Default Credentials

It is common to find devices with `default credentials` due to human error or a breakdown in/lack of proper process. According to Rapid7's Under the hoodie report for 2020, Rapid7's gained access using known default credentials or guessable accounts during 21% of their engagements

Unfortunately, default credentials are also used by maintainers working on Industrial Control Systems (ICS) environments. They prefer having well-known access credentials when doing maintenance than relying on a company user, who is often not cyber security-savvy, to store a complex set of credentials securely. All penetration testers have witnessed such credentials stored on a Post-it note. This does not justify default credentials being used, though. A mandatory step of security hardening is changing default credentials and using strong passwords at an early stage of application deployment.

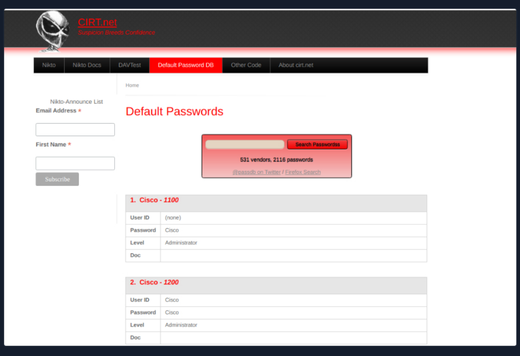
Another well-known fact is that some vendors introduce hardcoded hidden accounts in their products. One example is CVE-2020-29583 on Zyxel USG. Researchers found a hardcoded account with admin privileges and an unchangeable password. As you can see on the NIST website, Zyxel is not alone. A quick search against the CVE list with "CWE-798 - use of hardcoded credentials" as filter returns more than 500 results.

There is an old project, still maintained by CIRT.net and available as a web database used to collect default credentials split by vendors. Also, SecLists has a good list based on CIRT.net. The two options above have some overlap but also differences. It is a good idea to check both lists. Back to SCADA, in 2016 SCADA StrangeLove published a list of known passwords for industrial systems, both default and hardcoded, on their own GitHub repository.

Even by today's security standards, it is common to come across well-known/poor sets of credentials in both critical and non-critical devices or applications. Example sets could be:

- `admin:admin`
- `admin:password`

It's always a good idea to try known or poor `user`/`password` sets with the help of the abovementioned lists. As an example, after having found a Cisco device during a penetration test, we can see that passdb contains 65 entries for Cisco devices:



Depending on which device we have found, for example, a switch, a router, or an access point, we should try at least:

- `empty:Cisco`
- `cisco:cisco`
- `Cisco:Cisco`
- `cisco:router`
- `tech:router`

It should be noted that we may not find default or known credentials for every device or application inside the lists mentioned above and databases. In this case, a Google search could lead to very interesting findings. It is also common to come across easily guessable or weak user accounts in custom applications. This is why we should always try combinations such as: (`user:user`, `tech:tech`).

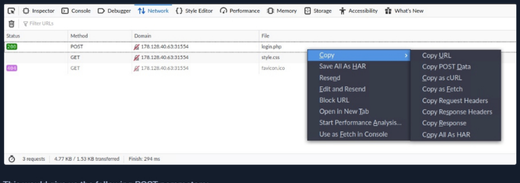
When we try to find default or weak credentials, we prefer using automated tools like `ffuf`, `wfuzz`, or custom Python scripts, but we could also do the same by hand or using a proxy such as Burp/ZAP. We encourage you to test all methods to become familiar with both automated tools and scripting.

Hands-On Example

To start warming up, download this Python script, read the source code, and try to understand the comments. If you want to try this script against a live environment, download this basic PHP code and place it on a web server that supports PHP. It will be helpful while trying to solve the next question.

Before we can start attacking any web page, we must first determine the URL parameters accepted by the page. To do so, we can use `Burp Suite` and capture the request to see what parameters were used. Another way to do so is through our browser's built-in developer tools. For example, we can open Firefox within the Pwnbox and then bring up the Network Tools with `[CTRL + SHIFT + E]`.

Once we do this, we can try to log in with any credentials (`test:test`) to run the form, after which the Network Tools would show the sent HTTP requests. Once we have the request, we can right-click on one of them and select `Copy` > `Copy POST data`:



This would give us the following POST parameters:

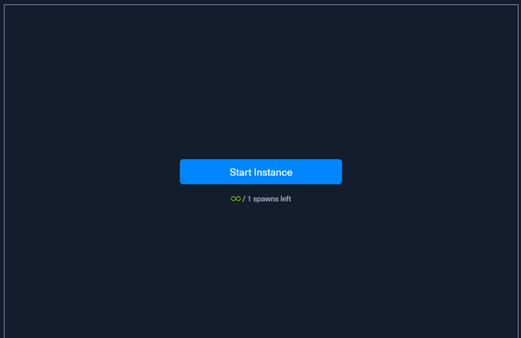
Code: bash

```
username=test&password=test
```

Another option would be to use `Copy` > `Copy as cURL command`, which would copy the entire `cURL` command, which we can use in the Terminal to repeat the same HTTP request:

```
Govardhan Gujji22@htb[/htb]$ curl 'http://URL:PORT/login.php' -H 'User-Agent: Mozilla/5.0 (Windows NT 10.
```

As we can see, this command also contains the parameters `--data-raw 'username=test&password=test'`.



Waiting to start...

Questions

Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

Target: Click here to spawn the target system!

+ 1 Inspect the login page and perform a bruteforce attack. What is the valid username?

advantech

Submit Hint

Previous Next

Mark Complete & Next

Cheat Sheet
Resources
Go to Questions

Table of Contents

Broken Authentication
What is Authentication
Overview of Authentication Methods
Overview of Attacks Against Authentication

Login Bruteforcing
Default Credentials
Weak Bruteforce Protections
Bruteforcing Usernames
Bruteforcing Passwords
Predictable Reset Token

Password Attacks
Authentication Credentials Handling
Guessable Answers
Username Injection

Session Attacks
Bruteforcing Cookies
Insecure Token Handling

Skill Assessment
Skill Assessment - Broken Authentication

My Workstation
OFFLINE
Start Instance
∞ / 1 spawns left