HAB-01 Write-up

Introduction

This is a quick write-up for HAB-01 version 2019-02-17 The box is very much still work in progress and I will not spend a lot of time keeping this write-up up to date.

With that in mind, let's get to it:

Todo

- Add references to OTG
- Add links to relevant tools (and techniques?)

Useful tools

- Browser Developer Tools
- cuRL / wget
- nmap
- nikto
- burp suite / ZAP
- Wireshark

Challenges

WHO'S YOUR SOURCE

Very basic. Navigate to www.hackab.se and view the source of the document. The flag is in the meta-tags.

KILL ALL HUMANS

Still on a basic level. The title obviously refers to robots.txt which can be accessed via www.hackab.se/robots.txt. The file instructs all crawlers not to visit the following URIs:

- `/!admin`
- `/tpl`

Or any any files with the following extensions:

- `.php`
- `.phps`

Additionally, a sitemap is referenced with the URI www.hackab.se/sitemap.xml.

These are all hints! More on that later.

GLOBETROTTER

This one is harder. sitemap.xml (referenced in robots.txt) lists www.hackab.se/international as a location. However, the resource may only be accessed from an IP originating from from a set of regions (such North America, etc) or localhost (127.0.0.1). The candidate may solve this challenge in a few ways:

1. By using a personal or public proxy located in a white-listed region
2. By using Google Translate, the 'Anonymous View' functionality of Startpage.com or a equivalent 'indirect' proxy.
3. By using Hack ABs proxy (more on this later).

Please note that if the box is run a VM, the first two options are not applicable since the host is not accessible from the Internet.

CWE-209

The title refers to CWE-209 (Information Exposure Through an Error Message). So obviously the challenge is about triggering an error somehow. There are a few ways to create an error that gives away the intended information:

- By visiting www.hackab.se/index.php (remember the hint from robots.txt?)
- By trying to access an URI with an invalid (e.g. non-whitelisted) character, for example www.hackab.se/!admin.

The application will return a 500 error code and display a rather friendly message explaining what caused the error. But but no sensitive information is directly visible. To find the "sensitive" information, the candidate must either:

1. View the source code of the response.
2. Review www.hackab.se/js/error.js and notice that the script looks for a 'debug' key in the sessionStorage object, and create that key manually (by using Developer Tools).

Other than the flag for the challenge, the error message gives away a few interesting pieces of information:

- The current working directory (`cwd`) of the script.
- The regular expression that checks the URI for invalid characters

- A list of templates (see the "IDOR" challenge below)
- A list of regions that may access www.hackab.se/international.

NOW TELL ME YOUR REAL SOURCE

Earlier, the candidate may have noticed that .phps extensions may not be indexed by crawlers. The intention is that the candidate should visit www.hackab.se/index.phps to reveal the source of the PHP script (which is a very uncommon, but real function in Apache's mod_php).

Other than the flag for the challenge (which is hidden in the 'doc-block', the resource reveals the following:

- The "WAF" function that checks URIs for invalid characters.
- The logic and if-statements that is needed for any page to render.
- References to the 'administration' and 'international' resources.
- That the argument passed to the function that checks whether access is allowed to www.hackab.se/international is `$_SERVER['REMOTE_ADDR']`

IDOR

The title refers to the 'Insecure Direct Object Reference' weakness. When causing the application to throw an error message, a list of templates is revealed to the candiate.

The objective of this challenge that the candiate should use that information in order to find the unreferenced template available at www.hackab.se/tpl/flag-backup.tpl.

REDACTED

A number of .PDFs are available for download at www.hackab.se/tjanster. The resources are referenced by IDs (i.e. `?id=1`, `?id=2`, etc.). The objective of this challenge is for the candidate to try to access an unreferenced id. Granted, a rather 'special' PDF can be found at www.hackab.se/files?id=6.

This PDF contains insufficiently blacked out information that may be revealed by simply selecting the text with the mouse cursor.

IT WORKS!

'It works!' is a reference to the default content of a freshly installed Apache HTTP server. Typically, a web server is accessed via a domain name (such as www.hackab.se), However, accessing a server directly via IP often reveals other information, such as a flag. :)

CLOSE BUT NO CIGAR

An administrative portal is available at www.hackab.se/administration. However, users must successfully authenticate in order to access it's contents. Typically, applications come with default crendetials such as admin/admin or, as in this case, admin/password.

If the candidate tries to authenticate to the portal using `admin` as username and `password` as password, the login is successfull, but access is restricted due to the account being disabled. The candidate is, however, rewarded a flag for their effort.

WELCOME, SUPERMAN. I LOVE YOU!

The title is a reference to three of the most common passwords of 2018. A list of employees is available at www.hackab.se/om-oss. This page contains the e-mail addresses of all employees. Also note that the 'username' input field has a placeholder that reveals that the expected syntax is username@hackab.se.

The objective of this challenge, obviously, is to successfully authenticate to the administrative portal. There are a few ways to do this:

1. By guessing or using brute-force and the above information.
2. By finding the password revealed in clear text in the picture of 'Thomas' (more on this later).
3. By hijacking a session belonging to the user 'Steve' (with session id `SEFCX1NFU1MtNDIw` which is `HAB_SESS-420` in base64-encoding).

Upon successful authentication, a flag can be found in a message sent to all users.

ROBAC

The title is a reference to the concept of 'Role-Based Access Control'. The candidate may have noticed that upon successfully authenticating to the administrative portal, a `HAB_ROLE` cookie is created with the value `user`. By manually manipulating this cookie and changing it's value to `admin`, a 'Server Status' section is revealed to authenticated users.

Other than containing a flag, and some non-interesting server status, the section references a CGI-script (more on this later).

YOU'VE GOT MAIL

Observant candidates may have noticed that the administrative portal makes XMLHttpRequests to www.hackab.se/administration/messages?uid= to retrieve all messages sent to the authenticated user. It just so happens that this resource is lacking proper security checks, namely:

1. Missing controls for authentication and authorization
2. User IDs (uids) are the md5 sum of the username, and consequently easily guessable

Several messages contain interesting information, but the following is specific to this challenge: If the candidate tries to access the messages of user 'Kathleen' (uid `788aaa9041b7a573943488941b0a4a3e`) an email containing an attachment is revealed. In order to access the contents of the file, the candidate must:

1. Spot the attachment,
2. Base64-decode it to retrieve a .zip file,
3. Guess or bruteforce the password `password`.

COMPUTER-GENERATED IMAGERY?

The title is a reference to the Common Gateway Interface (CGI) protocol. The administrative portal references a CGI-script used for status monitoring. By enumerating the box, the candidate should find the script located at www.hackab.se:8008/cgi-bin/status.cgi. This script offers some more or less usual information through common UNIX shell commands such as `netstat`, `ps`, etc (more on these later).

By passing `?command=help` as arguments to the script, it is revealed that the non-standard command `sf` ('show flag') is also available. By issuing this command a flag is returned.

HOME SWEET HOME

The title is a reference to the concept of UNIX home directories. This one may be hard, but has a few hints:

1. In an e-mail reminder sent from the user 'Steve' to himself, a lack of proper backup system is mentioned. Specifically, `userdir` is pointed out to be a particularly bad idea (`UserDir` is Apache module which allow system users to make a section of their home directory publicly available via HTTP).
2. By monitoring the `ps` command of the aforementioned CGI-script observant candidates may notice that a backup job is scheduled to run every minute, in the command line of the process, the path `/home/steve/public_html/` is revealed.

By using this information, the candidate is expected to find the directory index available at `/~steve/` if accessing the web server directly via IP. A few interesting resources can be found here (more on this later), but the flag specific to this challenge may be found in `/~steve/todo.txt`.

GOT GPU?

The title is a reference to graphic card processors units (GPUs) being more capable than computer processor units (CPUs) for certain tasks (such as recovering passwords). The `userdir` mentioned in the previous challenge makes a complete backup of the main Hack AB website available for download. This file includes a lot of sensitive information, such as:

1. All scripts and other static files that make the web application.
2. A `.htpasswd` file that contain authentication details, including hashed passwords.
3. The public as well as the private SSL keys used by Apache (more on this later).

The candidate is expected to find the `.htpasswd` file and to recover the passwords by brute force or dictionary attacks. Most passwords are weak and will be recovered within seconds. However, the password for the user 'Steve' is relatively strong and will require some effort from the candidate. As a hint, a message sent by 'Steve' to all users may be found in the administrative portal. In this message, everyone is reminded to use secure passwords but 'Steve' inadvertently reveals that his password is '8 characters long with numbers, lower- as well as uppercase characters'. The candidate is expected to use this information to tailor an attack to more effectively recover the password.

Once the password (`a40xGGh1`) is recovered, it may be used to access a password-protected resource available at `/~steve/private/flag.txt`

READY, GET SET, TXT

The title is a reference to to TXT (text) DNS-records. The candidate is expected to gather information from all sources, including DNS.

By querying hackab.se's name servers, a TXT record including a flag may be revealed.

THERE'S NO JWTO WAYS ABOUT IT

The title is a reference to JWT (JSON Web Tokens) and the fact that there are two ways to solve this challenge. An 'Open Data API' is available at api.hackab.se:8008/. By reading the documentation the candidate is expected to:

1. Send GET requests in order to retrieve data from 'public' endponts (such as `api.hackab.se:8008/services`).
2. Retrieve a list of documented (and undocumented) endpoints by sending a GET request to `api.hackab.se:8008/endpoints`
3. Authenticate to- and retrieve a 'guest' token by sending a POST request to `api.hackab.se:8008/login`.
4. Send authenticated GET requests to non-public endpoints.

The objective of this challenge is to authenticate to the API as an `admin` user. There are two ways to achieve this:

1. Steal a token inadvertently leaked by the user 'Steve' (this token can be found in an Apache access log available to to authenticated users at ftp://hackab.se/logs).
2. Guess that the private key used to sign the tokens has not been changed from the library defaults and still is `super-secret` (literally) and manipulate the token to grant 'admin' access.

In order to give the candidate a hint, some 'pointers' have been sent by 'Steve' to his colleagues 'Tobias' and 'Miriam' (this message may be found in the administrative portal).

EXTRA-INTRANET

The title is a reference to an 'intranet' resource inadvertently being accessible from the Internet. This one may be hard, because there are no explicit references or hints to the solution. The 'intranet' is listening on `127.0.0.1:31337` and can not be be accessed directly. However, there is a proxy listening on port `3128`. The proxy has been configured to allow authenticated users to use it in order to access a set of ports listening on the local host (this information may be revealed by using the `netstat` command of the CGI-script mentioned earlier). By using this information, the candidate may be able to figure out that the intranet can be accessed by authenticating to the proxy and sending a request to `http://127.0.0.1:31337`.

The flag is revealed directly upon a successful request, there is no intranet. :P

MON0ITORED AT WORK?

The title is a reference to that in order to sniff Wi-Fi network traffic, the network adapter must be set to `monitor mode` which typically creates a new, virtual, interface called `mon0`. As mentioned in passing, an FTP server is listening on port `21`. The FTP allows anonymous- as well as authenticated access (but certain resources are only available to authenticated users). A Wireshark capture (`.pcap`) may be downloaded anonymously from `ftp://hackab.se/pub/`. This file contains a capture of two clients exchaning data on the `HackAB` WPA2-protected Wi-Fi-network. In order to retrieve the flag, the candidate must:

1. Download and analyze the capture file.
2. Realize that it contains a log of Wi-Fi traffic protected by WPA2.
3. Use tools to perform a dictionary attack in order to retrieve the WPA2 pre-shared key.
4. Configure Wireshark to decrypt Wi-Fi traffic using the pre-shared key.
5. Use a function in Wireshark to export a file containing the flag.

MISHAP

The title is a reference a configuration error that allows the weak/deprecated `AES128-SHA` cipher to be negotiated by clients connecting to Apache. There is a 'legacy' service running on the local host. This service is using HTTP/1.0 and TLS 1.0 (with AES128-SHA) when sending data to the web application using HTTPS POST requests, which is causing errors. The 'Steve' user is annoyed by this fact, and in his attempt to resolve the issue he used `tcpdump` to create a traffic dump. This log is available to authenticated FTP users. In order to solve this challenge, the candidate must:

1. Download and analyze the capture file.
2. Realize that it contains a log of encrypted HTTPS traffic.
3. Import the private SSL key (found in a previous challenge) into Wireshark.
4. Configure Wireshark to decrypt HTTPS traffic using the private key.
5. Find the POST requests that contain the flag.

Please note that this challenge may, possibly, be solved by solved without the private SSL key by using `Wiener's attack` but this has not been tested.

UNDOCUMENTED/BONUS CHALLENGE

This is a "side-quest" created to honor the typical tools and techniques used in CTF's. The challenge does not contain a flag. In order to "complete" the challenge the candidate must:

1. Discover the content available at www.hackab.se/medarbetare/thomas.
2. (Optionally) realize that there are larger images available via `-medium`, `-large` or `-original`.
3. Find the meme that may be discovered by adjusting the brightness and contrast of the image.
4. (Optionally) discover the hint hidden in the EXIF data of the image.
5. Use `binwalk` or other tools to extract a hidden file within the image.
6. Hex-decode, base64-decode, reverse and decrypt a Caesar cipher to reveal a key.
7. Use the key to decrypt XOR a file containing Base-2 (binary) data.