Basic Web Attacks Problem Set

Description

The goal of this problem set is to exploit a fictional web application. First, you will need to gain access to the application as an authenticated user. Then, you will need to deanonymize the administrator. Finally, you will need to patch the vulnerabilities used to exploit the application.

To complete the problem set, you will need to ssh to your container at `$user@amplifier.ccs.neu.edu:$port`, where `$user` is your gitlab username and `$port` is your assigned ssh port (https://seclab-devel.ccs.neu.edu/snippets/6). Authentication is performed using any of your uploaded ssh public keys in gitlab.

You will also need to clone the problem set repository located at `git@seclab-devel.ccs.neu.edu:softvulnsec/prset02.git`.

Important Information	
Available	Tue 17 Feb 20:00 EST
Submission Deadline	Fri 27 Feb 18:00 EST
Gitlab URL	https://seclab-devel.ccs.neu.edu/softvulnsec/prset02 (https://seclab-devel.ccs.neu.edu/softvulnsec/prset02)

Password Hashes

The target application is running on port 3000 on your container. To access the application, you can use ssh to forward a local port on your machine to the container port 3000 using the `-L` option. For instance:

```
# From a shell
$ ssh -A -L3000:localhost:3000 -i $path_to_ssh_private_key -p $ssh_port $user@amplifier.ccs.neu.edu

# From your browser, access http://localhost:3000/
```

See `man ssh(1)` for more information on port forwarding.

You will be presented with a login page. Locate a SQL injection vulnerability in the source code of the application. Exploit this vulnerability to obtain pairs of password hashes and usernames for the application.

Authentication

The hashes you obtained in the previous step were created using a known-insecure algorithm. Study the source code of the application to identify the hashing algorithm. Write a program to crack at least one hash out of those you recovered in order to authenticate to the application.

The recommended approach is to perform a brute-force search of the space of possible passwords. A clue to reducing the search space can be found in the application source code.

The use of existing password crackers like John the Ripper, hashcat, or others is **not permitted** for this problem.

Deanonymize the Admin

After gaining access to the application, your next goal is to deanonymize the administrator. To accomplish this, locate and exploit a XSS vulnerability to inject content that will result in the administrator issuing a request to a web server that you control when the administrator visits the application. Record the administrator's IP address as well as a secret token that is embedded in a cookie.

To receive the request, you will need to set up a web application of your own that is accessible to the administrator. Feel free to use the framework of your choice (e.g., Flask (http://flask.pocoo.org/)). Port 5000 will be exposed on your container; running your application on that port will allow the administrator to access your application. The administrator will visit your application every few minutes.

Patch the Vulnerabilities

Fork the repository for this problem set in gitlab – your copy should have the URL `git@seclab-devel.ccs.neu.edu:$user/prset02.git`. Develop a patch to the original source files that removes the SQL injection and XSS vulnerabilities while preserving the intended functionality of the program.

Commit the patch to your repository (i.e., edit the original source code and commit the changed version).

Answer Submission

In your forked repository, commit a JSON object to `solution.json` with the following format:

```
{
    "user_hash": "<user's hashed password>",
    "user_password": "<user's cracked password>",
    "admin_addr": "<admin IP address>",
    "admin_secret": "<admin secret value from cookie>"
}
```

You are responsible for submitting valid JSON at the correct path. Use a validator if you're unsure about this, and double-check that your JSON follows the format above exactly.

In addition, commit the following source code to the specified locations:

- Your password cracker to `cracker/`
- Your XSS exploit to `xss/`
- Your deanonymization application to `deanonymize/`

Finally, commit a `README.md` that describes the vulnerabilities, your exploits, and how your patch fixes the vulnerabilities.

Push all commits to gitlab.

Extra Credit

Crack all of the user accounts and compute the size of the password space for this application. To receive credit, add the following fields to your `solution.json` submission:

```
{
    // original fields omitted...
    "password_space": <size of password space as an integer>,
    "passwords": [
        {
            "username": "<user's username>",
            "password": "<user's password>"
        },
        // ...
    ]
}
```

As usual, make sure that the resulting `solution.json` is valid JSON.

Updated Fri 07 Aug 2015 10:38 EDT
Revision master/52d826a