SaaS Security Scanner Report

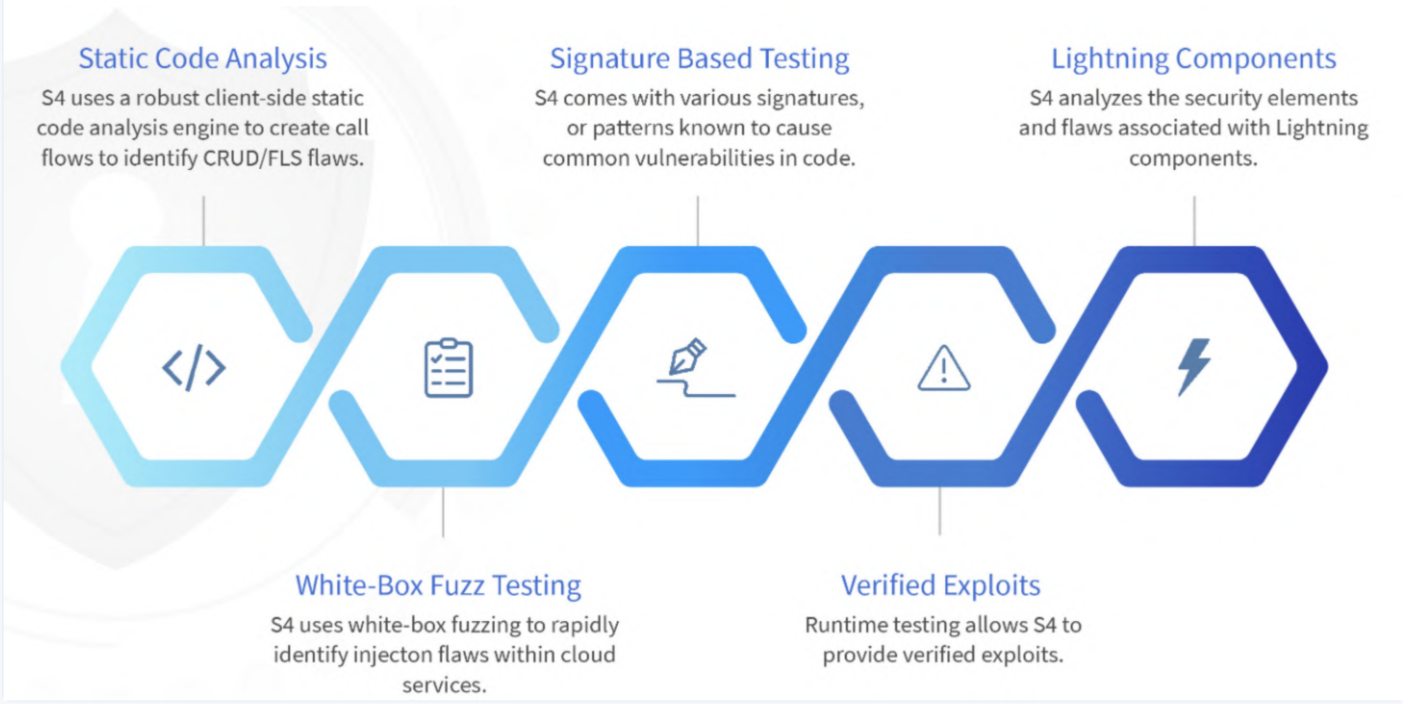
Thu Feb 04 2021 17:04:37 GMT-0800 (Pacific Standard Time)

Overview

A successful security scan was executed against your Salesforce environment. This document lists the vulnerabilities identified by the **SaaS Security Scanner** for Salesforce. It is recommended to fix all **high risk** vulnerabilities detailed in this report. Medium and low risk issues should be fixed as a best practice.

Process

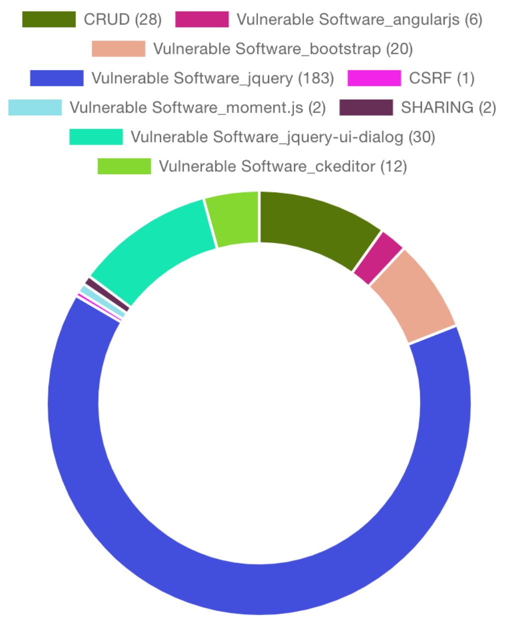
S4 — SaaS Security Scanner for Salesforce utilizes **static**, **dynamic**, along with **configuration** testing in order to rapidly identify vulnerabilities in a Salesforce environment. The following figure shows the key components of the scanning process.


Static Code Analysis
S4 uses a robust client-side static code analysis engine to create call flows to identify CRUD/FLS flaws.
Signature Based Testing
S4 comes with various signatures, or patterns known to cause common vulnerabilities in code.
Lightning Components
S4 analyzes the security elements and flaws associated with Lightning components.
White-Box Fuzz Testing
S4 uses white-box fuzzing to rapidly identify injecton flaws within cloud services.
Verified Exploits
Runtime testing allows S4 to provide verified exploits.

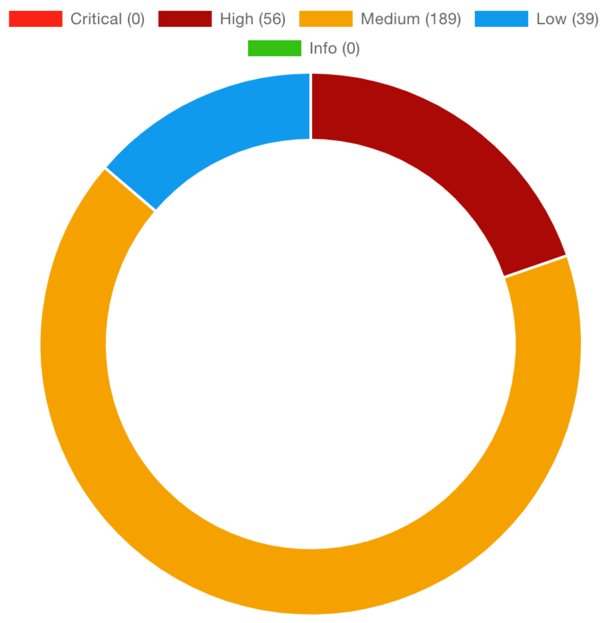
Findings Overview

Following is a visualization of vulnerabilities identified by S4.

Vulnerability Types



Vulnerability Severity Overview



HIGH

1 - Authorization Bypass in file classes/MyProfilePageController.cls on line: 15

ACTIVE | Wed Jan 06 2021 21:32:46 GMT-0800 (Pacific Standard Time) | Authorization Bypass

Background

Authorization, or access control, is a way of mediating access to resources and application functionality based on the identity of a user. Anytime a user can gain access to a resource that is denied to their role, they have performed authorization bypass. There are countless ways authentication bypass can occur. In this case, the user is able to execute a Read, Update, or Delete (CRUD) function in Apex code without their permission being validated correctly. Authorization Bypass occurs when user permissions are not validated prior to executing a Create, Read, Update, or Delete (CRUD) function in Apex code.

Issue (Code Snippet, Exploit)

Remediation Guidelines

- User permissions should always be validated before a CRUD function is executed in Apex code. This way, if the user does not have the correct permissions for what they are trying to do, they will now be allowed to proceed.
- Implement access control checks such as isAccessible() prior to Select. isAccessible() checks to see if the user who is trying to select something has permissions to access that information.
- Implement access control checks such as isCreateable() prior to insert. isCreateable() checks to see if the user who is trying to insert something is allowed to insert that type of information/object.
- Implement access control checks such as isUpdateable() prior to update. isUpdateable() checks to see if the user who is trying to update something is allowed to do so.
- Implement access control checks such as isDeleteable() prior to delete. isDeleteable() checks to see if the user who is trying to delete something is allowed to do so.
- Code Snippet: if(Schema.sObjectType.Case.isCreateable()) {insert SecurityScanCase;}

Vulnerability Trace

UserClass *MyProfilePageController* (*classes/MyProfilePageController.cls* ‣ Line: 5 Col: 27)

```
public with sharing class MyProfilePageController
```

↓

Method *MyProfilePageController* (*classes/MyProfilePageController.cls* ‣ Line: 14 Col: 12)

void MyProfilePageController()

↓

Assignment *user* (*classes/MyProfilePageController.cls* ‣ Line: 15 Col: 9)

user=[SELECT id, email, username, usertype, communitynickname, timezonesidkey, languagelocalekey, firstname, lastname, phone, title, street, city, country, postalcode, state, localesidkey, mobilephone, extension, fax, contact.email FROM User WHERE id = :UserInfo.getUserId()]

↓

Soql *[SELECT id, email, username, usertype, communitynickname, timezonesidkey, languagelocalekey, firstname, lastname, phone, title, street, city, country, postalcode, state, localesidkey, mobilephone, extension, fax, contact.email FROM User WHERE id = :UserInfo.getUserId()]* (*classes/MyProfilePageController.cls* ‣ Line: 15 Col: 16)

user=[SELECT id, email, username, usertype, communitynickname, timezonesidkey, languagelocalekey, firstname, lastname, phone, title, street, city, country, postalcode, state, localesidkey, mobilephone, extension, fax, contact.email FROM User WHERE id = :UserInfo.getUserId()]

HIGH

2 - Authorization Bypass in file classes/MyProfilePageController.cls on line: 48

ACTIVE | Wed Jan 06 2021 21:32:46 GMT-0800 (Pacific Standard Time) | Authorization Bypass

Background

Authorization, or access control, is a way of mediating access to resources and application functionality based on the identity of a user. Anytime a user can gain access to a resource that is denied to their role, they have performed authorization bypass. There are countless ways authentication bypass can occur. In this case, the user is able to execute a Read, Update, or Delete (CRUD) function in Apex code without their permission being validated correctly. Authorization Bypass occurs when user permissions are not validated prior to executing a Create, Read, Update, or Delete (CRUD) function in Apex code.

Issue (Code Snippet, Exploit)

Remediation Guidelines

- User permissions should always be validated before a CRUD function is executed in Apex code. This way, if the user does not have the correct permissions for what they are trying to do, they will now be allowed to proceed.
- Implement access control checks such as isAccessible() prior to Select. isAccessible() checks to see if the user who is trying to select something has permissions to access that information.
- Implement access control checks such as isCreateable() prior to insert. isCreateable() checks to see if the user who is trying to insert something is allowed to insert that type of information/object.
- Implement access control checks such as isUpdateable() prior to update. isUpdateable() checks to see if the user who is trying to update something is allowed to do so.
- Implement access control checks such as isDeleteable() prior to delete. isDeleteable() checks to see if the user who is trying to delete something is allowed to do so.
- Code Snippet: if(Schema.sObjectType.Case.isCreateable()) {insert SecurityScanCase;}

Vulnerability Trace

UserClass *MyProfilePageControllerTest* (*classes/MyProfilePageControllerTest.cls* ‣ Line: 5 Col: 35)

```
public with sharing class MyProfilePageControllerTest
```

↓

Method *testSave* (*classes/MyProfilePageControllerTest.cls* ‣ Line: 7 Col: 42)

void testSave()

↓

If-Else Block *condition taken* (*classes/MyProfilePageControllerTest.cls* ‣ Line: 11 Col: 9)

existingPortalUsers.isEmpty()

↓

MethodCall *cancel* (*classes/MyProfilePageControllerTest.cls* ‣ Line: 42 Col: 28)

controller.cancel()

↓

Method *cancel* (*classes/MyProfilePageController.cls* ‣ Line: 46 Col: 17)

void cancel()

↓

Assignment *user* (*classes/MyProfilePageController.cls* ‣ Line: 48 Col: 9)

user=[SELECT id, email, username, communitynickname, timezonesidkey, languagelocalekey, firstname, lastname, phone, title, street, city, country, postalcode, state, localesidkey, mobilephone, extension, fax, contact.email FROM User WHERE id = :UserInfo.getUserId()]

↓

Soql *[SELECT id, email, username, communitynickname, timezonesidkey, languagelocalekey, firstname, lastname, phone, title, street, city, country, postalcode, state, localesidkey, mobilephone, extension, fax, contact.email FROM User WHERE id = :UserInfo.getUserId()]* (*classes/MyProfilePageController.cls* ‣ Line: 48 Col: 16)

user=[SELECT id, email, username, communitynickname, timezonesidkey, languagelocalekey, firstname, lastname, phone, title, street, city, country, postalcode, state, localesidkey, mobilephone, extension, fax, contact.email FROM User WHERE id = :UserInfo.getUserId()]

UserClass *MyProfilePageControllerTest* (*classes/MyProfilePageControllerTest.cls* ‣ Line: 5 Col: 35)

public with sharing class MyProfilePageControllerTest

↓

Method *testSave* (*classes/MyProfilePageControllerTest.cls* ‣ Line: 7 Col: 42)

void testSave()

↓

If-Else Block *condition taken* (*classes/MyProfilePageControllerTest.cls* ‣ Line: 11 Col: 9)

existingPortalUsers.isEmpty()

↓

MethodCall *cancel* (*classes/MyProfilePageControllerTest.cls* ‣ Line: 19 Col: 24)

controller.cancel()

↓

Method *cancel* (*classes/MyProfilePageController.cls* ‣ Line: 46 Col: 17)

void cancel()

↓

Assignment *user* (*classes/MyProfilePageController.cls* ‣ Line: 48 Col: 9)

user=[SELECT id, email, username, communitynickname, timezonesidkey, languagelocalekey, firstname, lastname, phone, title, street, city, country, postalcode, state, localesidkey, mobilephone, extension, fax, contact.email FROM User WHERE id = :UserInfo.getUserId()]

↓

Soql *[SELECT id, email, username, communitynickname, timezonesidkey, languagelocalekey, firstname, lastname, phone, title, street, city, country, postalcode, state, localesidkey, mobilephone, extension, fax, contact.email FROM User WHERE id = :UserInfo.getUserId()]* (*classes/MyProfilePageController.cls* ‣ Line: 48 Col: 16)

user=[SELECT id, email, username, communitynickname, timezonesidkey, languagelocalekey, firstname, lastname, phone, title, street, city, country, postalcode, state, localesidkey, mobilephone, extension, fax, contact.email FROM User WHERE id = :UserInfo.getUserId()]

HIGH

3 - Authorization Bypass in file classes/PasswordManipulation.cls on line: 14

ACTIVE | Wed Jan 06 2021 21:32:46 GMT-0800 (Pacific Standard Time) | Authorization Bypass

Background

Authorization, or access control, is a way of mediating access to resources and application functionality based on the identity of a user. Anytime a user can gain access to a resource that is denied to their role, they have performed authorization bypass. There are countless ways authentication bypass can occur. In this case, the user is able to execute a Read, Update, or Delete (CRUD) function in Apex code without their permission being validated correctly. Authorization Bypass occurs when user permissions are not validated prior to executing a Create, Read, Update, or Delete (CRUD) function in Apex code.

Issue (Code Snippet, Exploit)

Remediation Guidelines

- User permissions should always be validated before a CRUD function is executed in Apex code. This way, if the user does not have the correct permissions for what they are trying to do, they will now be allowed to proceed.
- Implement access control checks such as isAccessible() prior to Select. isAccessible() checks to see if the user who is trying to select something has permissions to access that information.
- Implement access control checks such as isCreateable() prior to insert. isCreateable() checks to see if the user who is trying to insert something is allowed to insert that type of information/object.
- Implement access control checks such as isUpdateable() prior to update. isUpdateable() checks to see if the user who is trying to update something is allowed to do so.
- Implement access control checks such as isDeleteable() prior to delete. isDeleteable() checks to see if the user who is trying to delete something is allowed to do so.
- Code Snippet: if(Schema.sObjectType.Case.isCreateable()) {insert SecurityScanCase;}

Vulnerability Trace

UserClass *PasswordManipulation* (*classes/PasswordManipulation.cls* ‣ Line: 1 Col: 27)

```
public with sharing class PasswordManipulation
```

↓

Method *PasswordManipulation* (*classes/PasswordManipulation.cls* ‣ Line: 2 Col: 12)

```
void PasswordManipulation()
```

↓

Assignment *u.profileId* (*classes/PasswordManipulation.cls* ‣ Line: 13 Col: 9)

```
u.profileId='00eG0000000FgyJ'
```

↓

DmlInsertStatement *u* (*classes/PasswordManipulation.cls* ‣ Line: 14 Col: 9)

```
insert u
```

Contact

DigitSec, Inc.
214 1st Ave S. Suite B03
Seattle, WA 98104
206-659-9521
info@digitsec.com

Confidential Information