NETCOMP 3.0

1. I swear this is not a web or reverse

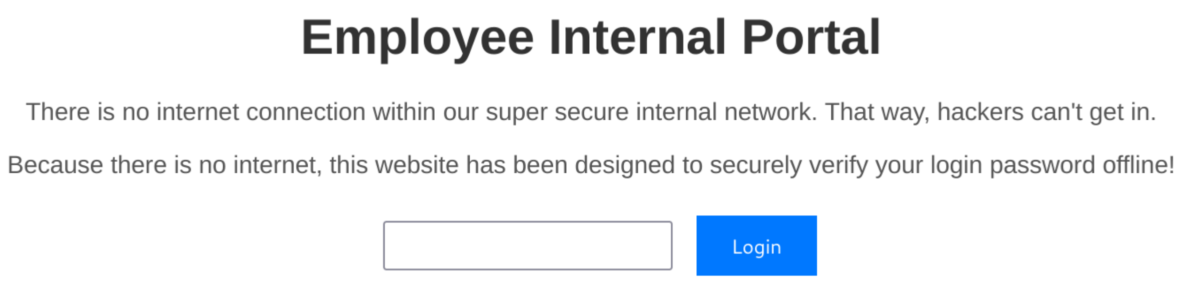
Cryptography

It's been a few months since I only reported HTTP Headers findings 😭

Today, I am pentesting a company's internal web app and wonder if you could help me get a critical finding here...

`Author: BerlianGabriel`

Downloadable file: password.html



Employee Internal Portal

There is no internet connection within our super secure internal network. That way, hackers can't get in.

Because there is no internet, this website has been designed to securely verify your login password offline!

After opening the file, the login page appears.
If you look at the html code,

```
function secureHash() {
        const passwordInput = document.getElementById('passwordInput').value;
        const expectedString = "7X!7|!@V|7eV77_!|@8S";
        const magic = [
                BigInt('0x1fa9787f52d6819dac3e51c96c9850ac9a68a000'),
                BigInt('0x551e7b2ade66a9cd21538d24f8232eb9e3c6a00'),
                BigInt('0x685130edf575c5fd89b4ea52d8ce440fb75d40'),
                BigInt('0x4d2b06845e7f210fd15f3697fe234c69919a0'),
                BigInt('0x267227d769f1422427c2f550f7852c59bfec'),
                BigInt('0xd9fd323c23dd5a26579cb53a8a42996b38'),
                BigInt('0x388a9fbf545b3b1a5e4b80376e94de767'),
                BigInt('0xadef7b085371d7244d43d0011e7c6d5'),
                BigInt('0x18cbc26aefc3b3b1ef4588ce4acc6b'),
                BigInt('0x296e5ed6f99d55e5efb08eb856e9'),
                BigInt('0x314ef6584d10a8c5226f105685'),
                BigInt('0x2798a7a450463592994fc72f'),
                BigInt('0x133caaa3da819c1ca0087d'),
                BigInt('0x445974d799d8bcf9c3b'),
        ];
        let magic2 = BigInt('0x2971713e56d0006e6a0b48126ca34000');
        let calculatedString = '';
        let oneChar = 0;
        let result = BigInt(0);
        let nresult = BigInt(0);
        for (let i = 0; i < passwordInput.length; i++) {
                result = BigInt(0);
                oneChar = -passwordInput.charCodeAt(i);
                for (let j = 0; j < magic.length; j++) {
                result *= BigInt(oneChar);
                result += magic[magic.length - 1 - j];
                }
                nresult = result % magic2;
                result = Number(-result / magic2);
                result += (888 - result) * (result > 127);
                result += (888 - result) * (!(nresult == 0));
                result += (888 - result) * (result < 33);
                calculatedString += String.fromCharCode(result);
        }
        alert(`${calculatedString}`)

        if (calculatedString === expectedString) {
                alert('Congrats, you are in! Wrap the password with Netcomp{} and submit it as
the flag.');
        } else {
                attempts--;
                if (attempts === 0) {
                document.getElementById('passwordInput').disabled = true;
                document.querySelector('button').disabled = true;
                alert('Too many wrong attempts, you are blocked from accessing this website
');
                } else {
                alert(`Remember harder. You have ${attempts} attempts left.`);
                }
```

There is a method where there is password hashing with the secureHash() function. And if the password matches the expected string, then we can access it.

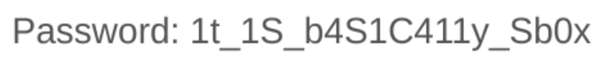
Since the expected string is: "7X!7|!@V|7eV77_!|@8S", we can work backwards to reverse the Hashing process and
determines the input password that produces this output.

The following is the script I used for reverse hashing:

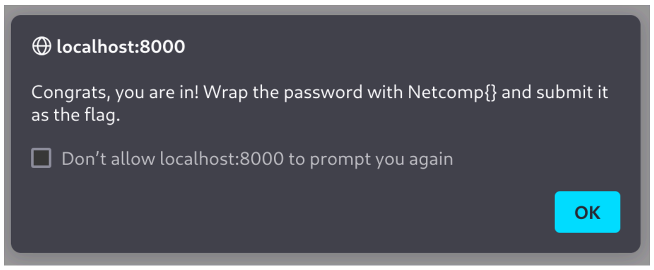
```
!DOCTYPE html>
<html>
```

```
<head>
        <title>Reverse Hashing</title>
        <style>
        body {
        font-family: Arial, sans-serif;
        text-align: center;
        }
        h1 {
        color: #333;
        }
        p {
        color: #555;
        }
        </style>
</head>
<body>
        <p id="status"></p>

        <script>
        const expectedString = "7X!7|!@V|7eV77_!|@8S";
        const magic = [
        BigInt('0x1fa9787f52d6819dac3e51c96c9850ac9a68a000'),
        BigInt('0x551e7b2ade66a9cd21538d24f8232eb9e3c6a00'),
        BigInt('0x685130edf575c5fd89b4ea52d8ce440fb75d40'),
        BigInt('0x4d2b06845e7f210fd15f3697fe234c69919a0'),
        BigInt('0x267227d769f1422427c2f550f7852c59bfec'),
        BigInt('0xd9fd323c23dd5a26579cb53a8a42996b38'),
        BigInt('0x388a9fbf545b3b1a5e4b80376e94de767'),
        BigInt('0xadef7b085371d7244d43d0011e7c6d5'),
        BigInt('0x18cbc26aefc3b3b1ef4588ce4acc6b'),
        BigInt('0x296e5ed6f99d55e5efb08eb856e9'),
        BigInt('0x314ef6584d10a8c5226f105685'),
        BigInt('0x2798a7a450463592994fc72f'),
        BigInt('0x133caaa3da819c1ca0087d'),
        BigInt('0x445974d799d8bcf9c3b'),
        ];
        const magic2 = BigInt('0x2971713e56d0006e6a0b48126ca34000');

        function computeHashChar(inputChar) {
        let result = BigInt(0);
        const oneChar = -inputChar.charCodeAt(0);
        for (let j = 0; j < magic.length; j++) {
                result *= BigInt(oneChar);
                result += magic[magic.length - 1 - j];
        }
        let nresult = result % magic2;
        result = Number(-result / magic2);
        result += (888 - result) * (result > 127);
        result += (888 - result) * (!(nresult == 0));
        result += (888 - result) * (result < 33);
        return String.fromCharCode(result);
        }

        function reverseHash() {
        let password = '';
        for (let i = 0; i < expectedString.length; i++) {
                const targetChar = expectedString[i];
                let found = false;
                for (let charCode = 0; charCode < 128; charCode++) {
                const inputChar = String.fromCharCode(charCode);
                const hashChar = computeHashChar(inputChar);
                if (hashChar === targetChar) {
                password += inputChar;
                found = true;
                break;
                }
                }
                if (!found) {
                document.getElementById('status').innerText = `Could not reverse character at
position ${i}`;
                return;
                }
        }
        document.getElementById('status').innerText = `Password: ${password}`;
        }

        reverseHash();
        </script>
</body>
</html>
```

After running the file:


Password: 1t_1S_b4S1C411y_Sb0x

Then we insert it in the website:


localhost:8000

Congrats, you are in! Wrap the password with Netcomp{} and submit it as the flag.

Flag: **Netcomp{1t_1S_b4S1c411y_Sb0x}**