

# **Application Penetration Test – Technical Report**

**Destination Hotels & Resorts / Project Number: OP-10707**

**Revision Number: 1.0**

**Date: 04/18/2014**

## Table of Contents

<b>Executive Summary.....</b>	<b>3</b>
Scope .....	3
<b>Findings and Recommendations .....</b>	<b>4</b>
Finding Composition.....	4
<i>Definition of a Technical Finding.....</i>	<i>4</i>
<i>Description of Findings Groups.....</i>	<i>4</i>
<i>Raw Data and Evidentiary Support .....</i>	<i>4</i>
Findings Ranking System .....	5
Severity Categories.....	5
<b>Findings Matrix .....</b>	<b>6</b>
<b>Assessment Findings .....</b>	<b>7</b>
Application Technical .....	7
<i>Blind SQL Injection .....</i>	<i>7</i>
<i>Reflected Cross Site Scripting .....</i>	<i>9</i>
<i>Insecure Session Cookies - HttpOnly flag and Secure flag.....</i>	<i>12</i>
Environment and Configuration.....	15
Secure Channel Enforcement Issues.....	15
Directory Indexing Enabled .....	18
<b>Appendix A – Project Information.....</b>	<b>20</b>
Assessment Project Team.....	20
<b>Appendix B – Methodology Overview .....</b>	<b>21</b>
Application Profiling .....	21
Threat Analysis .....	21
Dynamic Testing.....	21

# Executive Summary

Destination Hotels & Resorts (Destination Hotels) engaged Accuvant LABS to perform a security assessment of the organization's Gant Aspen application and supporting application environment and infrastructure. This report details the assessment of the application and supporting environment as of May 2014. The objective was to assess the current security posture and the effectiveness of the controls in place within the application environment, compare the results of the assessment with industry best practices and identify vulnerabilities that could negatively affect the application or business as a whole.

It is important to note that this report represents a snapshot of the security of the environment assessed at a point in time. Conditions may have improved, deteriorated or remained the same since this assessment was completed.

Accuvant LABS knows the importance Destination Hotels & Resorts places on data security and sincerely appreciates the opportunity to have worked with Destination Hotels on this engagement. Should you have any questions regarding these findings or the contents of this report, please feel free to contact us.

## Scope

Destination Hotels & Resorts identified the following application components for this security assessment:

- Gant Aspen Application

The following application components were out of scope for this engagement:

- Legacy systems and underlying infrastructure components
- Underlying shared service infrastructure components
- All other Destination Hotels applications and systems

Project Scope Details	
Application Name and Version	Gant Aspen Production release
Engagement Length	3 days
Application Access	Working instance of the application
Consultant Location	Testing was performed remotely via Internet.
Environment	Production environment.
User Roles	1 – Anonymous User

The phases and associated sub-components of this assessment included:

Assessment Phase	Tasks
Application Profiling	Through a review of available documentation, runtime analysis and developer interviews, the assessment team created a profile of the application security model and its key functionality.
Threat Analysis	The assessment team documented critical data held by the application and likely attacker goals based on input from the application owners and the assessment team's experience.
Dynamic Testing	The assessment team executed manual testing procedures, including unauthenticated and authenticated test scenarios within the user roles defined for the engagement. The application execution environment was also tested using a comprehensive suite of application and network assessment tools and manual verification, to identify common vulnerabilities. Where vulnerabilities were found, the assessment team created a proof of concept to demonstrate the issue and provided reproduction information for the development staff.

# Findings and Recommendations

---

Because of this assessment, Accuvant LABS identified a number of areas where security controls could be improved, augmented or refined. The remainder of this report describes the details of Accuvant LABS' observations regarding security vulnerabilities and/or control deficiencies, the severity associated with the issues identified and recommendations for resolving those issues.

Accuvant LABS recommends that Destination Hotels & Resorts developers first test the recommended changes to ensure that they do not adversely affect application functionality.

## Finding Composition

### Definition of a Technical Finding

Technical findings in this document each represent a class of security vulnerabilities identified during testing. Findings are grouped based on common root causes. For example, if there were 10 unique URLs vulnerable to SQL injection in an application, a single SQL injection finding would be documented with general remediation steps. Within the finding, each instance of SQL injection vulnerability would be enumerated.

### Description of Findings Groups

The findings groups are described as follows:

- **Application Technical Findings** – Application technical findings represent issues within the application that directly relate to a confirmed or potential attack vector. In these findings, the application implementation deviates from best practices for secure application design or development. Exploitation may result in violation of data integrity, application availability or data confidentiality.
- **Application Architecture Findings** – Application architecture findings relate to specific application design components that do not align with best practices. These issues are not specific vulnerabilities in the application, but can increase the attack surface, increase the likelihood of the presence of an attack vector or elevate the severity of existing attack vectors. The degree of deviation from appropriate controls and the potential impact to the application determines the severity level.
- **Environment and Configuration Findings** – Environment and configuration findings deal with issues that weaken the security posture of the application's supporting hosts and services. This includes findings related to the patch level of exposed hosts and services. This group also contains findings in system and service configurations that deviate from industry best practices and company policies. The existence of direct attack vectors for some issues in this category determines the severity level.

## Raw Data and Evidentiary Support

Throughout each phase of the assessment, a variety of tools, utilities, scripts and processes were leveraged for the identification, analysis and testing of the assets targeted. An overview of the assessment methodologies used during the engagement is available in Appendix B.

The raw data output, tool reports, proof of concept information and custom testing scripts used when analyzing the application and generating the findings are available in a compressed archive provided as a deliverable supplement to this document.

## Findings Ranking System

In order to prioritize the assessment results, each finding was categorized based on severity classifications. Final analysis of the risk or impact to the application will require an internal evaluation by Destination Hotels & Resorts personnel. Accuvant LABS has developed classifications using the severity nomenclature for ranking the issues identified within the various severity categories.

### Severity Categories

Based on Accuvant LABS' analysis of the particular finding and assets affected, a finding will fall into one of the following severity level categories:



**Severity – Critical:** Critical vulnerabilities require an immediate response through mitigating controls, direct remediation or a combination thereof. Exploitation of critical severity vulnerabilities results in privileged access to the target system, application or sensitive data and enables further access to other hosts or data stores within the environment. Findings with a critical ranking will cause significant losses when they are exploited, although the total cost is difficult to quantify in advance. In general, a critical severity ranking is warranted when the issue has a direct impact on regulatory or compliance controls imposed on the environment, accesses personally identifiable information (PII) or financial data or could cause significant reputational or financial harm.



**Severity – High:** Findings with a high severity ranking require immediate evaluation and subsequent resolution. Exploitation of high severity vulnerabilities leads directly to an attacker gaining privileged, administrative-level access to the system, application or sensitive data. However, it does not enable further access to other hosts or data stores within the environment. If left unmitigated, high severity vulnerabilities can pose an elevated threat that could affect business continuity or cause significant financial loss.



**Severity – Medium:** A finding with a medium severity ranking requires review and resolution within a short time. From a technical perspective, vulnerabilities that warrant a medium severity ranking can lead directly to an attacker gaining non-privileged or user-level access to the system, application or sensitive data. Findings that can cause a denial-of-service (DoS) condition on the host, service or application are also classified as medium risk. Alternately, the vulnerability may provide a way for attackers to gain elevated levels of privilege. From a less technical perspective, observations with this ranking are significant, but they do not pose as much of a threat as high or critical severity exposures.








**Severity – Low:** Low severity findings should be evaluated for review and resolution once the remediation efforts for critical, high and medium severity issues are complete. From a technical perspective, vulnerabilities that warrant a low severity ranking may leak information to unauthorized or anonymous users used to launch a more targeted attack against the environment. From a process perspective, observations with this ranking provide awareness and should be addressed over time as part of a comprehensive information security program, but do not presently pose a substantial threat to business operations or have any significant loss associated with the exposure.



**Informational:** An informational finding presents no direct threat to the confidentiality, integrity or availability of the data or systems supporting the environment. These issues pose an inherently low threat to the organization and any proposed resolution should be considered as an addition to the information security procedures already in place.

# Findings Matrix

The table below provides a summary of the assessment findings categorized by group and ranked by severity. The table provides Destination Hotels & Resorts with an overview of all of the findings from the assessment and allows the remediation team to focus efforts on the areas of highest severity as determined by Accuvant LABS. Click the individual link below to go directly to that finding.

Finding Category	Severity
Application Technical	
<a href="#">Blind SQL Injection</a>	
<a href="#">Reflected Cross Site Scripting</a>	
<a href="#">Insecure Session Cookies - HttpOnly flag and Secure flag</a>	
Environment and Configuration	
<a href="#">Secure Channel Enforcement Issues</a>	
<a href="#">Directory Indexing Enabled</a>	

# Assessment Findings

## Application Technical

Application technical findings represent issues within the application that directly relate to a confirmed or potential attack vector. Issues in this section describe instances where the application implementation deviates from best practices for secure application design or development. Exploitation may result in violation of the application's security policy, data integrity, application availability or data confidentiality.



### Blind SQL Injection

**Observation:** The application is vulnerable to SQL injection because it takes client side input values and embeds them into SQL statements without sufficient validation. This may allow injection of arbitrary commands or unauthorized access to data.

**Reproduction:** The following request can be used to reproduce this issue. Blind SQL injection attack values in the `object_id` parameter result in widely different response times from normal parameter values, which indicates the parameter is vulnerable to blind SQL injection attacks.

Shows a normal request, using 123 as the value for `object_id`. Notice the time the request takes 0.263 seconds:

```
$ time curl -IL "http://www.gantaspen.com/?geo_mashup
content=render-map&map_content=global&map_data_key=aeaa23e2e8f050b717ada3597262
51d&name=gm-map-1&object_id=618&object_ids=123&width=100%"

HTTP/1.1 200 OK
Date: Fri, 04 Apr 2014 19:32:44 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.28
X-Pingback: http://www.gantaspen.com/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=UTF-8

real    0m0.263s
user    0m0.010s
sys     0m0.000s
```

Figure 1 - Normal GET request

Shows a request with the blind SQL attack `123%20AND%20SLEEP(1)` inside the `object_id` parameter. Notice the request takes 49.491 seconds.

```
$ time curl -IL "http://www.gantaspen.com/?geo_mashup_
content=render-map&map_content=global&map_data_key=aeaa23e2e8f050b717ada3597262f
```

```
51d&name=gm-map-1&object_id=618&object_ids=123%20AND%20SLEEP(1) &width=100%"

HTTP/1.1 200 OK
Date: Fri, 04 Apr 2014 20:03:15 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.28
X-Pingback: http://www.gantaspen.com/xmlrpc.php
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=UTF-8

real    0m49.491s
user    0m0.010s
sys     0m0.000s
```

Figure 2 - GET request with a blind SQL Injection attack

**Severity – High:** Failing to validate user-supplied input prior to passing information to the database exposes the database to attackers to attack via malicious input that is used by the application in SQL queries. Potential impact may include compromise of confidential data stored on internal database systems, exposure of customer usernames and passwords, or system level exploitation. Attackers use SQL injection vulnerabilities to circumvent perimeter security mechanisms and gain access to backend database systems that are typically not available to the Internet. This presents great risk to internal assets that communicate with web-facing components.

SQL injection is in the Open Web Application Security Project (OWASP) top 10 vulnerabilities and the Payment Card Industry Data Security Standard (PCI-DSS) section 6.5.2.

**Attack Scenario:** The most obvious attack scenario involves using SQL injection to steal or corrupt the vulnerable databases. Attackers can use sophisticated SQL statements to:

- Extract data from the database
- Bypass login pages and other access controls
- Modify or destroy data within the database
- Escalate privileges within the database server
- Gain access to other database servers on the internal network
- Access other internal network services such as HTTP, FTP, DNS and SMTP through the database server
- Upload and download files on the database server's host OS file system
- Execute commands on the database server's host operating system
- Embed back doors within the database by using the DBMS's scheduling, triggers, or run-on-startup features

**Recommendation:** Review all areas of the application that accept user input and form SQL queries and ensure that prepared statements are in place. Since SQL injection vulnerabilities are a result of lexical parsing issues for the constructed query strings, prepared statements are considered secure from attack. The following are general principles to consider in securing an application from SQL injection:

### 1. Inclusion Validation not Exclusion

Do not create a list of characters to exclude, or a "bad characters" list. An "inclusion" list of allowable characters or words is much simpler and more effective in most cases. Regular expressions are an excellent means to control placement and quantity of characters as well.



## 2. Validate by Data Type

The more specific your inclusion type validation is the more effective it will protect against injection based attacks. For example, if a field is designed to only accept integers, only the digits 0-9 should be allowed with no leading zeros. For some types, such as first and last names, building extremely specific filters may be difficult, and your filter may allow some potentially dangerous meta-characters such as single-quotes. For dangerous meta-characters, the content should be properly escaped before being passed onto the business logic layer.

## 3. Validate, Prepare and Escape by Destination

Data should be validated and prepared based on the components of the application that will be processing it. For example, data that is going to a SQL Server query should be filtered for certain values, and some things like single-quotes, must be escaped if they are allowed at all. Data that is going to an XML file or an XPath statement should be filtered for other values. In each case, analyze the control characters of the destination language and ensure that these characters are properly removed or escaped.

## 4. Prepared Statements

The use of prepared statements (parameterized queries) is how database queries should be written. They are simple to write, and easier to understand than dynamic queries. Parameterized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied.

Prepared statements ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker. If an attacker were to enter the userID of tom' or '1'='1, the parameterized query would not be vulnerable and would instead look for a username which literally matched the entire string tom' or '1'='1.

Data to be displayed as HTML should be HTML encoded (i.e. replace '<' and '>' with &lt; and &gt;, etc.) to prevent cross-site scripting by escaping meta-characters that are interpreted by the user's browser.

Understanding the source and prior filtering of data is also important in identifying what controls need to be applied to data before passing it into a component of the application. For example, data that is stored in the database may not have been filtered for HTML or cross-site-scripting safety. If the data was submitted to the database by an attacker, it may contain cross-site scripting in HTML tags that were not filtered out. These types of issues are often a problem with web-based message boards. When the data is retrieved from the database, it should be HTML escaped prior to being written to the response page.

### Asset(s) Affected:

- [http://www.gantaspen.com/?geo\\_mashup\\_content=render-map](http://www.gantaspen.com/?geo_mashup_content=render-map) [object\_ids parameter]



## Reflected Cross Site Scripting

**Observation:** Reflected attacks are those where the injected code is reflected off the web server, such as in an error message, search result or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other web server. When a user is tricked into clicking on a malicious link or submitting a specially crafted form, the injected code travels to the vulnerable web server, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server.

### Proof of Concept:

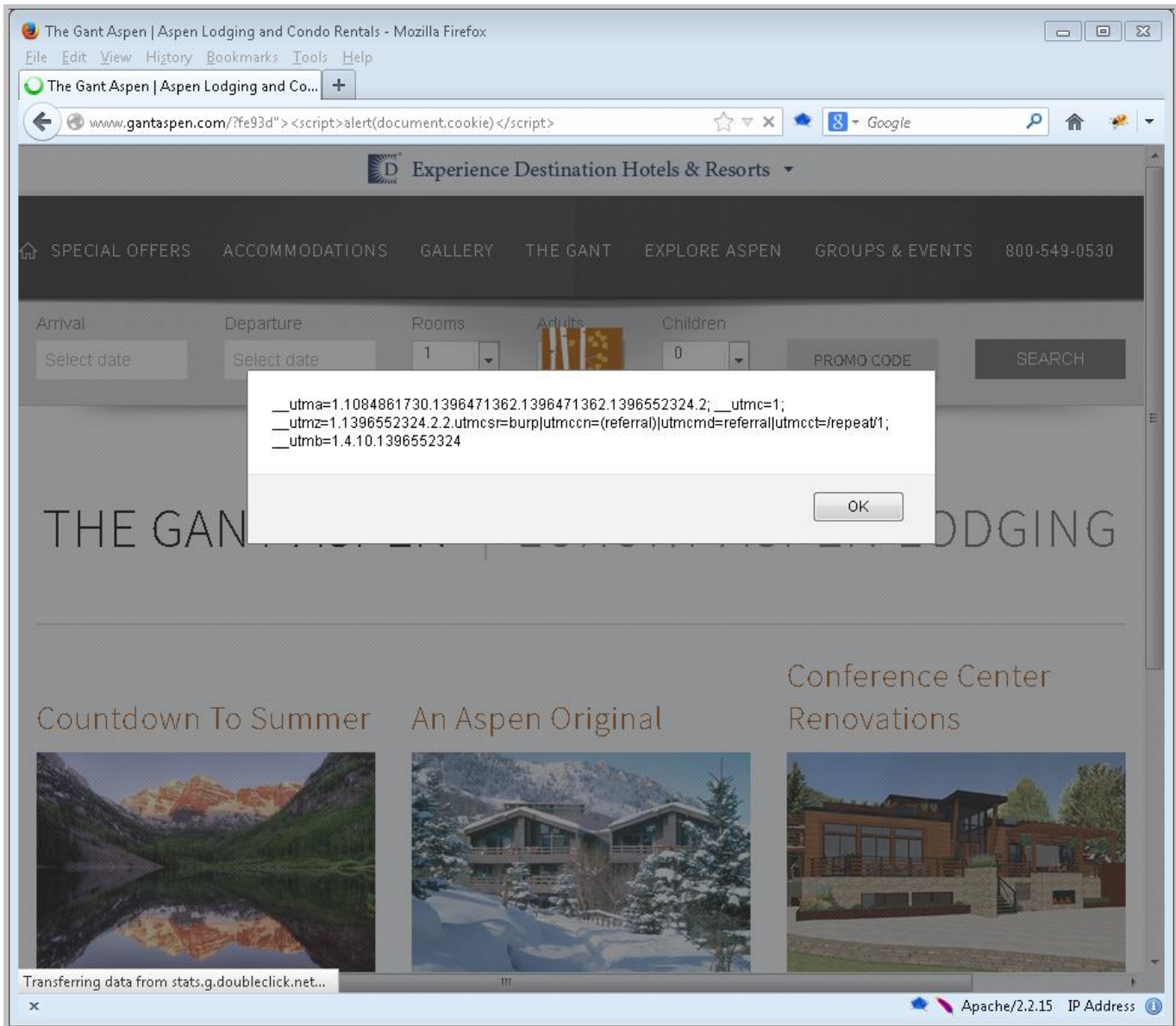


Figure 3 - Reflected XSS attack showing the current application cookies

### Reproduction:

Send the following POST request to the application:

```
GET /?fe93d"><script>alert(document.cookie)</script> HTTP/1.1
Host: www.gantaspen.com
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

```
Cookie: __utma=1.1084861730.1396471362.1396471362.1396471362.1; __utmc=1;
__utmz=1.1396471362.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none)
```

The ">" breaks out of the "<a href="">" tag on the application page in question, and then runs the code inside of the "<script>" tag, which in this case displays the user's cookies in a popup window, as shown in Figure 3.

**M Severity – Medium:** XSS vulnerabilities can occur when data enters a web application through an untrusted source, most frequently a web request, and the data is subsequently included in dynamic content sent to a web user without validation. XSS attacks occur when an attacker uses a web application to send malicious content, generally in the form of a browser-side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user in the output it generates without validating or encoding it.

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. Because the malicious content appears to come from a trusted source, the end user's browser has no way to know that it should not be trusted, and will execute any script or code within. The malicious code can access cookies, session tokens, or other sensitive information retained by the user's browser and used on that site. Malicious users can use XSS vulnerabilities to rewrite the contents of the HTML page.

Using a XSS attack, an attacker could code a remote attack that would allow the attacker to compromise a valid user's session. If successful, the attacker could retrieve the valid user's session cookies and present them to the application as that user. The attacker could then interact with the website posing as the valid user. Using JavaScript, ActiveX and Java code, the attacker could also target the valid user's own PC.

Reflected attacks are those where the injected code reflects off the web server, such as in an error message, search result or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an email message or on some other web server. When a user is tricked into clicking on a malicious link or submitting a specially crafted form, the injected code travels to the vulnerable web server, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server. Reflected XSS attacks are the most common form of XSS attacks in web-based applications.

XSS is listed on the Open Web Application Security Project (OWASP) Top 10 vulnerabilities and in the Payment Card Industry Data Security Standard (PCI-DSS), section 6.5.X. XSS is also one of the most prevalent issues in modern web applications with SQL Injection. XSS, in many cases, is easily identified using both "black-box" (no access to source code) and "white-box" (with full access to the application environment including source code) testing methodologies. As a result, XSS is one of the most high profile and prevalent application security flaws seen in both web-based and non-web based applications.

**Attack Scenario:** The most severe XSS attacks involve disclosure of the user's session cookie, allowing an attacker to hijack the user's session and take over the account. Other damaging attacks include disclosure of end user files, installation of Trojan horse programs, redirecting the user to another page or site or modifying the presentation of content.

**Recommendation:** Preventing XSS involves both checking user-supplied values and encoding values displayed back to the user. The goal is to ensure that the application only encounters expected values and to delineate user-supplied data versus HTML or scripting constructs.

Input validation can be performed using inclusion (whitelisting) and exclusion (blacklisting) filtering. Exclusion filters block known bad data such as script tags. One drawback of exclusion filters is that it is difficult to predict all attack scenarios. Inclusion filters, on the other hand, restrict data to known good values. Inclusion filters can be implemented using regular expressions and/or strong type casting.

Deciding what type of encoding to perform depends on the circumstances where the data is returned. Values returning a URL should utilize URL encoding and values returning straight HTML should leverage HTML encoding routines to escape dangerous characters. However, values returning attributes of HTML tags or XML may require a different approach. A simple solution is use of an encoding library designed to prevent XSS, such as Reform (part of the OWASP Encoding Project), which provides methods for a variety of situations where data is returned to the end user.

In vulnerable applications, an attacker can inject harmful code within dynamically constructed HTML, or dynamically constructed JavaScript blocks. All dynamically generated content or code must sanitize special characters or use a whitelist of safe values.

Avoid reflected XSS vulnerabilities in an application by implementing the following procedures:

- Validate all user-supplied input. Where possible, allow only alphanumeric characters. In general, use whitelists rather than blacklists, and apply the strictest validation possible.
- Identify all application code that reflects user-originated data back to the end-user and transform all printable non-alphanumeric characters into their corresponding HTML named or numeric entities before rendering the data in the response page.
- Carefully design application components or features that could allow an anonymous user to post input to the application. Adherence to application security best practices will eliminate XSS vulnerabilities before introduction into the application environment.

#### **Resources:**

- General Resources:
  - [OWASP Article on XSS](#)
  - [OWASP XSS Prevention Cheat Sheet](#)
- XSS Testing
  - [OWASP Reviewing Code for XSS](#)
  - [OWASP Testing for XSS Vulnerabilities in an Application](#)
  - [XSS Test Cases](#)

#### **Asset(s) Affected:**

1. <http://www.gantaspen.com/> ['width' parameter]
2. <http://www.gantaspen.com/> ['p' parameter]
3. <http://www.gantaspen.com/> ['s' parameter]
4. <http://www.gantaspen.com/> [arbitrary parameters]



---

## Insecure Session Cookies - HttpOnly flag and Secure flag

---

**Observation:** The application does not utilize cookies with the Secure flag set in some instances. The Secure flag within a cookie requires that SSL be present when passing the cookies back and forth between the client and server. It was also observed that the application does not always use the HttpOnly cookie flag.

Utilizing the Secure flag of a cookie ensures that the user stays in SSL and supports a defense-in-depth strategy. If the HttpOnly attribute is set on a cookie, client-side JavaScript cannot read or set the cookie's value. This measure can prevent certain client-side attacks, such as cross-site scripting, from trivially capturing the cookie's value via an injected script.

Note: Until the web application utilizes HTTPS throughout, using the 'Secure' flag will break the application, as cookies with the 'Secure' flag set will not be allowed to transmit the network.

**Reproduction:** Interact with the application through an intercepting proxy and note that cookies are sent without the appropriate security flags. Figure 4 shows a screenshot and the cookie flag states:



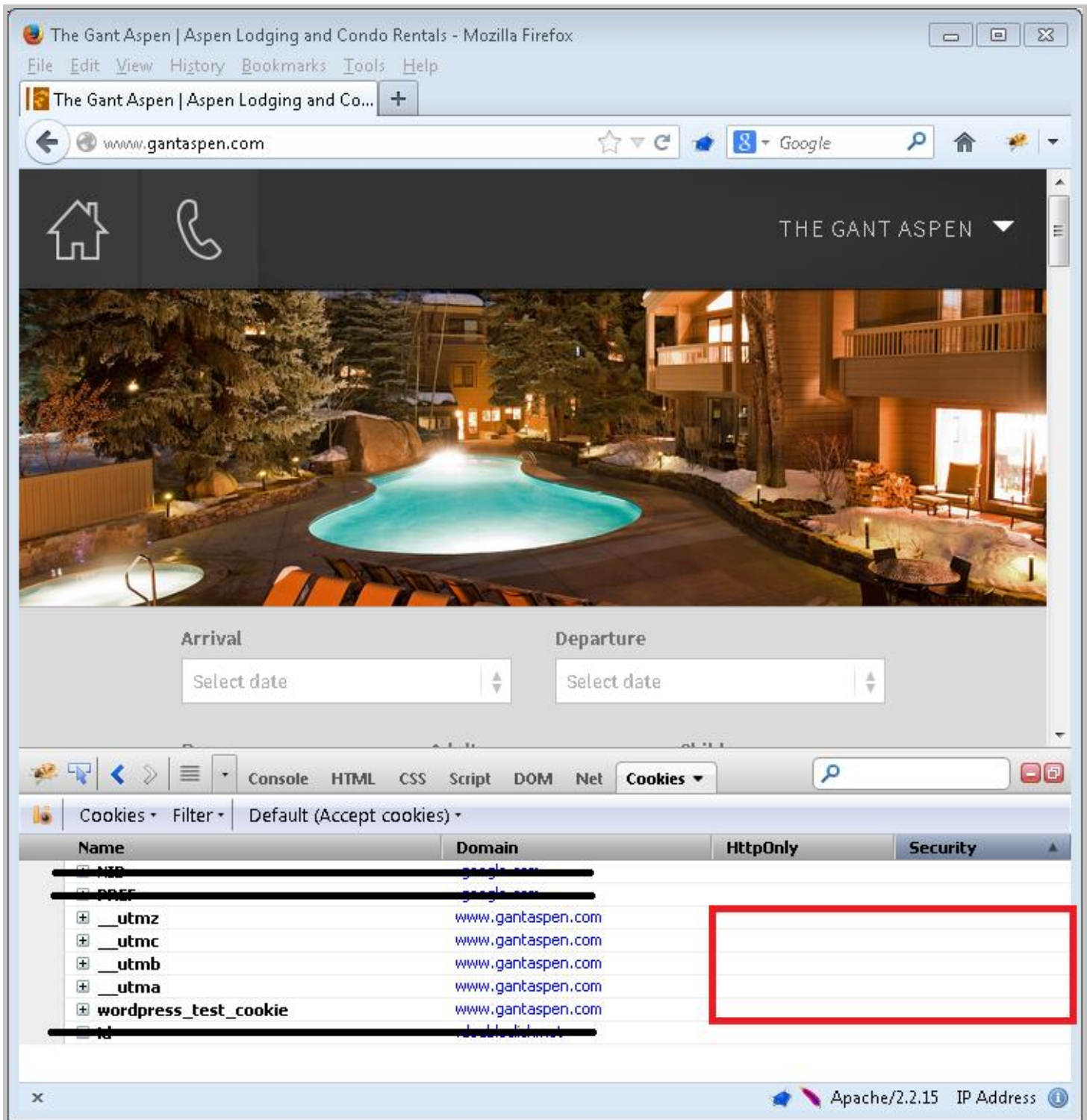


Figure 4 - Screenshot showing cookies missing important flags

The following are a few of the cookies that were issued by the application that did not have both the HttpOnly flag set and the Secure flag set.

Name	Domain	HttpOnly Flag	Secure Flag
__utmz	<a href="http://www.gantaspen.com">www.gantaspen.com</a>	No	No
__utmc	<a href="http://www.gantaspen.com">www.gantaspen.com</a>	No	No
__utmb	<a href="http://www.gantaspen.com">www.gantaspen.com</a>	No	No
__utma	<a href="http://www.gantaspen.com">www.gantaspen.com</a>	No	No
wordpress_test_cookie	<a href="http://www.gantaspen.com">www.gantaspen.com</a>	No	No

**Severity – Low:** A common way of abusing a cross-site scripting vulnerability is to obtain a user's session variables. This is done by executing JavaScript in the user's browser to send cookie values to an attacker-controlled system. The HttpOnly flag prevents client-side scripting such as JavaScript from accessing the corresponding cookie value.

Additionally, when the Secure flag is not set, it may be possible for an attacker to trick an end-user into visiting a non-SSL version of the site, causing the cookies to be sent in the clear.

**Recommendation:** Modify the application settings to require the HttpOnly and Secure flags. As with any application change, perform testing and reevaluate typical use cases.

#### Resources:

- OWASP HTTPOnly: <http://www.owasp.org/index.php/HTTPOnly>
- OWASP Secure Flag: <https://www.owasp.org/index.php/SecureFlag>
- Setting the Secure Flag is easy, <http://enablesecurity.com/2008/08/29/setting-the-secure-flag-in-the-cookie-is-easy/>

## Environment and Configuration

Environment and configuration findings deal with issues that weaken the security posture of the application's supporting hosts and services. This includes findings related to the patch level of exposed hosts and services. This group also contains findings in system and service configurations that deviate from industry best practices and company policies. The existence of direct attack vectors for some issues in this category determines the severity level.



### Secure Channel Enforcement Issues

**Observation:** The application transmits all traffic including user credentials, session tokens, and potentially other sensitive information over an unencrypted channel (HTTP).

Passwords, session tokens, and other sensitive data submitted over an unencrypted connection are vulnerable to capture by an attacker who is suitably positioned on the network. This includes any malicious party located on the network, and within the application's hosting infrastructure. Even if switched networks are employed at some of these locations, techniques exist to circumvent this defense and monitor the traffic passing through switches. Without encrypting the entire session or allowing users to only access the site via secure HTTPS implementations, it is impossible to guarantee with a reasonable level of certainty that an attack is not taking place on an insecure network.

**Proof of Concept / Reproduction:** Interact with the application and note that it does not make use of or require HTTPS. Also note that the wp-login.php page contains the following login form action, which is explicitly directed to a non-HTTPS endpoint.

```
<form name="loginform" id="loginform" action="http://www.gantaspen.com/wp-login.php" method="post">
  <p>
    <label for="user_login">Username<br />
    <input type="text" name="log" id="user_login" class="input" value="" size="20"
  /></label>
  </p>
  <p>
    <label for="user_pass">Password<br />
    <input type="password" name="pwd" id="user_pass" class="input" value="" size="20"
  /></label>
  </p>
  <p class="forgetmenot"><label for="rememberme"><input name="rememberme"
type="checkbox" id="rememberme" value="forever" /> Remember Me</label></p>
  <p class="submit">
    <input type="submit" name="wp-submit" id="wp-submit" class="button button-primary
button-large" value="Log In" />
    <input type="hidden" name="redirect_to" value="http://www.gantaspen.com/wp-admin/" />
    <input type="hidden" name="testcookie" value="1" />
  </p>
</form>
```

Figure 5 - Login form located at wp-login.php

**Severity – High:** Passing credential handling tokens, such as session cookies, in plain text format may allow an attacker to hijack a legitimate user's session. Not transferring data in SSL mode may allow an attacker to intercept user data including password information. By leveraging other methods, such as attacking clients at the network layer through man-in-the-middle attacks, a user may intercept the credential information over a local area network and perform session hijacking.

Issues with the handling of secure communication channels may result also in non-compliance with industry standards or regulatory requirements. The PCI Data Security Standard (PCI-DSS) specifically requires the use of secure communication channels for transmission of data over public networks.

The presence of other attack vectors in many cases will greatly amplify the risk of session hijacking in particular within a web application. Using modern attack vectors such as [Man-in-the-middle](#) (MITM) or [SSL stripping](#) attacks access to a live session token may be simplified. In many cases an organization has no control over where an application will be used at, such as a public wireless hotspot or on a compromised PC. Public wireless hotspots are highly susceptible to both sniffing and MITM attacks that allow access to live session tokens in many cases without the knowledge of the user. Compromised systems can also be used to harvest live session tokens or credentials from an application. This method of attack is referred to as a [Man-in-the-browser](#) attack where a Trojan or compromised application component (such as a browser itself) is used to capture sensitive information.



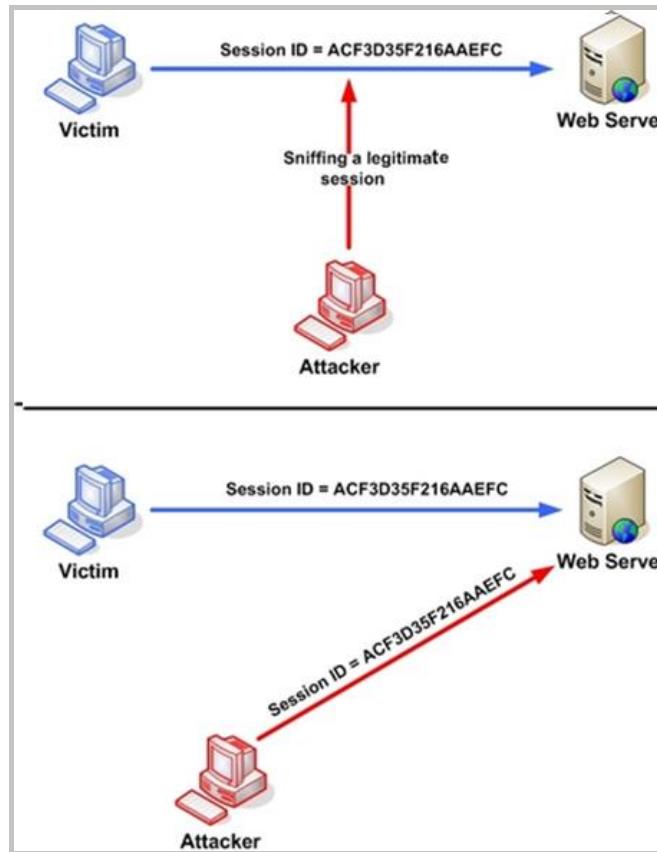


Figure 6 - An overview of session hijacking via a man-in-the-middle attack

**Recommendation:** The following controls should be implemented at a minimum:

- Disable plaintext HTTP interfaces on the affected web servers, or at a minimum make them redirect to HTTPS.
- Require all session cookies be set with the *Secure* flag (see the *\_finding* for more details). This prevents session tokens from being transmitted over HTTP.

Restrict all web traffic to SSL only for applications transmitting sensitive information by implementing the following steps:

- Create a redirection directive inside of the web server configuration to redirect users to SSL mode regardless of the entry point URL to the application.
- Modify application behavior to force the usage of SSL throughout the entire application by implementing a global redirect to HTTPS mode in the server configuration.
- Modify application behavior to provide session cookies using the *Secure* flag to prevent plaintext disclosure of session information.
- Modify all HTML form actions to directly reference HTTPS in the script path.

#### References:

- OWASP
  - [OWASP Session Management Category](#)
  - [Session Hijacking](#)
  - [Man-in-the-middle Attack](#)

- [Man-in-the-browser Attack](#)
- [Session Prediction](#)
- [Insecure Communications](#)
- PCI-DSS
  - [PCI-DSS 6.5.7 - Broken Authentication and Session Management](#)
  - [PCI-DSS 6.5.X - Insecure Transmissions](#)
- PA-DSS
  - [PA-DSS 5.2.7 - Broken Authentication and Session Management](#)
  - [PA-DSS 5.2.X - Insecure Transmissions](#)

**Asset(s) Affected:**

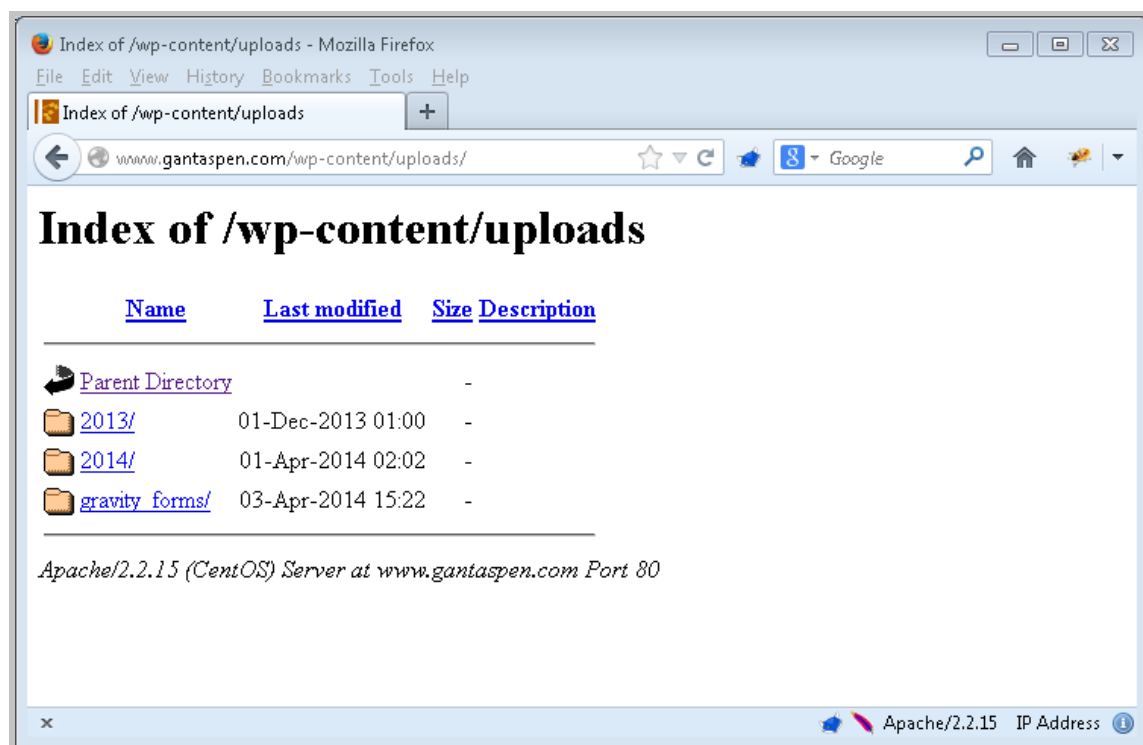
- <http://www.gantaspen.com/wp-login.php>



## Directory Indexing Enabled

**Observation:** The Apache web application server does not properly restrict directory indexes from being displayed to users, which may contain sensitive files.

**Reproduction:** See image below:



**Figure 7 - Directory listing of the /wp-content/uploads directory**

**Severity – Informational:** While the potential for revealing sensitive information can be high, given the viewable directory listings are part of an existing WordPress installation, and viewing source code files is impossible with the current Apache configuration, items such as database passwords, administrative or user account passwords, or other sensitive information leakage appears to not exist.

**Recommendation:** Disallow directory indexing inside the Apache configuration.

The “Indexes” keyword controls the display of directory indexes. When placed inside the Apache configuration with a “-” in front of the keyword, as shown in the snippet below:

```
Options -Indexes
```

Directory indexes will be disabled globally.

If disabling indexes per directory is required, the following example configuration snippet, which does not specify the “Indexes” keyword at all, will disable directory indexes for the ‘/’ directory only:

```
<Directory />
  Options FollowSymLinks
  AllowOverride None
</Directory>
```

**Asset(s) Affected:**

- [https://www.gantaspen.com/wp-content/\\*](https://www.gantaspen.com/wp-content/*)
- [https://www.gantaspen.com/wp-includes/\\*](https://www.gantaspen.com/wp-includes/*)

## Appendix A – Project Information

---

This section provides an overview of the Accuvant team members assigned to the project.

### Assessment Project Team

The tables below contain a listing of the Accuvant personnel involved in this engagement. Feel free to contact the appropriate persons at any time to discuss the results of the engagement.

Name:	<b>Sean McAllister</b>
Title:	Consultant – Accuvant LABS
Phone:	Mobile: +1.323.638.7831
E-mail Address:	smcallister@accuvant.com
Project Involvement:	Served as the project lead and performed technical testing and analysis. Direct questions regarding the deliverable here.

Name:	<b>John Bock</b>
Title:	Director –Software Security Group – Accuvant LABS
Phone:	Mobile: +1.312.852.0120
E-mail Address:	jbock@accuvant.com
Project Involvement:	Project oversight and quality assurance.

Name:	<b>Phil Brass</b>
Title:	Practice Manager –Software Security Group – Accuvant LABS
Phone:	Mobile: +1.678.296.5568
E-mail Address:	pbrass@accuvant.com
Project Involvement:	Project oversight and quality assurance.

Name:	<b>Elizabeth Kinneel</b>
Title:	Account Manager
E-mail Address:	ekinneel@accuvant.com
Project Involvement:	Sales contact

## Appendix B – Methodology Overview

---

This section provides a brief overview of the methodologies Accuvant LABS uses for each phase of the assessment.

### Application Profiling

Application profiling is the process of reviewing the available documentation and performing active analysis to determine how the application functions. User facing and developer documentation is reviewed, along with mapping the major application use cases. The application technology stack information is obtained either via communication with the development team or via technical analysis from interaction with the application. The user group and role matrix is considered, along with other critical aspects of the security model. Finally, key properties of the application's security controls are documented.

### Threat Analysis

Accuvant LABS works with the application owners to identify critical data and functionality in the application that could be a target for attackers. This data is identified along with notes regarding its use and location within the application environment. The assessment team documents threat scenarios based on information from the development team as well as their own experience with application security.

### Dynamic Testing

Dynamic testing procedures assess a running instance of the application for vulnerabilities. This is primarily a manual effort, with some support from assessment tools. The manual tests included review of the following:

- Direct interaction with the application (anonymous)
- Direct interaction with the application (privileged access with credentials)
- Review of system configurations
- Third party API interfaces and services

Further testing components may include:

- Application Environment Testing – Network vulnerability assessment of server and network platforms identify any issues in the application's supporting infrastructure.
- Unauthenticated Testing – Conduct tests with application assessment tools without authenticating to simulate an unauthenticated attacker. This includes a manual walkthrough of the application target with the assessment tool to configure the test run.
- Authenticated Testing – Conduct tests with user accounts included in the scope of the assessment. These include manual walkthroughs of the application target with the assessment tool to configure the test runs.

Below is a listing of some of the high-level categories that may be included in application testing. This is not a comprehensive list of all the areas to be tested in the assessment.

- Abuse of Functionality
- Application Automation
- Audit and Logging Controls
- Authentication Deficiencies
- Authorization Deficiencies
- Command Execution
- Data Privacy and Integrity Controls
- Data Validation and Sanitization
- Encryption Implementations and Key Management
- Endpoint and Client Security Controls
- File and I/O Handling
- Injection Based Flaws
- Race Conditions
- Secure Channel Enforcement

- Service Configuration and Security
- Session Management
- UI and Content Security

The last step in this phase is to validate and potentially exploit the vulnerabilities identified through a proof of concept. This activity helps demonstrate the impact a particular issue can have on the environment, allows for further system penetration, and aids in identifying false positives.



### About Accuvant

Accuvant is the only research-driven information security partner delivering alignment between IT security and business objectives, clarity to complex security challenges, and confidence in complex security decisions.

Based on our clients' unique requirements, Accuvant assesses, architects and implements the policies, procedures and technologies that most efficiently and effectively protect valuable data assets.

Since 2002, more than 4,500 organizations, including half of the Fortune 100 and 800 federal, state and local entities, have trusted Accuvant with their security challenges. Headquartered in Denver, Accuvant has offices across the United States and Canada. For more information, please visit [www.accuvant.com](http://www.accuvant.com), follow us on Twitter: @Accuvant, or keep in touch via Facebook: <http://tiny.cc/facebook553>.