Denial of Service (DoS attack) is a cyber attack that aims to temporarily or indefinitely disrupt the services of an internet connected host, making a machine or network resources inaccessible to actual users. It is usually performed in the form of making the target system unable to respond to incoming requests due to overload due to overloading the target machine or resource with unnecessary requests.The attackers can use amplification techniques for attacks with higher bandwidth than they have and the DNS Protocol statistically has been used/exploited much more than the other protocols.

a.Why is this type of attack preferred, especially in the DNS protocol?

Please explain briefly.

A Domain Name Server (DNS) Amplification attack is a popular form of Distributed Denial of Service (DDoS), in which attackers use publically accessible open DNS servers to flood a target system with DNS response traffic. The attacker sends a DNS name lookup request to an open DNS server with the source address spoofed to be the target’s address. When the DNS server sends the DNS record response, it is sent instead to the target. Attackers will typically submit a request for as much zone information as possible to maximize the amplification effect. The spoofed queries sent by the attacker are of the type, “ANY,” which returns all known information about a DNS zone in a single request. Because the size of the response is considerably larger than the request, the attacker is able to increase the amount of traffic directed at the victim. By leveraging a botnet to produce a large number of spoofed DNS queries, an attacker can create an immense amount of traffic with little effort. Additionally, because the responses are legitimate data coming from valid servers, it is extremely difficult to prevent these types of attacks. For these reasons, the DNS protocol is preferred for DDOS attack.

b.In DNS Amplification Attacks, are there any technical limitations or disadvantages for the attacker?

* **Source IP Verification:** Because the DNS queries being sent by the attacker-controlled clients must have a source address spoofed to appear as the victim’s system, the first step to reducing the effectiveness of DNS amplification is for Internet Service Providers to reject any DNS traffic with spoofed addresses.
* **Disabling Recursion on Authoritative Name Servers:** Many of the DNS servers currently deployed on the Internet are exclusively intended to provide name resolution for a single domain. In these systems, DNS resolution for private client systems may be provided by a separate server and the authoritative server acts only as a DNS source of zone information to external clients. These systems do not need to support recursive resolution of other domains on behalf of a client, and should be configured with recursion disabled.
* **Response Rate Limiting (RRL):** To provide the most effective protection, recommended that authoritative and recursive name servers run on different systems, with RRL implemented on the authoritative server and access control lists implemented on the recursive server. This will reduce the effectiveness of DNS amplification attacks by reducing the amount of traffic coming from any single authoritative server while not affecting the performance of the internal recursive resolvers.
* **Limiting Recursion to Authorized Clients:** For DNS servers that are deployed within an organization or Internet Service Provider, the resolver should be configured to perform recursive queries on behalf of authorized clients only. These requests typically should only come from clients within the organization’s network address range. Recommended that all server administrators restrict recursion to only clients on the organization’s network.

c.Please find a suitable IP address for amplification attacks in DNS protocol and explain step by step how to find such servers easily.

* [**64.32.16.8**](https://www.shodan.io/host/64.32.16.8)
* public-resolver.sharktech.net
* [Sharktech](https://www.shodan.io/search?query=open+dns+resolver+org%3A%22Sharktech%22)
* https://www.shodan.io/static/flags/16/US.png[United States](https://www.shodan.io/search?query=open+dns+resolver+country%3A%22US%22), [Los Angeles](https://www.shodan.io/search?query=open+dns+resolver+city%3A%22Los+Angeles%22)
* 9.16.1-Ubuntu
* Recursion: enabled
* **Resolver** name: **open**-**resolver**-**dns**
* First we go to Shodan Search Engine page.
* Second, search for "open dns resolver" in the search bar.
* Finally, by examining the IP addresses that appear, we find the IP address with "Recursion: enabled".

The source files of a sample application are attached as a .zip file. Please unzip the zip file in a directory. The code in the app.py file in the unzipped directory contains several OWASP Top 10:2021 vulnerabilities. Please review the code, identify contained vulnerabilities, and suggest possible mitigations to resolve the identified vulnerabilities. Each area has to include:

● What is the vulnerability?

● Why does the vulnerability arise (what is the reason for the vulnerability)?

● What is your mitigation suggestion to close the vulnerability?

Note 1: 10 areas are given below to write your answers. However, the application may or may not include 10 vulnerabilities.

Note 2: Applicants can run the sample application using docker with the “docker-compose up” command if necessary.

* 5 import hashlib,dbinit

6 from Crypto.Cipher import DES

7

8 app = Flask(\_\_name\_\_)

* CWE-327: Use of a Broken or Risky Cryptographic Algorithm
* CWE CATEGORY: OWASP Top Ten 2021 Category A02:2021 - Cryptographic Failures
* Vulnerability: The pyCrypto library and module DES is no longer actively maintained and is deprecated.
* Mitigation: Using the pyca/cryptography library can be effective at mitigating this vulnerability.



* 67 key = password + salt

68 m = hashlib.md5(key.encode('utf-8'))

69 key = m.digest()

* 79 key = password + salt

80 m = hashlib.md5(key.encode('utf-8'))

81 key = m.digest()

* CWE-327: Use of a Broken or Risky Cryptographic Algorithm
* CWE CATEGORY: OWASP Top Ten 2021 Category A02:2021 - Cryptographic Failures
* Vulnerability: Use of weak MD4, MD5, or SHA1 hash for security.
* Mitigation: Using SHA-256(SHA-512,etc) hash algorithm can be effective at mitigating this vulnerability.



* 70 (dk, iv) = (key[:8], key[8:])

71 crypter = DES.new(dk, DES.MODE\_CBC, iv)

72 ciphertext = crypter.encrypt(value.encode('ascii'))

* 82 (dk, iv) = (key[:8], key[8:])

83 crypter = DES.new(dk, DES.MODE\_CBC, iv)

84 encrypted\_value = base64.b64decode(value.encode('ascii'))

* CWE-327: Use of a Broken or Risky Cryptographic Algorithm
* CWE CATEGORY: OWASP Top Ten 2021 Category A02:2021 - Cryptographic Failures
* Vulnerability:U se of insecure cipher Crypto.Cipher.DES.new.
* Mitigation: Replace with a known secure cipher such as AES,etc can be effective at mitigating this vulnerability.



* 13 dbinit.db\_init()

14 mydb = mysql.connector.connect(

15 host="mysqldb",

16 user="root",

17 password="p@ssw0rd1",

18 database="assignment"

19 )

* 64 def des\_encrypt(value):

65 password = "qwerty123"

66 salt = '\x28\xAB\xBC\xCD\xDE\xEF\x00\x33'

* 76 def des\_decrypt(value):

77 password = "qwerty123"

78 salt = '\x28\xAB\xBC\xCD\xDE\xEF\x00\x33'

* 90 try:

91 mydb = mysql.connector.connect(

92 host="mysqldb",

93 user="root",

94 password="p@ssw0rd1",

95 database="assignment"

96 )

* CWE-259: Use of Hard-coded Password
* CWE CATEGORY: OWASP Top Ten 2021 Category A07:2021 - Identification and Authentication Failures
* Vulnerability: It is vulnerable to dictionary attacks with the use of weak and known usernames and passwords.
* Mitigation: It is useful to hide the authorized accounts by using different names for the username, instead of the names that indicate the authorized user such as root and admin.

For the password, it will be beneficial to use a password that consists of a minimum of 14 characters, uppercase and lowercase letters, special character and number combinations and that does not make sense.

It would be useful to use Multifactor Authentication.

* 21 cursor = mydb.cursor()

22 statement = "SELECT username,password from users where username = '"+username+"' and password='"+password+"'"

23 cursor.execute(statement)

* 47 token = secrets.token\_hex(16)

48 statement = "INSERT INTO tokens(token\_value) VALUES ('"+token+"')"

49 cursor.execute(statement)

* CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
* CWE CATEGORY: OWASP Top Ten 2021 Category A03:2021 – Injection
* Vulnerability: Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.
* Mitigation: Input Validation; when constructing SQL query strings, use stringent allowlists that limit the character set based on the expected value of the parameter in the request. This will indirectly limit the scope of an attack, but this technique is less important than proper output encoding and escaping.

Note that proper output encoding, escaping, and quoting is the most effective solution for preventing SQL injection, although input validation may provide some defense-in-depth. This is because it effectively limits what will appear in output. Input validation will not always prevent SQL injection, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, the name "O'Reilly" would likely pass the validation step, since it is a common last name in the English language. However, it cannot be directly inserted into the database because it contains the "'" apostrophe character, which would need to be escaped or otherwise handled. In this case, stripping the apostrophe might reduce the risk of SQL injection, but it would produce incorrect behavior because the wrong name would be recorded.

When feasible, it may be safest to disallow meta-characters entirely, instead of escaping them. This will provide some defense in depth. After the data is entered into the database, later processes may neglect to escape meta-characters before use, and you may not have control over those processes.



* 110 if \_\_name\_\_ == "\_\_main\_\_":

111 app.run(host='0.0.0.0', port=8000)

* CWE-605: Multiple Binds to the Same Port
* CWE CATEGORY: A05:2021 – Security Misconfiguration
* Vulnerability: When multiple sockets are allowed to bind to the same port, other services on that port may be stolen or spoofed.
* Mitigation: Restrict server socket address to known local addresses can be effective at mitigating this vulnerability.

Hizmet Reddi (DoS saldırısı), internete bağlı bir ana bilgisayarın hizmetlerini geçici veya süresiz olarak kesintiye uğratmayı, bir makineyi veya ağ kaynaklarını gerçek kullanıcılar için erişilemez hale getirmeyi amaçlayan bir siber saldırıdır. Genellikle hedef makinenin veya kaynağın gereksiz isteklerle aşırı yüklenmesi nedeniyle hedef sistemin gelen isteklere aşırı yüklenme nedeniyle yanıt verememesi şeklinde gerçekleştirilir. Saldırganlar, sahip olduklarından daha yüksek bant genişliğine sahip saldırılar için amplifikasyon tekniklerini kullanabilir ve DNS Protokol istatistiksel olarak diğer protokollere göre çok daha fazla kullanılmış/sömürülmüştür.

a.Özellikle DNS protokolünde bu tür saldırılar neden tercih ediliyor?

Lütfen kısaca açıklayınız.

b.DNS Güçlendirme Saldırılarında saldırgan için teknik sınırlamalar veya dezavantajlar var mı?

c.Lütfen DNS protokolünde amplifikasyon saldırıları için uygun bir IP adresi bulun ve bu tür sunucuların nasıl kolayca bulunacağını adım adım açıklayın.

Örnek bir uygulamanın kaynak dosyaları bir .zip dosyası olarak eklenir. Lütfen zip dosyasını bir dizinde açın. Sıkıştırılmamış dizindeki app.py dosyasındaki kod, birkaç OWASP Top 10:2021 güvenlik açığı içeriyor. Lütfen kodu inceleyin, içerdiği güvenlik açıklarını belirleyin ve tanımlanan güvenlik açıklarını çözmek için olası azaltmaları önerin. Her alan şunları içermelidir:

● Güvenlik açığı nedir?

● Güvenlik açığı neden ortaya çıkıyor (güvenlik açığının nedeni nedir)?

● Güvenlik açığını kapatmak için azaltma öneriniz nedir?

Not 1: Cevaplarınızı yazmanız için aşağıda 10 alan verilmiştir. Ancak, uygulama 10 güvenlik açığı içerebilir veya içermeyebilir.

Not 2: Başvuru sahipleri, gerekirse “docker-compose up” komutuyla docker kullanarak örnek uygulamayı çalıştırabilirler.