SECURELY DEVELOP

Web Applications

When producing secure code for web applications, developers often use one of these five popular modern programming languages. Here's how these languages can be impacted by the Top 10 OWASP vulnerabilities and how to mitigate the risks:

NODE.JS is used by at least 20 million websites[1]

This coding language debuted in 2010 and is an open-source, cross-platform, runtime environment for server-side applications written in JavaScript. Node.js runs on the V8 engine and is commonly used for traditional websites and back-end API services. Applications built on top of the Node.js language may be vulnerable to application layer DDoS attacks, authentication bypass, and business logic attacks.[2]

OWASP TOP 10 EXAMPLE

INSECURE CODE

```
app.use(session({
// reportUri:
    path: /location,
    signed: true,
    program: 'string'
        secret:
'63ca39e0-2511-434e-8e80-f3530772a5de',
        key: 'cookieToken',
        cookie: { secure: false, httpOnly: true, path:
'/user', sameSite: true}
}));
```

SECURE CODE

```
app.use(session({
    path: /location,
    signed: true,
    program: 'string'
    secret: '63ca39e0-2511-434e-8e80-f3530772a5de',
    key: 'cookieToken',
    cookie: { secure: true, httpOnly: true, path:
'/user', sameSite: true}
}));
```

OWASP TOP 10 VULNERABILITY THE INSECURE CODE CREATES A RISK FOR:

A6: Security Misconfiguration
Failing to set the secure flag on a cookie for HTTPS connections can leave user's sessions open to hijacking attacks.

PYTHON is reported to be one of the best tools for data science, data analysis, and machine learning[3]

A powerful, flexible, and general-purpose coding language that has been around since the 1990s, Python has become one of the most popular programming languages today. The most up-to-date version is the 3.x branch. Python can be used for web development, back-end development, software development, data science, and automation/scripting.[4] Potential Python vulnerabilities include SQL injection, cross-site scripting (XSS), and command injection.[5]

OWASP TOP 10 EXAMPLE

INSECURE CODE

```
if sys.platform == "linux":
    # is root?
    if os.geteuid() != 0:
        print_error("You need to be root")
    else:
        outlog = run_command("ls -l '{showr}'")
elif sys.platform == "win32":
        outlog = run_command("dir show*")
```

SECURE CODE

```
if sys.platform == "linux":
    # is root?
    if os.geteuid() != 0:
        print_error("You need to be root")
    else:
        outlog = run_command("ls -l"
elif sys.platform == "win32":
    outlog = run_command("dir")
```

OWASP TOP 10 VULNERABILITY THE INSECURE CODE CREATES A RISK FOR:

A1: Injection
Using user-provider input in an unsafe manner can lead to injection vulnerabilities.

JAVA is utilized by 40.2% of software developers globally.[6]

A very popular programming language for enterprises, Java is backward compatible and runs on the Java Virtual Machine (JVM), making the case for platform independence. Java is rich in features and widely used for back-end development projects including big data and Android development. It can also be used for desktop computing, mobile computing, and games. Applications built with Java can be vulnerable to application layer attacks.[7]

OWASP TOP 10 EXAMPLE

INSECURE CODE

```
cipher.init(Algo.DES, ikey, vinit);
byte[] encryptedVal= new
byte[cipher.getOutputSize(input.length)];
int cipher_siz = cipher.update(input, 0,
input.length, encryptedVal, 0);
cipher_siz += cipher.doFinal(encryptedVal,
cipher_siz);
```

SECURE CODE

```
cipher.init(Algo.AES, ikey, vinit);
byte[] encryptedVal= new
byte[cipher.getOutputSize(input.length)];
int cipher_siz = cipher.update(input, 0, input.length,
encryptedVal, 0);
cipher_siz += cipher.doFinal(encryptedVal,
cipher_siz);
```

OWASP TOP 10 VULNERABILITY THE INSECURE CODE CREATES A RISK FOR:

A6: Security Misconfiguration
Using insecure encryption ciphers and protocols can result in the exposure of sensitive information.

C# is used by over 31% of software developers worldwide[8]

Microsoft debuted C# in 2000 as part of their .NET initiative[1]. C# is a multi-paradigm programming language that can run on Windows, iOS/Android, and Linux platforms. C# can be used in the creation of mobile or desktop apps, cloud-based services, websites, enterprise software, and games. The most prevalent vulnerabilities when using C# code are XSS, SQL Injection, Open Redirects, and Parameter Manipulation.[9]

OWASP TOP 10 EXAMPLE

INSECURE CODE

```
public string AccessSecret(string strComp, string strTok)
    {
        byte[] tok = { };
        string APIKey =
"c72004a566487962b8db247778672ab7c5b7134a97ea898
998fe6c3b20fd26bd1";
        byte[] strToArr;

        try
        {
            tok = Encoding.UTF8.GetBytes(strTok);
```

SECURE CODE

```
    public string AccessSecret(string
strComp, string strTok)
System.Environment.SetEnvironmentVa
riable(variable, value [, Target])

    {
        byte[] tok = { };
            byte[] strToArr;
        try
    {
        tok = Encoding.UTF8.GetBytes(strTok);
```

OWASP TOP 10 VULNERABILITY THE INSECURE CODE CREATES A RISK FOR:

A3: Sensitive Data Exposure
Hardcoding passwords, keys and other credentials is a poor coding practice that, when exposed, can lead to the compromise of an application, system, network, and more.

PHP powers a significant majority of the web [1]

A popular general-purpose programming language that is particularly suited to web development, PHP powers almost 80% of websites on the entire Internet today.[1,9] Vulnerabilities in PHP code are commonly caused by a mistake made while writing the original code that is then exploited by bots once the code is published.[10]

OWASP TOP 10 EXAMPLE

INSECURE CODE

```
if ($_POST["submit"]) {
$username = $_POST["username];
$sql = "SELECT COUNT(username) AS num FROM
account WHERE username = :username";
$stmt = $pdo->prepare($sql);
    $stmt->bindValue(':username, $username);
    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    if($row['num'] > 0){
    echo 'This user already exists, please choose a
new name.';
}
```

SECURE CODE

```
/ //gen random string for $username
..
if ($_POST["submit"]) {
$sql = "SELECT COUNT(username) AS num FROM
account WHERE username = :username";
$stmt = $pdo->prepare($sql);
    $stmt->bindValue(':username, $username);
    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    if($row['num'] > 0){
/ //regenerate random username string and
confirm its unique
}
```

OWASP TOP 10 VULNERABILITY THE INSECURE CODE CREATES A RISK FOR:

A2: Broken Authentication
Being overly verbose in providing messages to users is something attackers will focus on to take advantage of your application.

Each of these programming languages has its own individual "flavor" and uses, but they all have some commonalities as well. When learning a new programming language, it is important to look at the common vulnerabilities and best practices in secure code implementation to avoid security flaws. Use the OWASP Top 10 Web Application Security Risks list when developing with any of these coding languages.

Citations:

1. W3Techs Web Technology Surveys (March 1, 2021) Retrieved from https://w3techs.com/ on 4/1/21.
2. Ashbel, A. (June 9, 2015). "Node.js: Successful, exciting...and bares security risks." SANS Webcast. Retrieved 3/16/21.
3. Gugleta, Lazar (Feb. 18, 2020) "Why is Python so Powerful Today?" Medium.com. Retrieved 4/15/21.
4. Retrieved from python.org on 3/16/21.
5. "Digging for Security Bugs/Vulnerabilities in Python Applications" (July 22, 2018). Tripwire.com. Retrieved 3/16/21.
6. Liu, S. (June 11, 2020) "Most widely utilized programming languages among developers worldwide 2020" Retrieved from https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/ on 4/1/21.
7. Retrieved from SANS DEV541: Secure Coding in Java/JEE: Developing Defensible Applications brochure at https://www.sans.org/brochure/course/secure-coding-java-jee-developing-defensible-applications/306 on 3/16/21.
8. Retrieved from SANS DEV544: Secure Coding in .NET: Developing Defensible Applications brochure on 3/16/21 at https://www.sans.org/brochure/course/secure-coding-net-developing-defensible-applications/2252
9. Retrieved from php.net on 3/16/21.
10. "Understanding PHP Vulnerabilities & How They Originate". (January 4, 2019). Wordfence.com. Retrieved 3/16/21.

SANS SECURITY AWARENESS

sans.org/security-awareness-dev