**Gezer phishing website**

[**https://gezer1.bgu.ac.il/meser/login.php**](https://gezer1.bgu.ac.il/meser/login.php)

[**https://github.com/amityahav/gezer-phishing**](https://github.com/amityahav/gezer-phishing)

**Amit Yahav 316328426**

**Ben Mor 324125913**

**Introduction:**

This phishing website we’ve created is an exact copy of the official grade’s website of Ben Gurion university. Our site enables any eager student that wants to view his grades to successfully view them, and as for the Hacker to easily retrieve all stolen credentials in a click of a button.

In this report we intend to take you step by step on planning, researching and implementing our phishing site and how we’ve managed to study Ben Gurion’s API behaviors and use it for our interests.

**A little bit of context:**

Amit is a 3rd year CS student in the university, this semester he is planning on taking a mini-project course about network security. Due to a high demand of this course all slots were taken.

After long failing negotiations with other students in order to try and get a slot he decides to take a different and illegal path. His plan is to steal a slot from students by stealing there’s university credentials and logging to their accounts in order to cancel their enrollment to the course hence giving him a free slot.

Amit is familiar with HTTP requests and Web APIs.

**Step 1: Research**

Amit decides to navigate to [**https://gezer1.bgu.ac.il/meser/login.php**](https://gezer1.bgu.ac.il/meser/login.php) and check what happens behind the scenes, hence he opens the ‘Inspect’ tab in google chrome in order to study the HTTP requests which are sent to BGU’s server whenever a user tries to log in.

When entering his own credentials, he sees the following request:

Text

Description automatically generated

With payload:

Graphical user interface, application

Description automatically generated

And for response, another html page is returned:

Graphical user interface, text, application, email

Description automatically generated

Amit then tries to make the same POST request using POSTMAN in order to study the payload which was sent. Username, pass and id are obvious but what the HELL is isheb: 1?

After some playing around and changing isheb: 0 Amit got another HTML page but now it was in English! Which means isheb == Is Hebrew? This flag is essential later when he will build the phishing site and make it more and more alike the real one.

In order to proceed to the grades page, one needs to agree the terms using the button ‘מסכים’ as above, which means another request is sent to BGU’s server! Let’s check that.

Using again the ‘inspect’ tab we see the following request:

Text

Description automatically generated

With Payload:

Graphical user interface, text

Description automatically generated with medium confidence

(agree is just the name of the button in Hebrew ‘מסכים’ hence it cannot be displayed)

And for response the desired grades page is returned:

Graphical user interface, table

Description automatically generated

(\*\*) Amit tries once again to make the same POST request using POSTMAN but this message returns: Your session has expired<br>Please login again at the starting page

Well... how then the server preserves the session of one user when moving from the login page to the agreement page and then to the grades page? With a Cookie of course.

Amit then decides to search within the former request for a special cookie which identifies the session of a client. As seen at the first POST request:



There are a lot of strange values here... Amit then tries to attach the ‘PHPSSID’ cookie into his last request (\*\*) but again with no luck the following message returns:

wrong use<br>If you use a Personal Firewall, please deactivate it<br>

Well... at least it’s a different one.

Amit then tries to check what headers are missing in his request which are present in the real request that is sent using the website, As seen in the actual request, this header occurs:



This URL belongs to the agreement page from before. Maybe the request needs some context from which it came from for security measurements. Amit attaches this header as well and BAM!! The response for the request is the authentic HTML page containing all Amit’s grades.

Amit now understands that if he can create a fake look alike website which asks for student’s credentials and once entering them, a POST request will be sent behind the scenes to the real BGU’s servers endpoint -> fetching the session id (cookie) -> creating another POST request with the cookie attached and the ‘Referrer’ field ,as written above, to the second endpoint (second request) then not only the students that will use it will be served with authentic HTML page containing all of their grades but he will be able to steal their credentials.

As for POC this sounds like a great idea, but a good social engineering strategy is required here.

How will Amit distribute his phishing site to students without them noticing it’s a fraud? We will try to answer that after a view of the design and implementation of the phishing site.

**STEP 2: Design and Implementation**

**Design**

The phishing site will be developed using Python with [Flask Framework](https://flask.palletsprojects.com/en/2.1.x/) for creating a web server which handles requests and serves HTML pages. The site will be deployed to the web using [PythonAnywhere](https://www.pythonanywhere.com/) with a similar domain **gezer1-bgu.info**.

In order to make our phishing site as most identical we had to use the same server’s endpoints, same website behaviors, same CSS file, static images and same HTML pages. All of those were ‘stolen’ from the original using the ‘inspect’ tab in Chrome and then written and stored at the app’s directory (more on that in the Implementation section).

**Endpoints**

* /meser/login.php – Login homepage in Hebrew.
* /meser/elogin.php – Login homepage in English.
* /meser/crslist.php – Student is redirected to this URL when his grades are ready for display.
* /meser/credentials – Secret endpoint for displaying all stolen credentials.

**Implementation**

All the code can be found [Here](https://github.com/amityahav/gezer-phishing), the code is fully commented and documented with detailed easy-to-understand explanations, in addition to that we will describe the project’s structure.

**Project’s Structure**

A picture containing text, receipt, screenshot

Description automatically generated

* **Libs directory**
  + **logic.py –** contains all the ‘business logic’, communication with BGU’s server, storing stolen credentials, fetching session cookie, etc.
  + **utils.py –** contains singleton implementation.
* **Website directory**
  + **\_\_init\_\_.py –** server initialization and endpoints registrations.
  + **courses\_list.py –** Request handler for /meser/crslist.php endpoint.
  + **credentials.py –** Request handler for /meser/credentials endpoint.
  + **elogin.py –** Request handler for /meser/elogin.php endpoint.
  + **login.py –** Request handler for /meser/login.php endpoint.
  + **Static directory –** contains all static elements needed such as css and bgu picture.
  + **templates directory –** contains all static HTML pages of the website.

**Flow**

Diagram

Description automatically generated

1. Student enters http://gezer1-bgu.info/meser/login.php in the browser (GET request).
2. Our server returns ‘login.html’ page to the browser.
3. Student enters credentials at the login page and clicks ‘המשך’, a POST request is sent to /meser/crslist.php endpoint with the form-data of {username = ‘username’, pass = ‘password’, id = ‘ID’

, isheb = ‘1’}.

1. Our server forwards the request to BGU’s server at /meser/main.php endpoint attempting to authenticate and create a session cookie.
2. If the authentication failed an HTML file with ‘errorfound’ string inside will be returned and our server will return ‘login\_fail.html’/’elogin\_fail.html’ to the user depends on ‘isheb’ flag which indicates which language (Hebrew or English) and then the student will start again from step 1. If the authentication was successful BGU’s server will create a session cookie which we can fetch.
3. With the session cookie attached and additional header ‘Referrer: https://gezer1.bgu.ac.il/meser/main.php’ another POST request is sent from our server to BGU’s server at /meser/crslist.php.
4. Upon succeeding BGU’s server returns the desired grades HTML page to our server where we decode it and inject the static original CSS file we’ve stolen.
5. We then return the authentic grades HTML file to the student where he can see his grades.

**Usage Examples:**

**Hebrew Login Page:**

Graphical user interface, text, application

Description automatically generated

**English Login Page:**

**Graphical user interface, text, application

Description automatically generated**

**Wrong Credentials:**

**Graphical user interface, application

Description automatically generatedGraphical user interface, application, email

Description automatically generated**

**Table

Description automatically generatedHebrew grades page:**

**English grades page:**

**Graphical user interface, text, application

Description automatically generated**

**STEP 3: Social Engineering**

The phishing website is finally up and running but there are a few obstacles that need to be overcome. Here are some **issues** that we’ve encountered:

1. The website Is up and running locally but in order to distribute it, it must be deployed to the internet.
2. A reasonable domain name is needed in order to not arouse suspicion among students.
3. A smart way to distribute the phishing site among students.

**Answers:**

1. We used (conceptually) [PythonAnywhere](https://www.pythonanywhere.com/) in order to deploy our Flask app to the internet.
2. In the PythonAnywhere service there’s an option for 5$ monthly subscription which grants you a custom domain name for your app. Using that we’ve picked the most similar domain name we could find **gezer1-bgu.info**.
3. There are few options to distribute the phishing site:
   1. Via WhatsApp – We could distribute our phishing site in all current courses WhatsApp groups 3 days after an exam was conducted with the following text:
      1. ״ http://gezer1-bgu.info/meser/login.php״יש ציונים!!!
   2. Via Facebook – There are plenty of BGU’s Facebook groups where an anonymous user can post a message containing the URL of the phishing site with the text of

״היי לכולם לעוד מישהו לא עובד הגזר? http://gezer1-bgu.info/meser/login.php משום מה אני מנסה להיכנס וזה טוען ואז נתקע..״

The timing of distributing the site is crucial for its success. Exam periods are known as stressful periods when all students are expecting their grades. Hence this is a good time to strike and distribute the phishing site with a message of ‘gezer site is not working can anyone check?’.

**Strengths and Weaknesses**

* We are aware that we cannot ensure the success of our phishing because there will always be students that are more aware and some less aware to the domain name.
* Our site has 95% of the functionalities the real site has – after the student is served with the grades page all the links inside it will be broken. We are aware that it affects the authenticity of the website but for a `quick hack` it is sufficient as when a student gets there his credentials have already been stolen.
* Our site looks exactly as the original site, style wise and endpoint wise.

**Summary**

This project has enriched our knowledge on web servers and http requests a lot.

We’ve also experienced in reverse engineering the backend of BGU’s server and used it for our interests. It was a lot of fun! as for Amit, he got a free slot for the course.