518332&disputeID=518332&id=11&sort&type
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=615293&disputeID=615293&id=11&sort&type
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=514309&disputeID=514309&id=11&sort&type
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=515924&disputeID=515924&id=11&sort&type
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=516177&disputeID=516177&id=11&sort&type
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=937966&disputeID=937966&id=11&sort&type
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=517781&disputeID=517781&id=11&sort&type
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=705056&disputeID=705056&id=11&sort&type
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
URL	http://dev.investorstatelawguide.com/ResearchTools/JurisprudenceCitators?cidsp=526125&disputeID=526125&id=11&sort&type
Parameter	UserCIS
Evidence	Set-Cookie: UserCIS
Instances	77
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
CWE Id	16
WASC Id	13

Low (Medium)	**Password Autocomplete in Browser**
Description	The AUTOCOMPLETE attribute is not disabled on an HTML FORM/INPUT element containing password type input. Passwords may be stored in browsers and retrieved.
URL	http://dev.investorstatelawguide.com/Home/Register?Length=0
Parameter	Password
Evidence	<input class="textbox" id="Password" maxlength="150" name="Password" type="password" />
URL	http://dev.investorstatelawguide.com/Home/Register?Length=0
Parameter	ConfirmPassword
Evidence	<input class="textbox" id="ConfirmPassword" maxlength="150" name="ConfirmPassword" type="password" />
URL	http://dev.investorstatelawguide.com/Home/Login?Logout=Yes&autologin=n
Parameter	Password
Evidence	

	<input class="textbox" id="Password" name="Password" type="password" value="" />
URL	http://dev.investorstatelawguide.com/Admin/AdminLogin
Parameter	PassWord
Evidence	<input class="login-inp" id="PassWord" name="PassWord" type="password" />
Instances	4
Solution	Turn off the AUTOCOMPLETE attribute in forms or individual input elements containing password inputs by using AUTOCOMPLETE='OFF'.
Reference	http://www.w3schools.com/tags/att_input_autocomplete.asp https://msdn.microsoft.com/en-us/library/ms533486%28v=vs.85%29.aspx
CWE Id	525
WASC Id	15