**WEAK SESSION ID**

Knowledge of a session ID is often the only thing required to access a site as a specific user after they have logged in, if that session ID is able to be calculated or easily guessed, then an attacker will have an easy way to gain access to user accounts without having to brute force passwords or find other vulnerabilities such as Cross-Site Scripting.

**Description:**

Weak session IDs can expose your users to having their session hijacked. If your session IDs are picked from a small range of values, an attacker only needs to probe randomly chosen session IDs until they find a match.

**Objective:**

This module uses four different ways to set the dvwaSession cookie value, the objective of each level is to work out how the ID is generated and then infer the IDs of other system users.

**Impact:**

If the session ID is weak your authentication scheme can be bypassed with relatively simple scripts, and unauthorized access can be gained by hackers.

**Prevention**:

1. Don’t pass session Ids in GET/POST variables:

Passing session IDs in query strings, or in the body of POST requests, is problematic. Not only does it make crafting of malicious URLs possible, but session IDs can be leaked in the following ways:

* If the user follows an out-bound link (the referer header will describe where the user browsed from).
* In the browser history and in bookmarks.
* In logs on your web server, and any proxy servers.

1. Regenerate the session ID at Authentication:

Session fixation attacks can be defeated by simply regenerating the session ID when the user logs in.

1. Accept only server-generated session IDs:

It is a good practice to ensure that only server-generated session IDs are accepted by your web server. (On its own, this won’t resolve session fixation vulnerabilities, though. A hacker can easily get a new server-generated ID and pass it onto a victim in a crafted URL.)

1. Timeout and replace old session ids:

Periodically replace session IDs as a second layer of defence, should they get leaked.

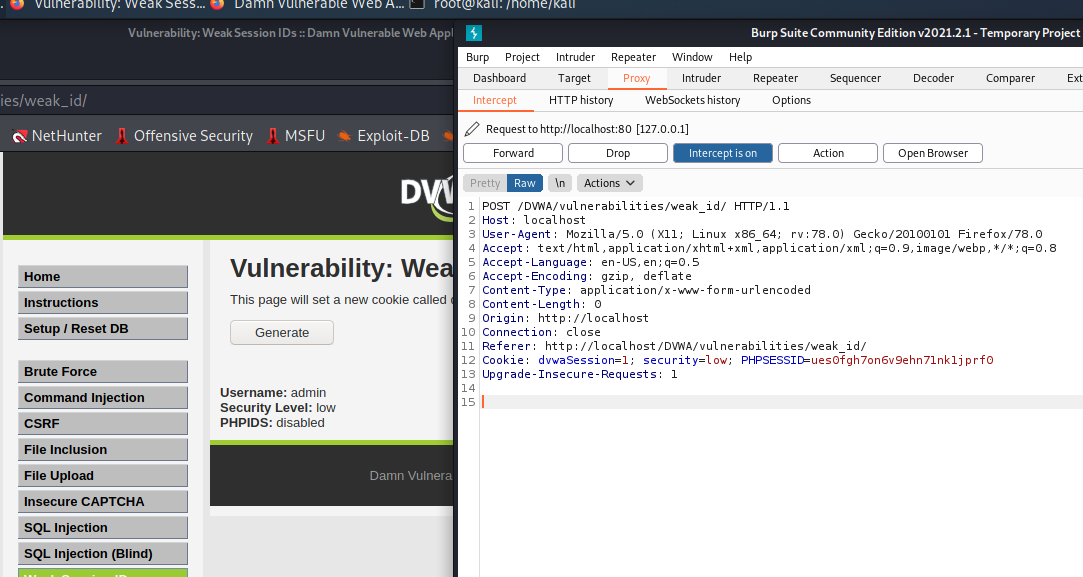
1. Implement a strong logout Function

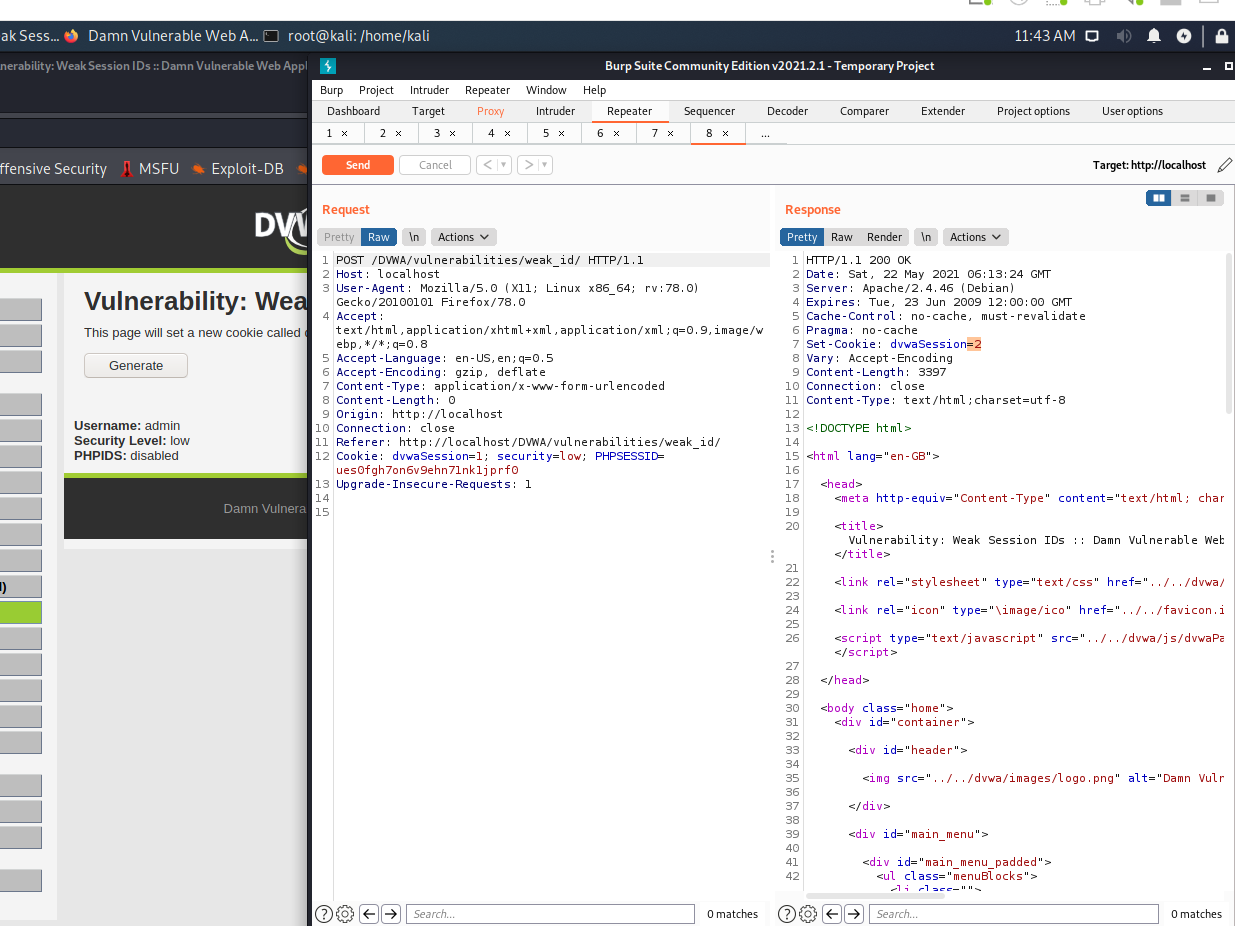
The logout function on your website should mark session IDs as obsolete.

**LOW**

**Steps to reproduce:**

1. Configure your browser and burp suite.
2. Go to the dvwa page and set level of weak session ids to the low level.
3. In the source code the session id is stored in dVWA session and the session value is incremented after each visit.
4. Click on the generate button and capture the request in the tool like burp suite
5. In request dvwa session id value is 1 now send request to repeater and click on the send button.
6. Now the session id is 2 again send request then session id is 3.
7. We can see that session id is incremented by 1.

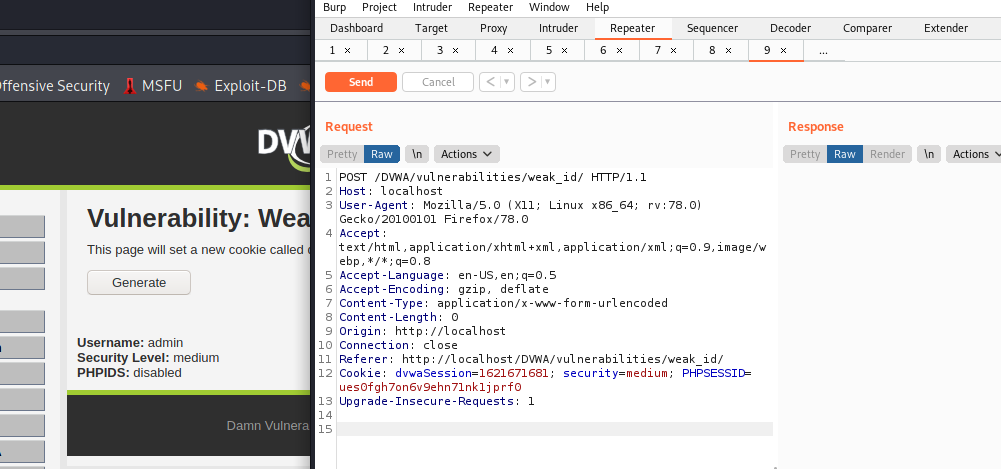


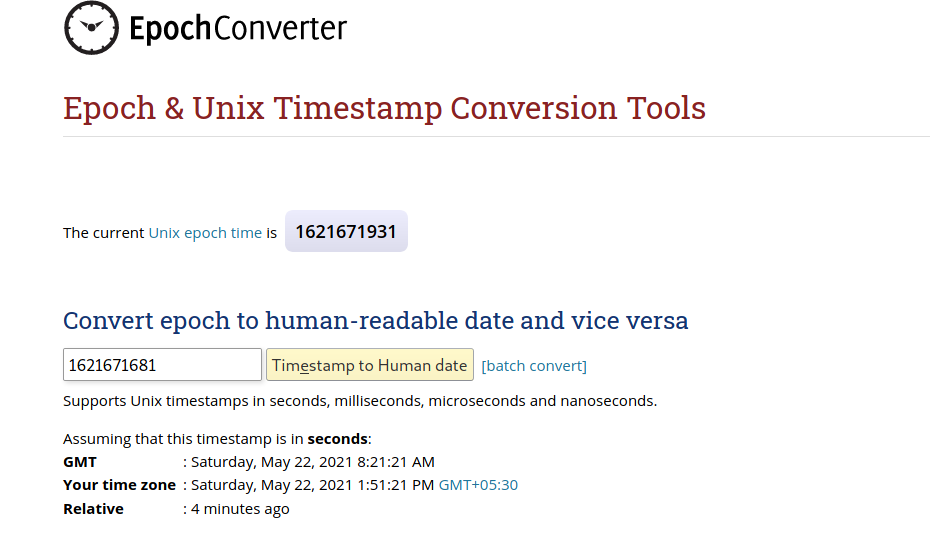


**MEDIUM**

**Steps to reproduce:**

1. Configure your browser and burp suite.
2. Go to the dvwa page and set level of weak session ids to the medium level.
3. In the source code the session id value is time so that means the session id value is based on timestamp.
4. Click on the generate button and capture the request in the tool like burp suite.
5. In request dvwa session id value is 1621671681 this is the timestamp.
6. Now we convert it using online websites to see what date and time it represents.





**HIGH**

**Steps to reproduce:**

1. Configure your browser and burp suite.
2. Go to the dvwa page and set level of weak session ids to the high level
3. In the source code the value of session id is increase by 1 each time and $cookie\_value = md5($\_SESSION['last\_session\_id\_high']); This means the session id is being converted into md5.
4. Now for this we open inspect element and go to storage tab.
5. Click on generate and see the value of dvwa session.
6. Copy the value of dvwa session and decrypt the md5 hash we get 1.
7. Now again click on generate and see that value again change and again we get 2.
8. We can see that session id is incremented by 1.

