Assignment -4

OWASP Top10 vulnerabilities overview

The Open Web Application Security Project (OWASP) is a non-profit organization with a mission of improving the security of web applications. OWASP pursues this mission by providing developers with free access to a wide variety of security resources, including vulnerability listings, security best practices, deliberately vulnerable systems for practicing web application testing, and more. OWASP also has supported the development of application security testing tools and hosts multiple annual conferences around the world.

- What is the OWASP Top 10?:
- OWASP has developed a number of resources that describe the most common vulnerabilities that exist in various systems, including web applications, APIs, mobile devices, and more. The most famous of these is the OWASP Top Ten, which describes the ten most common and impactful vulnerabilities that appear in production web applications. This list is updated every few years based on a combination of security testing data and surveys of professionals within the industry.
- The most recent version of the OWASP Top 10 list was released in 2021. This resource provides information on the most common vulnerabilities, examples of each type, best practices for preventing them, and descriptions of how the vulnerability can be exploited. Additionally, each vulnerability includes references to related Common Weakness Enumeration (CWE) specifications, which describe a particular instance of a vulnerability. For example, the use of hard-coded passwords (CWE-259) falls under the Identification and Authentication Failures vulnerability within the OWASP Top Ten List.

OWASP Top Vulnerabilities

The latest version of the OWASP Top Ten contained several changes from the previous version. The 2021 list includes the following vulnerabilities:

1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8. Software and Data Integrity Failures
9. Security Logging and Monitoring Failures
10. Server-Side Request Forgery

Of these, four vulnerabilities (4, 8, and 10) are brand new, four are unchanged other than ranking, and the remainder consolidates or rename categories from the previous version of the list.

1. Broken Access Control:

- Access control systems are intended to ensure that only legitimate users have access to data or functionality. Vulnerabilities in the broken access control category include any issue that allows an attacker to bypass access controls or that fails to implement the principle of least privilege. For example, a web application might allow a user to access another user’s account by modifying the provided URL.

2. Cryptographic Failures:

- Cryptographic algorithms are invaluable for protecting data privacy and security; however, these algorithms can be very sensitive to implementation or configuration errors. Cryptographic failures include a failure to use encryption at all, misconfigurations of cryptographic algorithms, and insecure key management. For example, an organization might use an insecure hash algorithm for password storage, fail to salt passwords, or use the same salt for all stored user passwords.

- <u>3. Injection:</u>
- Injection vulnerabilities are made possible by a failure to properly sanitize user input before processing it. This can be especially problematic in languages such as SQL where data and commands are intermingled so that maliciously malformed user-provided data may be interpreted as part of a command. For example, SQL commonly uses single (‘) or double (“) quotation marks to delineate user data within a query, so user input containing these characters might be capable of changing the command being processed.

4. Insecure Design:

- Vulnerabilities can be introduced into software during the development process in a couple of different ways. While many of the vulnerabilities on the OWASP Top Ten list deal with implementation errors, this vulnerability describes failures in design that undermine the security of the system. For example, if the design for an application that stores and processes sensitive data does not include an authentication system, then a perfect implementation of the software as designed will still be insecure and fail to properly protect this sensitive data.

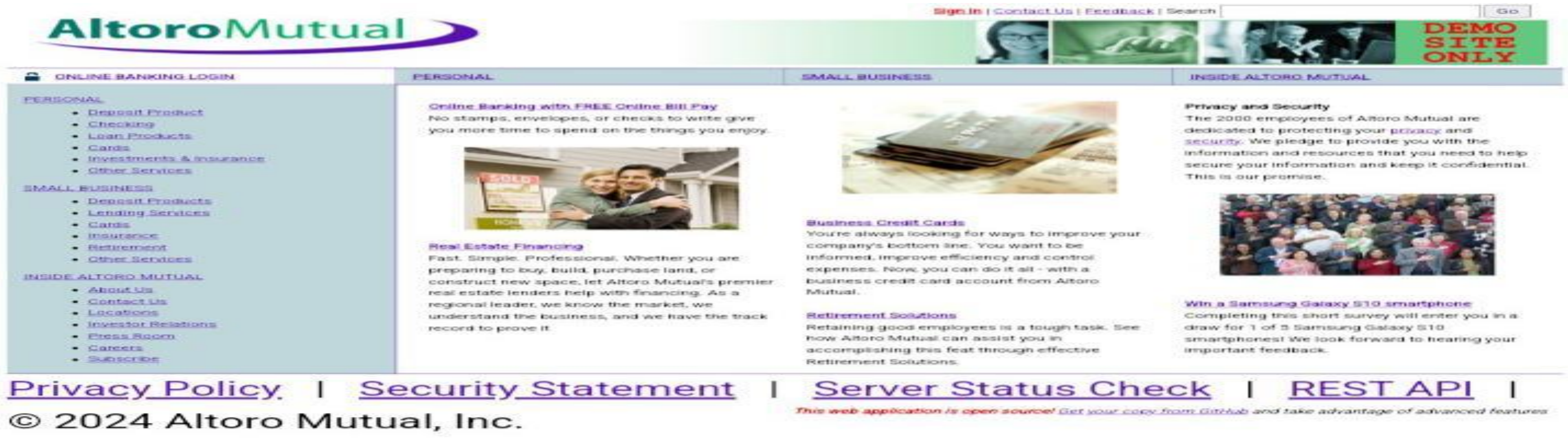
- 5. Security Misconfiguration:
- In addition to its design and implementation, the security of an application is also determined by how it is configured. A software manufacturer will have default configurations for their applications, and the users may also enable or disable various settings, which can improve or impair the security of the system. Examples of security misconfigurations could include enabling unnecessary applications or ports, leaving default accounts and passwords active and unchanged, or configuring error messages to expose too much information to a user.
- 6 Vulnerable and Outdated Components:
- Supply chain vulnerabilities have emerged as a major concern in recent years, especially as threat actors have attempted to insert malicious or vulnerable code into commonly used libraries and third-party dependencies. If an organization lacks visibility into the external code that is used within its applications — including nested dependencies — and fails to scan it for dependencies, then it may be vulnerable to exploitation. Also, a failure to promptly apply security updates to these dependencies could leave exploitable vulnerabilities open to attack. For example, an application may import a third-party library that has its own dependencies that could contain known exploitable vulnerabilities.

- <u>7. Identification and Authentication Failures:</u>
- Many applications and systems require some form of identification and authentication, such as a user proving their identity to an application or a server providing a digital certificate verifying its identity to a user when setting up a TLS-encrypted connection. Identification and authentication failures occur when an application relies upon weak authentication processes or fails to properly validate authentication information. For example, an application that lacks multi-factor authentication (MFA) might be vulnerable to a credential stuffing attack in which an attacker automatically tries username and password combinations from a list of weak, common, default, or compromised credentials.
- <u>8.Software and Data Integrity Failure:</u>
- The Software and Data Integrity Failures vulnerability in the OWASP Top 10 list addresses weaknesses in the security of an organization’s DevOps pipeline and software update processes similar to those that made the solar winds hack possible. This vulnerability class includes relying on third-party code from untrusted sources or repositories, failing to secure access to the CI/CD pipeline, and not properly validating the integrity of automatically applied updates. For example, if an attacker can replace a trusted module or dependency with a modified or malicious version, then applications that are built with that dependency could run malicious code or be vulnerable to exploitation.

- 9. Security Logging and Monitoring Failures:
- Security Logging and Monitoring Failures is the first of the vulnerabilities that are derived from survey responses and has moved up from the tenth spot in the previous iteration of the list. Many security incidents are enabled or exacerbated by the fact that an application fails to log significant security events or that these log files are not properly monitored and handled. For example, an application may not generate log files, may generate security logs that lack critical information, or these log files may only be available locally on a computer, making them only useful for investigation after an incident has been detected. All of these failures degrade an organization's ability to rapidly detect a potential security incident and to respond in real-time.
- 10. Server-Side Request Forgery
- Server-side request forgery (SSRF) is unusual among the vulnerabilities listed in the OWASP Top Ten list because it describes a very specific vulnerability or attack rather than a general category. SSRF vulnerabilities are relatively rare; however, they have a significant impact if they are identified and exploited by an attacker. The Capital One hack is an example of a recent, high-impact security incident that took advantage of an SSRF vulnerability.

Altoro mutual website analysis

- Altoro Mutual is a subsidiary of Altoro, a multi-state holding company located in the heart of Massachusetts. Altoro Mutual has been serving Boston and surrounding communities for nearly 75 years.
- Altoro Mutual offers a broad range of commercial, private, retail and mortgage banking services to small- and middle-market businesses and individuals. We pride ourselves on constantly surpassing the demands of our most loyal customers. And, we are determined to help you stay ahead of your expectations.



Sign In | Contact Us | Feedback | Search Go

ONLINE BANKING LOGIN | PERSONAL | SMALL BUSINESS | INSIDE ALTORO MUTUAL

PERSONAL
- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

SMALL BUSINESS
- Deposit Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

INSIDE ALTORO MUTUAL
- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers
- Subscribe

Online Banking with FREE Online Bill Pay
No stamps, envelopes, or checks to write give you more time to spend on the things you enjoy.

Real Estate Financing
Fast. Simple. Professional. Whether you are preparing to buy, build, purchase land, or construct new space, let Altoro Mutual's premier real estate lenders help with financing. As a regional leader, we know the market, we understand the business, and we have the track record to prove it.

Business Credit Cards
You're always looking for ways to improve your company's bottom line. You want to be informed, improve efficiency and control expenses. Now, you can do it all - with a business credit card account from Altoro Mutual.

Retirement Solutions
Retaining good employees is a tough task. See how Altoro Mutual can assist you in accomplishing this feat through effective Retirement Solutions.

Privacy and Security
The 2000 employees of Altoro Mutual are dedicated to protecting your privacy and security. We pledge to provide you with the information and resources that you need to help secure your information and keep it confidential. This is our promise.

Win a Samsung Galaxy S10 smartphone
Completing this short survey will enter you in a draw for 1 of 5 Samsung Galaxy S10 smartphones! We look forward to hearing your important feedback.

Privacy Policy | Security Statement | Server Status Check | REST API |

© 2024 Altoro Mutual, Inc.

This web application is open source! Get your copy from GitHub and take advantage of advanced features

H Authentication Bypass Using SQL Injection 1 TOC

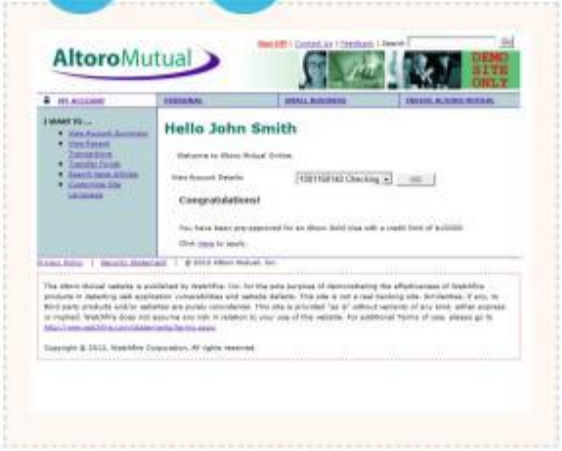
Issue 1 of 1 TOC

Authentication Bypass Using SQL Injection

Severity:	High
URL:	http://demo.testfire.net/bank/login.aspx
Entity:	uid (Parameter)
Risk:	It may be possible to bypass the web application's authentication mechanism
Causes:	Sanitation of hazardous characters was not performed correctly on user input
Fix:	Review possible solutions for hazardous character injection

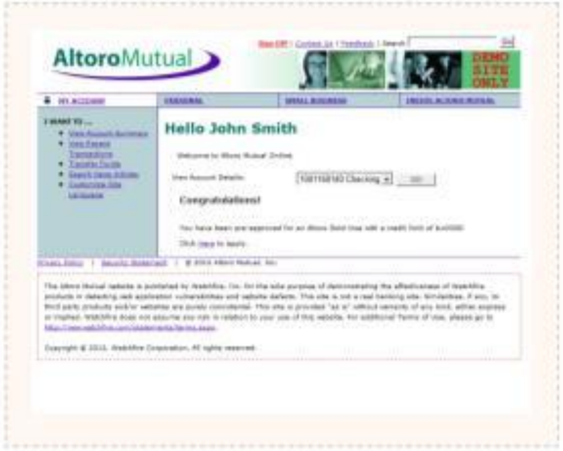
Reasoning: The test result seems to indicate a vulnerability because when four types of request were sent - a valid login, an invalid login, an SQL attack, and another invalid login - the responses to the two invalid logins were the same, while the response to the SQL attack seems similar the response to the valid login.

Valid Login



≈

Test Login



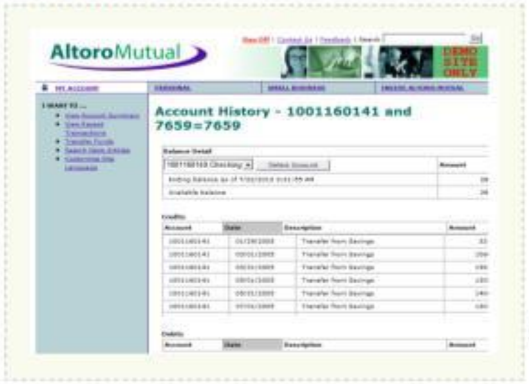
H Blind SQL Injection 1 TOC

Blind SQL Injection

Severity:	High
URL:	http://demo.testfire.net/bank/account.aspx
Entity:	listAccounts (Parameter)
Risk:	It is possible to view, modify or delete database entries and tables
Causes:	Sanitation of hazardous characters was not performed correctly on user input
Fix:	Review possible solutions for hazardous character injection

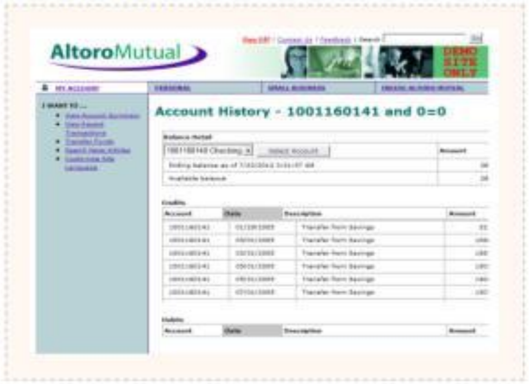
Reasoning: The test result seems to indicate a vulnerability because it shows that values can be appended to parameter values, indicating that they were embedded in an SQL query.HEX(0D)HEX(0A)In this test, three (or sometimes four) requests are sent. The last is logically equal to the original, and the next-to-last is different. Any others are for control purposes. A comparison of the last two responses with the first (the last is similar to it, and the next-to-last is different) indicates that the application is vulnerable.

Original Response

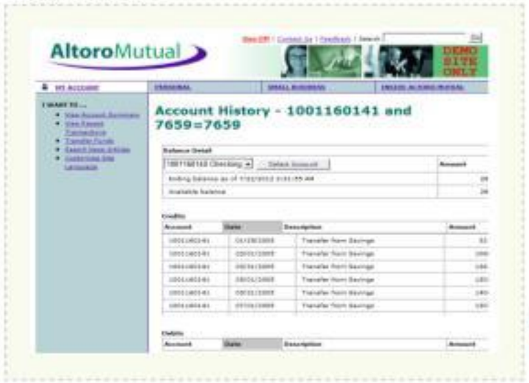


≈

Test Response (last)

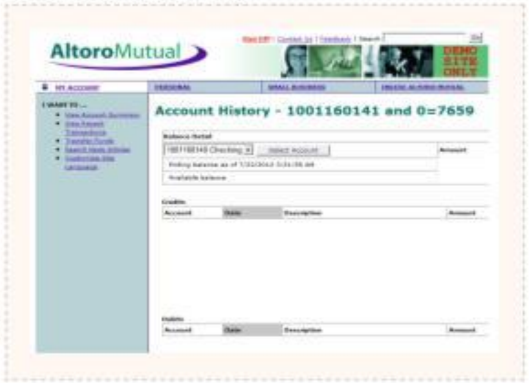


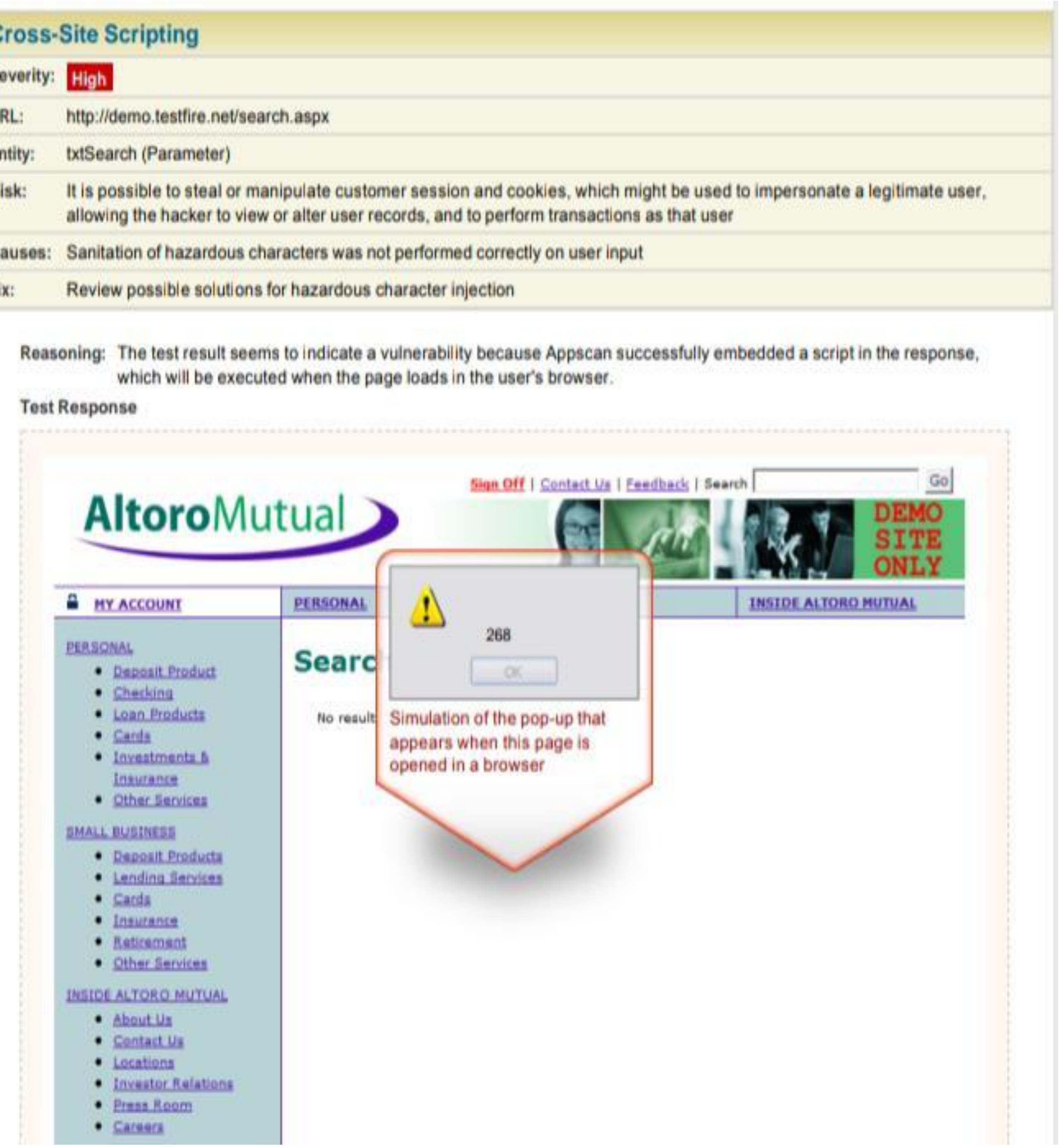
Original Response



≠

Test Response (next-to-last)

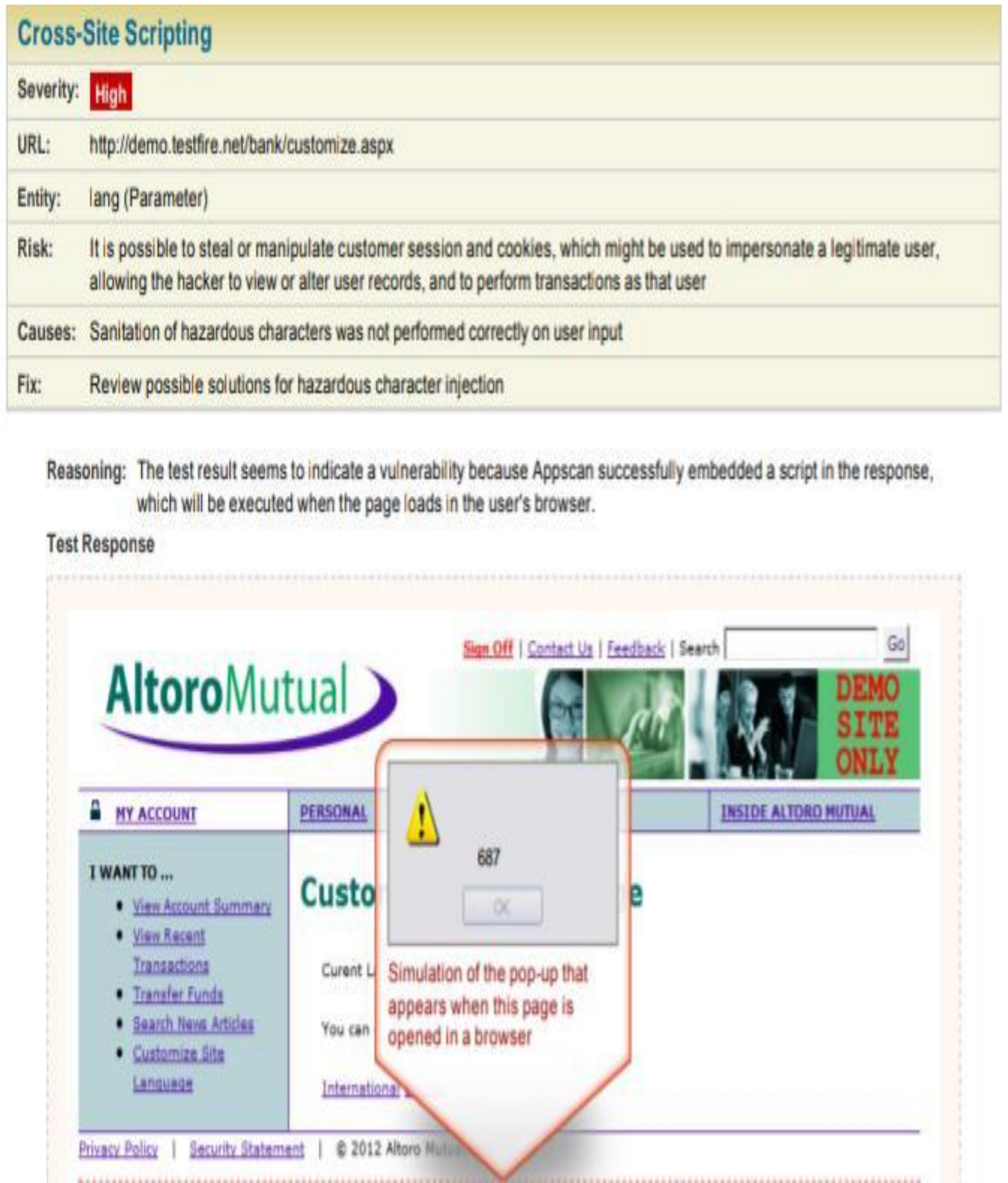



ross-Site Scripting

everity:	High
RL:	http://demo.testfire.net/search.aspx
ntity:	txtSearch (Parameter)
isk:	It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user
auses:	Sanitation of hazardous characters was not performed correctly on user input
ix:	Review possible solutions for hazardous character injection

Reasoning: The test result seems to indicate a vulnerability because Appscan successfully embedded a script in the response, which will be executed when the page loads in the user's browser.

Test Response


Cross-Site Scripting

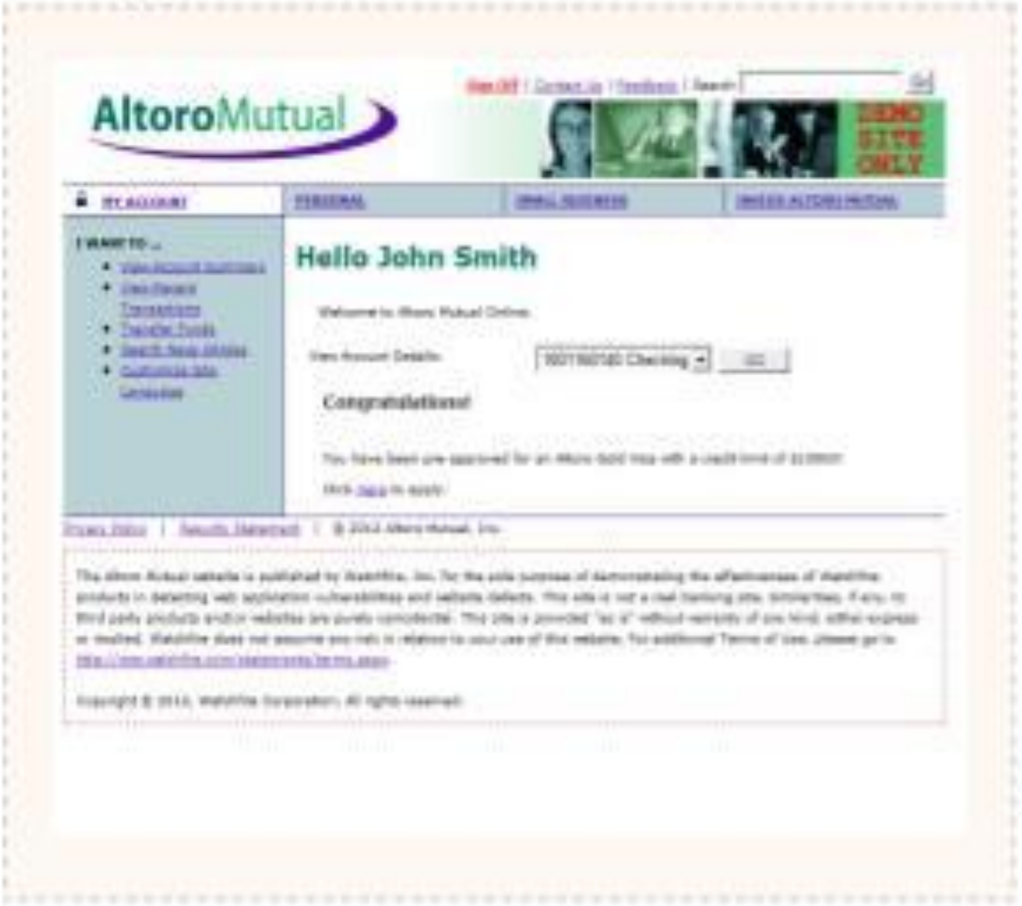
Severity:	High
URL:	http://demo.testfire.net/bank/customize.aspx
Entity:	lang (Parameter)
Risk:	It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user
Causes:	Sanitation of hazardous characters was not performed correctly on user input
Fix:	Review possible solutions for hazardous character injection

Reasoning: The test result seems to indicate a vulnerability because Appscan successfully embedded a script in the response, which will be executed when the page loads in the user's browser.

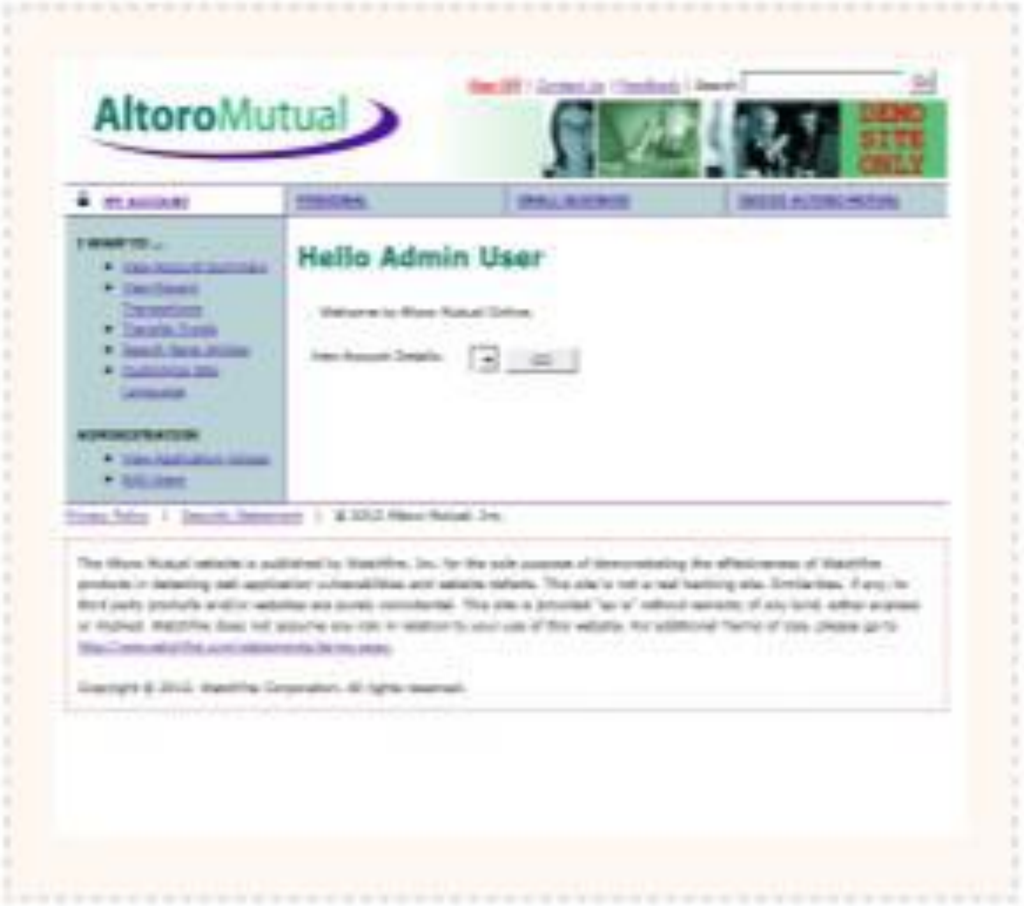
Test Response

Reasoning: This test consists of four requests: valid login, invalid login, login with predictable credentials, and another invalid login. If the response to the predictable credentials looks like the valid login (and different to the invalid logins), AppScan establishes that the application is vulnerable to this issue.

Valid Login



Test Login



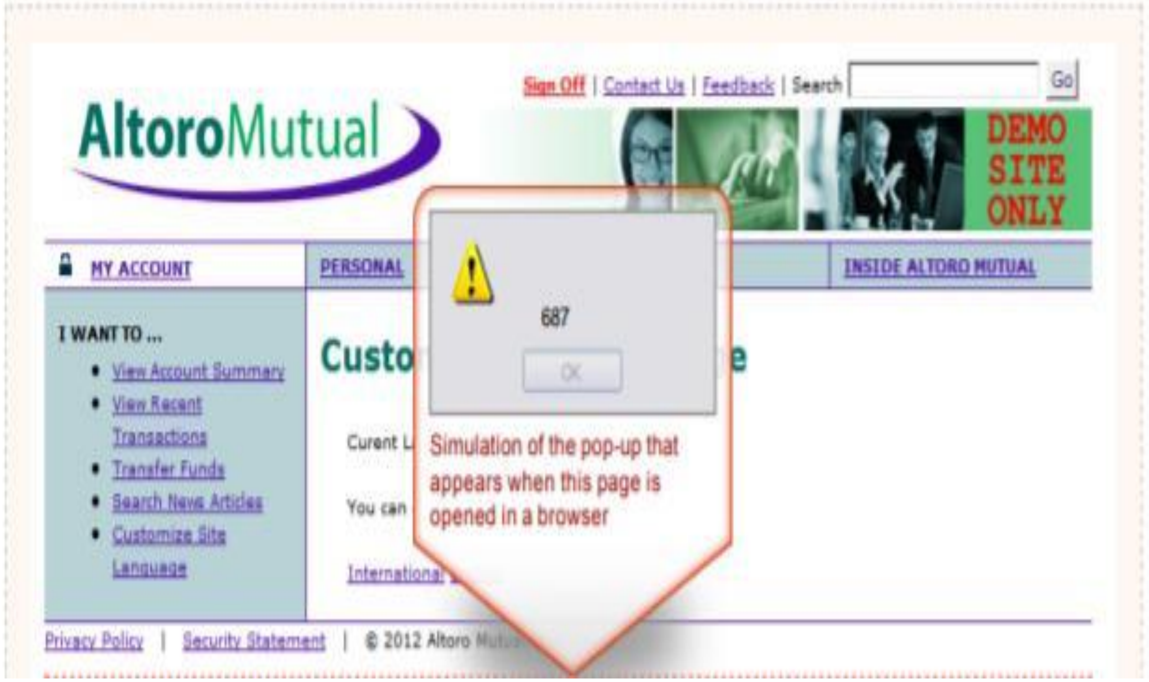
≈

Cross-Site Scripting

Severity:	High
URL:	http://demo.testfire.net/bank/customize.aspx
Entity:	lang (Parameter)
Risk:	It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user
Causes:	Sanitation of hazardous characters was not performed correctly on user input
Fix:	Review possible solutions for hazardous character injection

Reasoning: The test result seems to indicate a vulnerability because Appscan successfully embedded a script in the response, which will be executed when the page loads in the user's browser.

Test Response

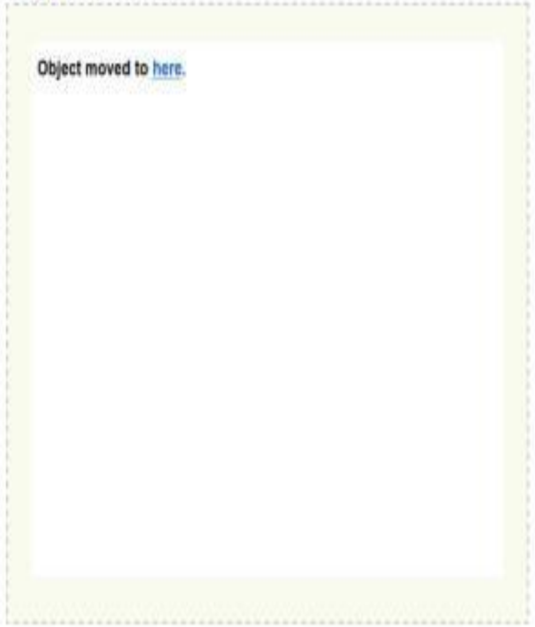


Cross-Site Request Forgery

Severity:	Medium
URL:	http://demo.testfire.net/bank/login.aspx
Entity:	login.aspx (Page)
Risk:	It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user
Causes:	Insufficient authentication method was used by the application
Fix:	Decline malicious requests

Reasoning: The test result seems to indicate a vulnerability because the Test Response (on the right) is identical to the Original Response (on the left), indicating that the login attempt was successful, even though it included hazardous characters.

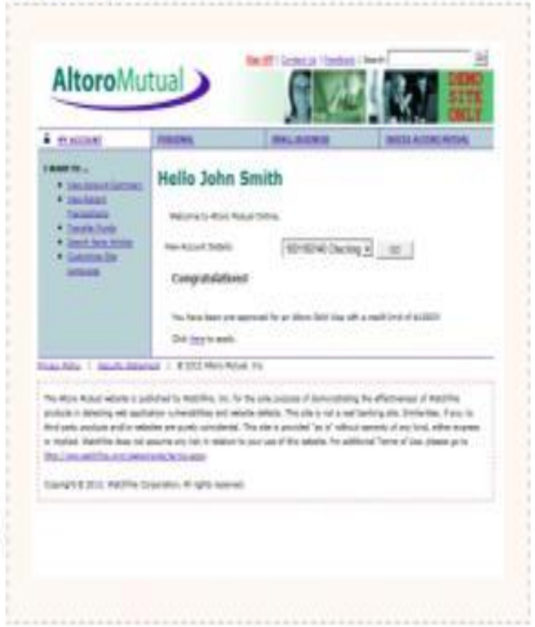
Original Response



Object moved to here.

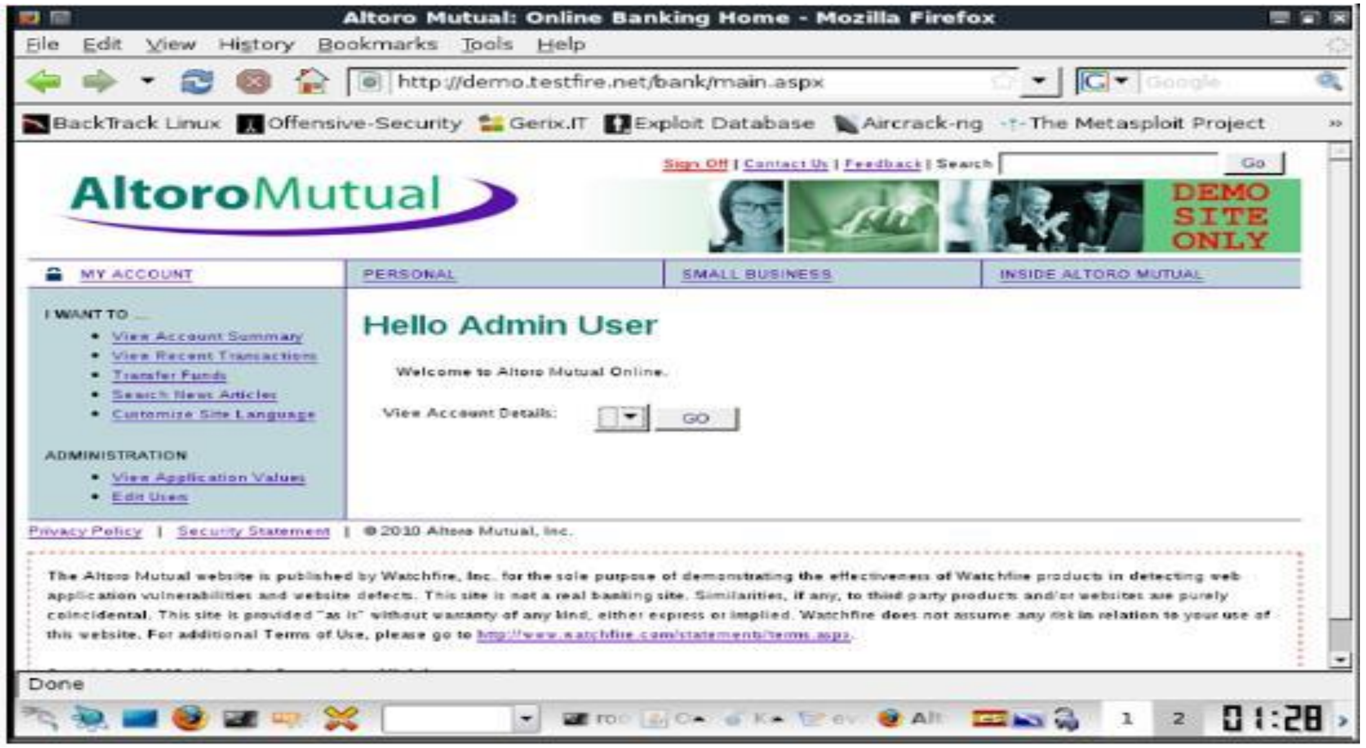
≈

Test Response



- It is very easy to circumvent login page according to the previous behavior exposed by the web page, we just have to use the following:
- · User: “admin’—“ (exclude the double quotes).
- · Password: whatever as its going to be ignored because of the “—“ symbols that are meant to comment lines in SQL.
- We see we have logged

With admin account:



Hello Admin User

Welcome to Altoro Mutual Online.

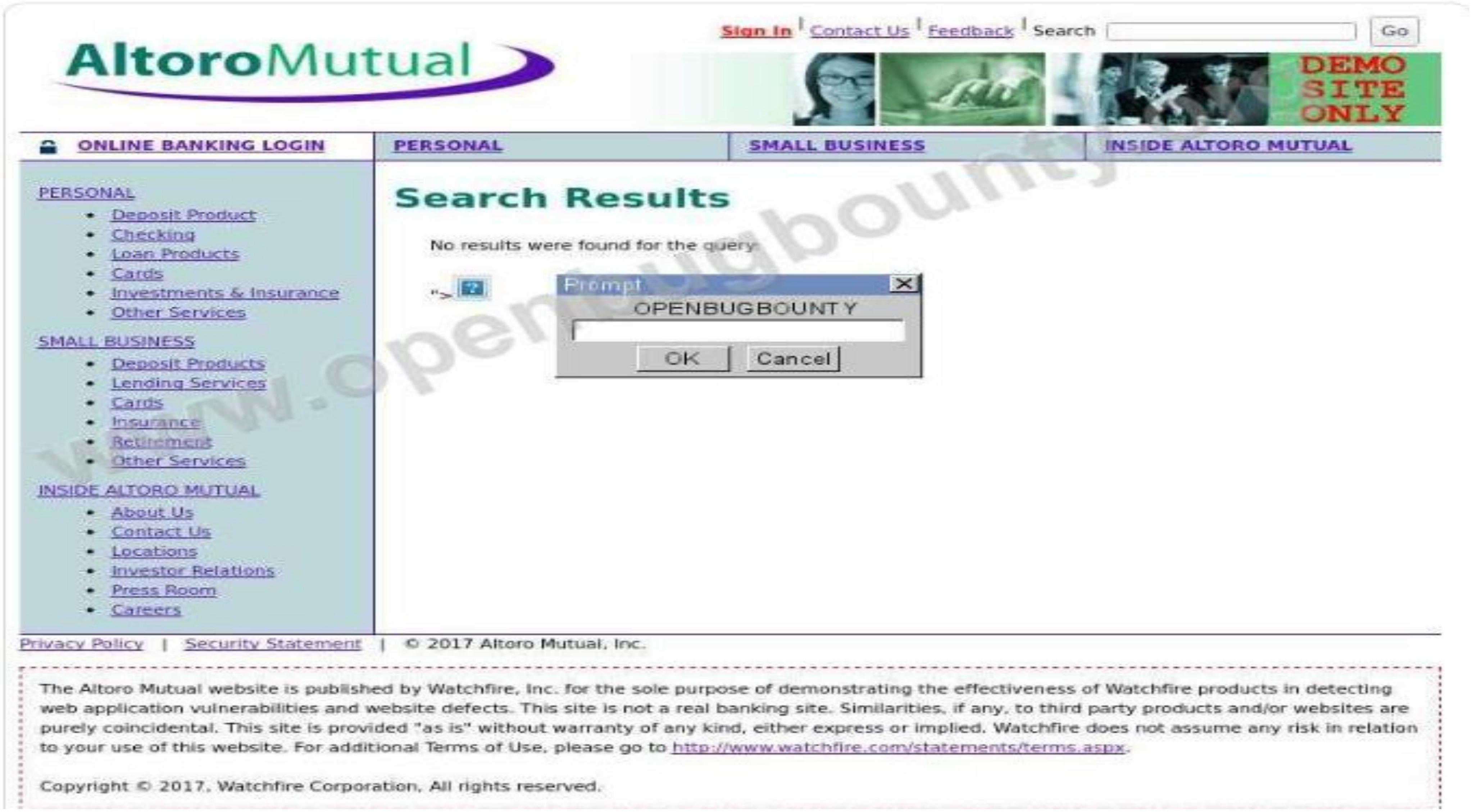
© 2010 Altoro Mutual, Inc.

The Altoro Mutual website is published by Watchfire, Inc. for the sole purpose of demonstrating the effectiveness of Watchfire products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. Watchfire does not assume any risk in relation to your use of this website. For additional Terms of Use, please go to http://www.watchfire.com/statements/terms.aspx.

Vulnerability identification report

- Altoromutual.com Cross Site Scripting Vulnerability
- Report ID: OBB-284871
- Security Researcher Y4r4G_, a holder of 3 badges for responsible and coordinated disclosure, found Cross Site Scripting security vulnerability affecting altoromutual.com website and its users.
- Following the coordinated and responsible vulnerability disclosure guidelines of the ISO 29147 standard, Open Bug Bounty has:
- a. verified the vulnerability and confirmed its existence
- b. notified the website operator about its existence.

- Affected Website: altoromutual.com
- Open Bug Bounty Program: Create your bounty program now. It’s open and free.
- Vulnerable Application: Custom Code
- Vulnerability Type: XSS (Cross Site Scripting) / CWE-79
- CVSSv3 Score: 6.1 [CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N]
- Discovered and Reported by: Y4r4G_
- Remediation Guide: OWASP XSS Prevention Cheat Sheet



PERSONAL
- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

SMALL BUSINESS
- Deposit Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

INSIDE ALTORO MUTUAL
- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers

Search Results

No results were found for the query:

">

Prompt

OPENBUGBOUNTY

OK Cancel

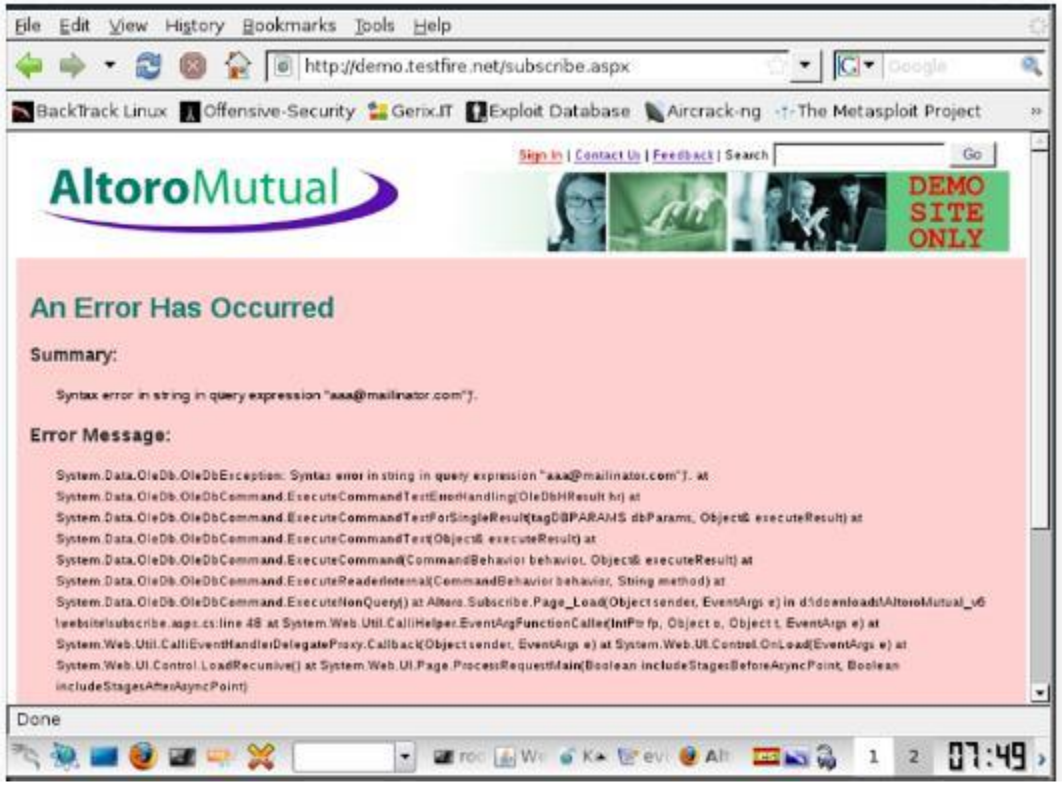
Privacy Policy | Security Statement | © 2017 Altoro Mutual, Inc.

The Altoro Mutual website is published by Watchfire, Inc. for the sole purpose of demonstrating the effectiveness of Watchfire products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. Watchfire does not assume any risk in relation to your use of this website. For additional Terms of Use, please go to http://www.watchfire.com/statements/terms.aspx.

Copyright © 2017, Watchfire Corporation, All rights reserved.

- Content security policy (CSP) is a browser mechanism that aims to mitigate the impact of cross-site scripting and some other vulnerabilities. If an application that employs CSP contains XSS-like behavior, then the CSP might hinder or prevent exploitation of the vulnerability.
- Validation, encoding and sanitization are primary XSS prevention techniques, but others can help limit damage from inadvertent mistakes. These include cookie attributes, which change how JavaScript and browsers can interact with cookies, and a content security policy allow list that prevents content from being loaded.

- We have a nice Database error that could derive into a SQL Injection attack.
- It seems to be an underlying insert clause but does not respond as expected to Boolean clauses.



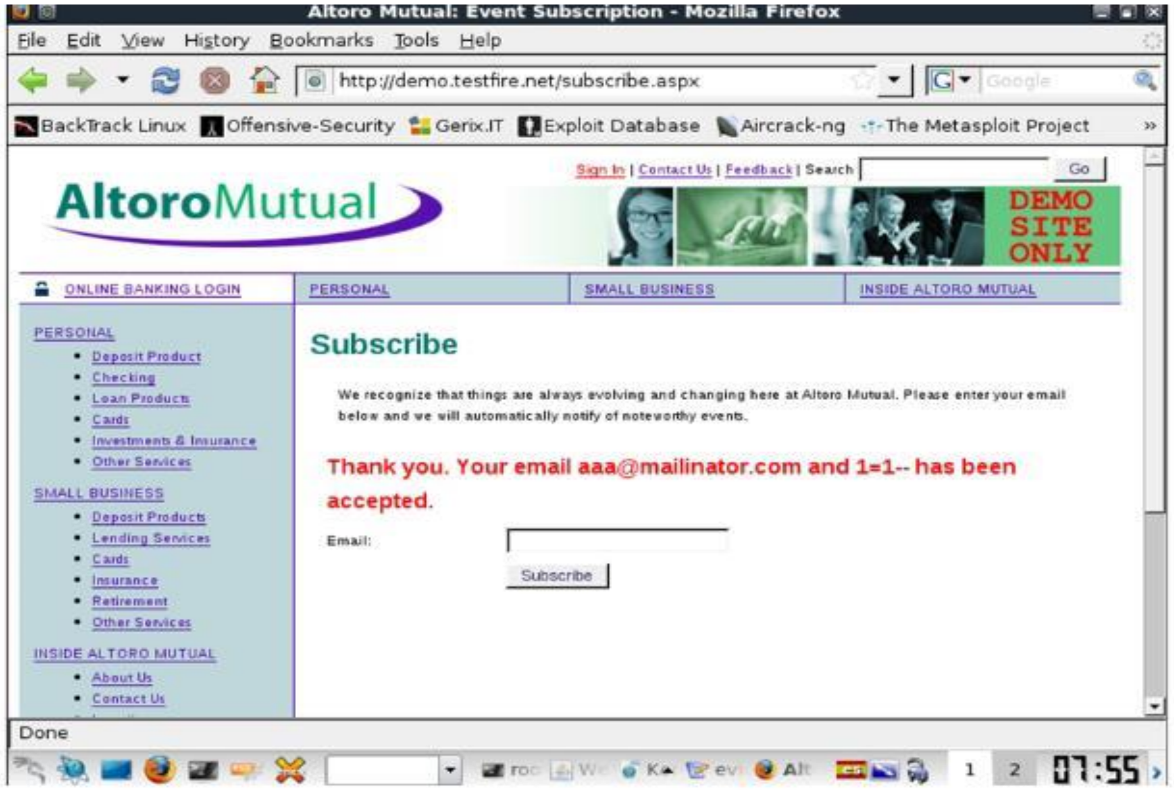
An Error Has Occurred

Summary:

Syntax error in string in query expression "aaa@mailinator.com'')'.

Error Message:

```
System.Data.OleDb.OleDbException: Syntax error in string in query expression "aaa@mailinator.com'')'. at
System.Data.OleDb.OleDbCommand.ExecuteCommandTextErrorHandling(OleDbHResult hr) at
System.Data.OleDb.OleDbCommand.ExecuteCommandTextForSingleResult(tagDBPARAMS dbParams, Object& executeResult) at
System.Data.OleDb.OleDbCommand.ExecuteCommandText(Object& executeResult) at
System.Data.OleDb.OleDbCommand.ExecuteCommand(CommandBehavior behavior, Object& executeResult) at
System.Data.OleDb.OleDbCommand.ExecuteReaderInternal(CommandBehavior behavior, String method) at
System.Data.OleDb.OleDbCommand.ExecuteNonQuery() at Altoro.Subscribe.Page_Load(Object sender, EventArgs e) in d:\downloads\AltoroMutual_v6
\website\subscribe.aspx.cs:line 48 at System.Web.Util.CalliHelper.EventArgFunctionCaller(IntPtr fp, Object o, Object t, EventArgs e) at
System.Web.Util.CalliEventHandlerDelegateProxy.Callback(Object sender, EventArgs e) at System.Web.UI.Control.OnLoad(EventArgs e) at
System.Web.UI.Control.LoadRecursive() at System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean
includeStagesAfterAsyncPoint)
```



Subscribe

We recognize that things are always evolving and changing here at Altoro Mutual. Please enter your email below and we will automatically notify of noteworthy events.

Thank you. Your email aaa@mailinator.com and 1=1-- has been accepted.

Email:

Mitigation strategy proposal

- High risks of SQL injection:
- The impact SQL injection can have on a business is far-reaching. A successful attack may result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.
- SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

- <u>SQL prevention and mitigation:</u>
- There are several effective ways to prevent SQLI attacks from taking place, as well as protecting against them, should they occur.
- The first step is input validation (a.k.a. sanitization), which is the practice of writing code that can identify illegitimate user inputs.
- While input validation should always be considered best practice, it is rarely a foolproof solution. The reality is that, in most cases, it is simply not feasible to map out all legal and illegal inputs—at least not without causing a large number of false positives, which interfere with user experience and an application’s functionality.

- For this reason, a web application firewall (WAF) is commonly employed to filter out SQLI, as well as other online threats. To do so, a WAF typically relies on a large, and constantly updated, list of meticulously crafted signatures that allow it to surgically weed out malicious SQL queries. Usually, such a list holds signatures to address specific attack vectors and is regularly patched to introduce blocking rules for newly discovered vulnerabilities.
- Modern web application firewalls are also often integrated with other security solutions. From these, a WAF can receive additional information that further augments its security capabilities.
- For example, a web application firewall that encounters a suspicious, but not outright malicious input may cross-verify it with IP data before deciding to block the request. It only blocks the input if the IP itself has a bad reputational history.

- <u>High risks of cross site scripting:</u>
- Cross-Site Scripting are one of the most dangerous applications that web applications and users face in the security supply chain. They result into loss of trust from the company when their web applications are attacked and users or visitors fall victim of such.
- XSS can pose various risks to the users and the web application owners, depending on the type and the severity of the attack. For instance, data theft can occur when an attacker uses XSS to steal cookies, session tokens, credentials, personal information, or payment details. An account takeover is another risk, where an attacker changes a user's password, email, or profile settings. Additionally, malware infection is possible when an attacker infects a user's device with ransomware, spyware, or keyloggers. Finally, website defacement is another risk where an attacker changes the layout, content, or images of a web application or redirects a user to another website. All of these risks could damage the web application's reputation, credibility, or revenue.

- <u>Preventing XSS:</u>
- To prevent XSS, the principle of least privilege should be applied, which requires that only the minimum amount of code and data necessary for functionality is allowed and anything else is rejected or filtered out. Input validation is one of the most effective practices to prevent XSS, as it validates user input and rejects or sanitizes any input that contains malicious or unexpected characters. Output encoding is also important, as it escapes any characters that have a special meaning in HTML and uses HTML entities or hexadecimal codes to represent them. Additionally, Content Security Policy (CSP) should be used to specify which sources of code and data are trusted and allowed by the browser, as well as nonce or hash attributes to verify the integrity of the code and data.