
Conceptos Básicos y Algoritmos Fundamentales

**Sistema de ordenamiento de IPs con Heaps**

Juan Carlos Garfias Tovar

A01652138

David Alonso Cantú Delgado

25 de octubre del 2020

## Introducción

En la actualidad los sistemas se ven en peligro debido a la gran cantidad de vulnerabilidades que se ven explotadas ya sea por zero day exploits, update exploits u otros elementos que son usados con el fin de robar información. Sistemas como Google auth o Facebook auth están constantemente utilizando registries para detectar posibles comportamientos inusuales como cambios drásticos en la ubicación del usuario, intentos continuos de login fallido, fallos en captcha, etc. Estos sistemas permiten verificar si el usuario real está intentando acceder al sistema o no, y en caso de detectar una anomalia reportarlo por correo y poner la cuenta en cuarentena.

El objetivo del proyecto IP verifier es lograr crear una demostración de como puede ser implementado un sistema similar a través de árboles, los cuales utilicen a la frecuencia de accesos, errores de login y puertos utilizados. Esto para posteriormente ser enviados a algún algoritmo de machine Learning o similar que sea capaz de detectar con precisión si un login es autentico o no.

## Funcionamiento del proyecto

IP verifier utiliza un struct llamado IPRegistry, en este se encuentran como atributos la IP, frecuencia, errores y puertos. Estos datos permitirían dar a conocer si un usuario o IP tiene un comportamiento indebido de manera sencilla. Esta estructura toma los errores y los puertos como vector, sin embargo, al momento de ser evaluados y pasados al JSON, estos son convertidos en un map, el cual tiene como llave el código de error y como value la frecuencia de aparición del error en los registros. De este modo seria posible detectar si un login ha sido autentico, se le ha olvidado la contraseña a un usuario o si su cuenta se ha visto comprometida/IP infectada. El código parte por la lectura de mas de dieciséis mil registros, los cuales van siendo partidos en los diferentes campos, siendo evaluados e insertados en una doubly linked list de IPRegistry, de esta manera a medida que se leen las líneas se van aumentado los valores de frecuencia tanto en aparición como en errores y puertos. Una vez completado este proceso se realiza un heap sort, este algoritmo tiene una complejidad de O (n log n). Permitiendo asi el ordenamiento de manera rápida, posteriormente se da la posibilidad de mostrar los cinco valores con mas frecuencia o de menor. Esta lista ya ordenada puede también ser exportada como JSON, facilitando asi su lectura y visualización, agilizando el proceso de lectura y permitiendo exportar los datos a otros lenguajes de programación.

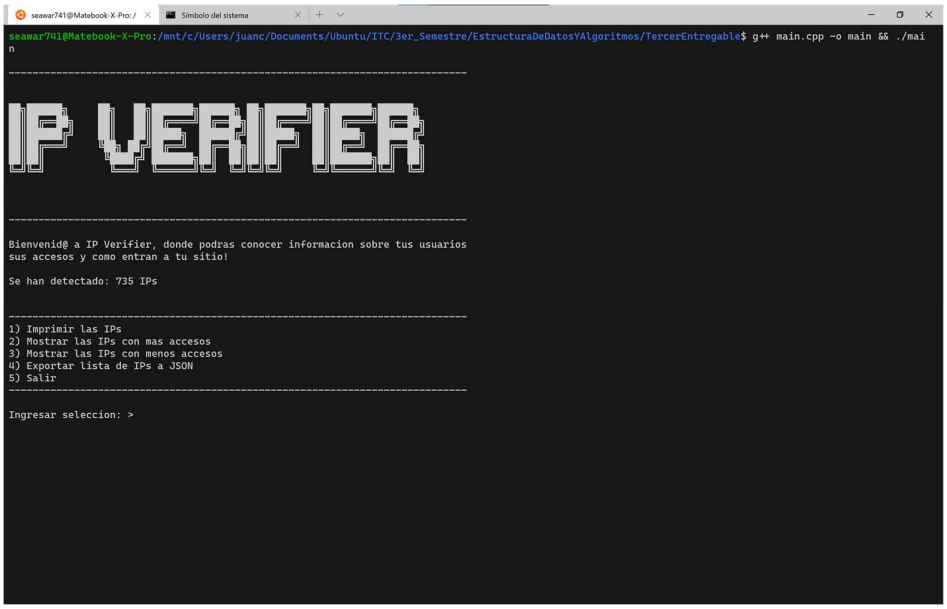


**Imagen 1.0:** Captura de pantalla del código en funcionamiento

La ventaja principal del código es que este permite no solo conocer que IPs están teniendo problemáticas con el login, sino que realmente permite evaluar la integridad de las mismas al ver que errores están causando que el servidor refute la conexión. Por otro lado, también se observa que el código almacena los puertos usados y la frecuencia de los mismos, lo cual es realmente importante al momento de comprender qué tipo de conexión esta buscándose realizar al server.

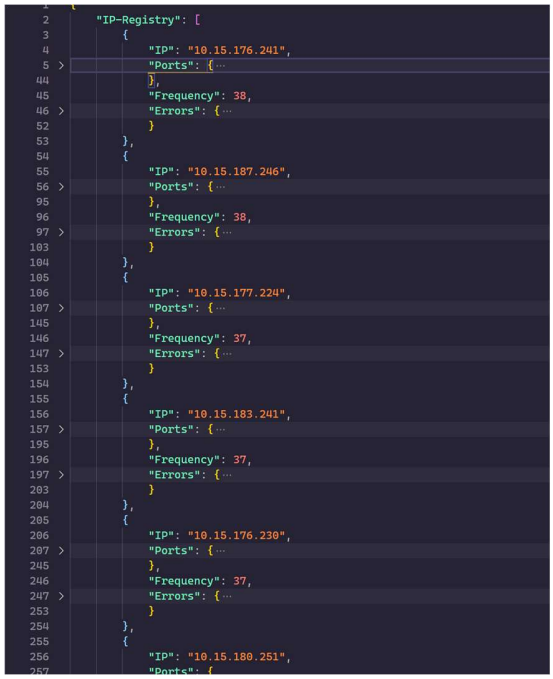


**Imagen 2.0:** Captura de pantalla del JSON generado

**Implementación en casos prácticos**

Los heaps son una implementación de una priority queue, esto suele utilizarse en casos donde hay un parámetro de priorización para la formulación y colocación de nodos. El problema de este tipo de estructuras de datos es que se pueden volver desbalanceadas, sin embargo, existen métodos de optimización que pueden solucionarlo. Grandes empresas como Google, Facebook, Bell, etc utilizan distintos métodos y estructuras de datos para poder detectar posibles ataques y conexiones infectadas. En una red privada, dependiendo de la topología de red es posible hacer un flag a aquellos nodos que debido a su comportamiento sean detectados como comprometidos.

Un ejemplo de ello es el uso nativo en sistemas Linux usado en servidores llamado iptables, el cual es un programa de utilizad de espacio de usuario que permite a un administrador configurar tablas para el firewall del núcleo de Linux y las cadenas y reglas que almacena. iptables se usa para inspeccionar, modificar, reenviar, redirigir y/o eliminar paquetes de red de IP. Por lo que el poder realizar un registro similar al dado y utilizar el código permitiría obtener IPs y puertos que deberían de llevar un tratamiento especial por parte del firewall del SO. Esto permitiría tener mucho mayor control y un nivel mayor de seguridad al momento de filtrar paquetes de IP a partir de un flag de posible infección o no.

En el escenario actual la seguridad la prioridad de todas las organizaciones por lo que uno de los pasos mas importantes es la autenticación, la cual es un mecanismo para detectar al usuario o a la entidad, garantizando su identidad. Esto puede ser riesgoso ya que teniendo la contraseña o implementando un ataque de diccionario se podría acceder a la cuenta sin ningún problema. Por ello es prudente crear sistemas de verificación de candidatos, los cuales almacenen la ip en el backend, permitiendo asi saber que lugares son frecuentados por el usuario y detectar logins de otras partes del mundo donde su cuenta se podría ver comprometida. Con ello se pueden realizar bloqueos de IP, los cuales anularían el intercambio de paquetes entre el servidor y el cliente evitando asi que continúe con los ataques de fuerza bruta, normalmente se aplican cooldowns para dar tiempo entre login y login. Sin embargo, hoy en día se pueden aplicar cambios de dirección ip a una dinámica, por lo que un método como el que se implementó en el código no seria tan efectivo en sistemas de muchos usuarios y de alto tráfico. Sin embargo, si se implementa el sistema propuesto junto con un captcha la posibilidad de que una ip maliciosa lograra entrar o hacer conexión seria mucho menor, esto por que evitan el funcionamiento de webcrawlers y scrappers, los cuales pueden actuar como bots en el intento de realizar llamadas http para acceder a las cuentas.

Los puertos también resultan de suma importancia, por lo que en la implementación se dio prioridad a almacenarlos al igual que los mensajes de error. Estos puertos permiten conocer que tipo de dispositivo u origen están tratando de

realizar conexión con el servidor. Normalmente se encuentran reservados gran cantidad de puertos y también existen convenciones para determinar que programas o paqueterías usan un rango de puertos, permitiendo asi detectar que intenta realizar la conexión. Existen puertos como el TCP, FTP, POP3, SQL, IMAP, DLS, DHCP, entre muchos otros. Generalmente están reservados del 0 al 1023, teniendo asi un rango amplio para detectar los posibles orígenes de acceso de las IPs.

La importancia de la detección de estos tipos de ataques es fundamental, principalmente ya que en ocasiones lo único que se desea es realizar ataques de DoS, los cuales hacen que no se pueda renderizar la aplicación y los servicios o procesos se ven sobresaturados causando que colapsen por una gran cantidad de solicitudes. Crear una tabla a partir de las IPs con mayores índices de probabilidad de infección permitiría reducir posibles ataques y sobre todo ayudaría a mantener el rendimiento del servidor. Por ello el inicio del proyecto podría partir perfectamente del código demostrado, ayudando a dar una idea de que, quienes y como se están conectando a los servidores. Esto a través del uso de arboles binarios o grafos los cuales permitan conocer la topología de la red y detectar aquellos nodos que por subsecuente puedan infectar otros, creando un denial of service a partir de un diccionario de ips bloqueadas.

**Referencias**


(2020). Retrieved 26 October 2020, from https://kruschecompany.com/web-app-security-101/

(2020). Retrieved 26 October 2020, from https://cloud.google.com/files/GCPDDoSprotection-04122016.pdf

Cryptography and Network Security Principles - GeeksforGeeks. (2020). Retrieved 26 October 2020, from https://www.geeksforgeeks.org/cryptography-and-network-security-principles/

Difference between graph and tree - GeeksforGeeks. (2020). Retrieved 26 October 2020, from https://bit.ly/3dXCZcP

iptables (Español) - ArchWiki. (2020). Retrieved 26 October 2020, from https://wiki.archlinux.org/index.php/Iptables_(Espa%C3%B1ol)

List of Well-Known TCP Port Numbers - Webopedia Reference. (2020). Retrieved 26 October 2020, from https://www.webopedia.com/quick_ref/portnumbers.asp#:~:text=Port%20numbers%20range%20from%200,process%20as%20its%20contact%20port.

Network Security - GeeksforGeeks. (2020). Retrieved 26 October 2020, from https://www.geeksforgeeks.org/network-security/?ref=rp