OBJETIVOS (VISTA RÁPIDA)

El Proyecto se inicia debido a que se precisa de una Herramienta que controle una Red. Pero...

¿A qué nos refiererimos con controlar?

Básicamente la capacidad de poder detectar a cada equipo que se conecta a la Red, vía Wi-Fi o Ethernet, y realizar una serie de acciones:

1. Bloquear o mantener sin conexión a dicho equipo nuevo en la Red.

2. Notificar al Administrador de la Red, usando el servicio de mensajería Telegram, de la conexión a la Red de un nuevo equipo.

3. Solicitar una acción sobre dicho cliente:

- Verificar (aceptar a dicho usuario en la red).
- Denegar (impedir el acceso a Internet a ese cliente o Intruso).
- Cazar (recopilar información del Intruso que se ha conectado).

4. Obtener la respuesta del Administrador, procesarla y ejecutarla.

El funcionamiento en esta primera version (1.0) es muy sencillo, como podemos ver.

El programa esta desarrollado en PHP, la razón de esta elección se reduce a la facilidad que nos aporta dicho lenguaje a la hora de trasladar esas ideas o procedimientos a nuestro código fuente.

UN POCO MÁS ALLÁ (EXPLICACIÓN DETALLADA)

Ahora pasaremos a explicar de forma detallada el funcionamiento de dicho programa, haciendo énfasis en los conceptos de los que se hacen uso además de las herramientas utilizadas.

Antes de nada, echemos un vistazo a los sources o programas que nos encontramos, ya que son 4, por tanto se explicará el funcionamiento de cada uno y cual de ellos se ha de ejecutar o no. Tenemos:

- ***TheHunter.php*** : es uno de los programas principales. En concreto, es el programa que se va a encargar de mapear nuestra red, detectar cuando se ha conectado un cliente e informar. Se explicará posteriormente con mayor detalle.
- ***TheHunter_Bot.php*** : es el otro programa principal. Este realiza una función vital, que es la de recibir los mensajes del Administrador y generar una respuesta a estos; todo mediante Telegram. Se explicará posteriormente con más detalle.
- ***TheHunter_Intruder.php*** : este programa contiene la Clase 'TheHunter' , donde figuran todas, o la gran parte de las funciones que se lanzarán por consola para tanto la detección de clientes y análisis de estos, como de datos de la Red, métodos de Bloqueo y de Desbloqueo.
- ***TheHunter_Querys.php*** : este programa contiene la Clase 'Hquery', donde encontramos todas y cada una de las funciones que establecen una comunicación con la Base de Datos, tanto consultas de extracción de Datos como de Insercción de Datos.

Como tenemos dos programas principales, dividiré la explicación en dos bloques:

THEHUNTER.php

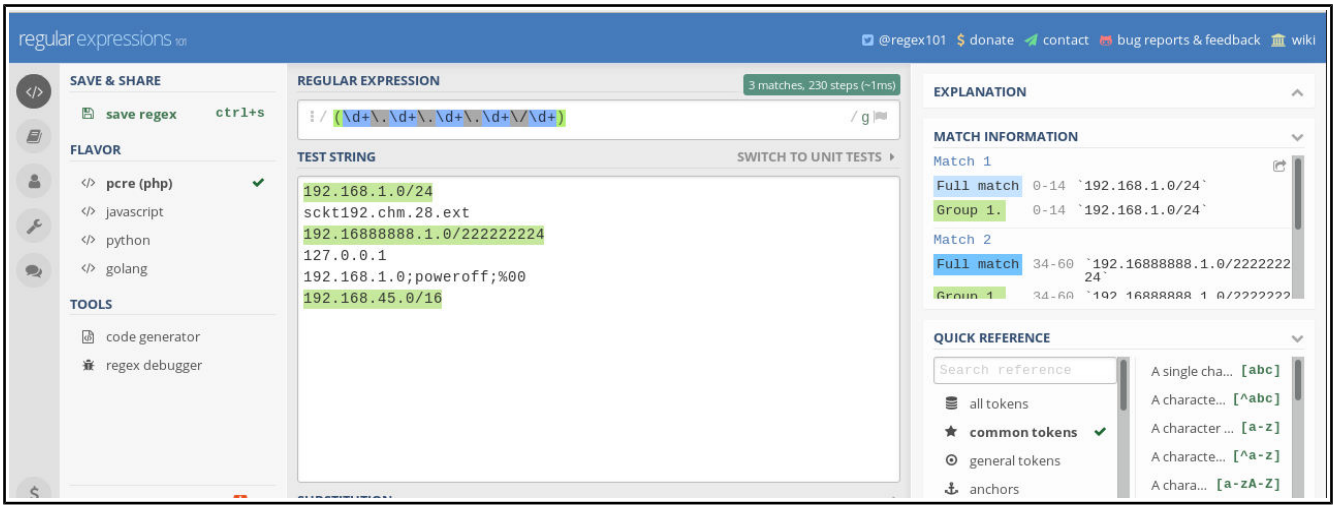
Recordamos que este programa se encargaba del bloque de detección y notificación de nuevos clientes en la Red, para ello sigue estos pasos:

1. Determina el Segmento de Red: Porque para escanear la red, debemos saber que segmento de red tenemos que escanear, para ello hemos utilizado el comando "ip route show default" que es una combinacion de parametros del comando ip que sirve para 'manipular o mostrar el enrutamiento, dispositivos, politicas de enrutamiento y tuneles'. Mediante la anterior combinación obtenemos información de la tabla de enrutamiento de entrada que hay por defecto, que nos vuelca la dirección IP de nuestro gateway o puerta de enlace, seguido del segmento de red en el que estamos.

```
[         TheHunter]$ ip route show default
default via 192.168.1.1 dev wlp2s0 proto dhcp metric 600
192.168.1.0/24 dev wlp2s0 proto kernel scope link src 192.168.1.9 metric 600
[         TheHunter]$
[         TheHunter]$ ip route show default | grep default
default via 192.168.1.1 dev wlp2s0 proto dhcp metric 600
[         TheHunter]$
[         TheHunter]$ ip route show default | awk {'print $0'}
default via 192.168.1.1 dev wlp2s0 proto dhcp metric 600
192.168.1.0/24 dev wlp2s0 proto kernel scope link src 192.168.1.9 metric 600
[         TheHunter]$ ip route show default | awk {'print $1'}
default
192.168.1.0/24
[         TheHunter]$ ip route show default | awk {'print $2'}
via
dev
[         TheHunter]$ ip route show default | awk {'print $3'}
192.168.1.1
wlp2s0
[         TheHunter]$ ip route show default | grep kernel | awk {'print $1'}
192.168.1.0/24
```

Pero, como sólo necesitamos el segmento de red, utilizaremos ‘grep’ que es un programa que nos permite mostrar tan sólo las lineas que coincide con un patrón, en este caso como nuestro segmento de red se posiciona en la segunda linea, filtrando por ‘kernel’ nos mostrará tan sólo la segunda linea, entonces finalmente, queremos obtener sólo la primera palabra de esta línea, para ello usaremos ‘awk’ un programa que nos permitirá imprimir en base a un patron, cualquiera de los campos de una linea, en este caso como queremos el primer campo (o palabra) usaremos awk {‘print $1’}, imprimiendo el primer parametro posicional, que en este caso es nuestro segmento de red.

2. Después de esto, nos permitimos hacer énfasis en una cosa muy importante, como se va a procesar esta dirección del segmento de red y se ejecutará un comando mediante “shell_exec” que es una función en PHP que te permite ejecutar comandos y retornar el stdout a una variable, nos tenemos que asegurar que esos datos que le pasamos al comando, no estan alterados y podrían costarnos la ejecuciónd de un comando, bajo permisos de superusuario (posteriormente explicaremos esto) en la terminal. Por ello utilizaremos una expresión regular para ver si lo que nos devuelve el segmento de red se trata de una dirección IP con un formato esperado. Para ello con la función ‘preg_match’ nos aseguraremos que nos llega una dirección IP, podemos comprobar dicha expresión regular en la página regex101.



Una vez hemos validado el segmento de red, procedemos a formar nuestro comando que quedará:

“sudo nmap -sP -PI 192.168.1.0/24 | grep 'Nmap scan report for\|MAC' “

La herramienta nmap nos permite hacer una infinidad de cosas, definir la quizás nos llevaría mucho tiempo y con la posibilidad de dejarnos muchas cosas. Para este caso nosotros la utilizaremos para que haga un escaneo de todo el segmento de red en busca de equipos, ahora veamos los parametros:

-sP : realiza un escaneo del tipo “ping scan” para comprobar sólo que los equipos estan vivos, es decir, que responden al ping.

-PI : este parámetro especifica que se manden ICMP Echo Request para recibir ICMP Echo Reply y confirmar que el equipo esta activo, es decir que no obtienes ningún código de error del tipo 3.

Con ‘preg_replace’ le especificamos que reemplace todos los grupos que coincidan con un patron dado, en este caso, como tenemos un salto de linea en la concatenación del comando, ya que el anterior shell_exec nos retornó en la variable un salto de línea junto con el segmento de red. Y junto a ‘trim’, conseguimos eliminar los espacios que preceden y suceden al string o segmento de red.

3. El escaneo nos tiene que devolver los equipos que estan conectados a la red, entonces con la función preg_match_all, filtraremos y extraeremos sólo los datos que nos interesan, esto son:

La IP, MAC y Marca (de la tarjeta de red) del Cliente.

Lo primero que descartaremos es la IP del router, ya que no vamos a bloquear a nuestro propio router. Después de eso, iremos validando cada Dirección IP extraida del escaneo, con los datos de las 3 tablas que tenemos en la Base de Datos (Friends, Blocks e Intruders).

¿Por qué es necesario hacer esta comprobación?

Pues porque una vez que hagamos distintos escaneos, nos encontraremos con mismos clientes, que ya estarán o Bloqueados o Verificados, por tanto, hacemos una primera comprobación a ver si se trata de un nuevo cliente.

4. Si se trata de un nuevo cliente, se cumplen una serie de acciones:

i. Se ejecuta el correspondiente ARP Spoofing que se trata de enviar paquetes ARP (generalmente ARP Request) diciendole al cliente que tu MAC corresponde a la de su puerta de enlace, es decir, que tu eres el router, de manera que esos paquetes ARP se quedan en las tablas caché del usuario, y conseguir envenenarlas (arp poisoning) o directamente falsear las autenticas. Por consiguiente, todo paquete que el cliente envie, va a ir dirigido a ti (falso router).

ii. Validaremos las dos IP's que nos llegan (del Router y del Cliente). Mediante 'nohup' se nos permite correr un comando y redirigir el STDOUT y STDERR a un fichero 'nohup.out'. O basicamente, nos permite ejecutar un comando y que se mantenga en ejecución incluso una vez hayas salido de la terminal. Con "> /dev/null 2>&1" le decimos que modifique la tabla de descriptores de esta forma: el 'stdout' irá a /dev/null y el stderr irá a stdout, mientras que con echo $! nos arroja el PID de nuestro último proceso enviado a background o a segundo plano. Pero… ¿Para qué necesitamos todo esto? Pues porque la salida del arpspoofing no manda por stdout un churro de que no nos interesa, nosotros solo queremos saber el PID del proceso del arpspoofing para poderlo parar o matar (kill) si el administrador verifica al cliente. Por tanto el comando anterior nos arroja únicamente el PID del arpspoofing ejecutado.

iii. Entonces, almacenamos la IP, MAC, Marca (de la tarjeta de Red) y PID (del arpspoofing) del cliente en la Base de Datos, en la Tabla 'Blocks', donde permanecerá hasta nueva orden (a partir de entonces, el cliente notará que no puede navegar, ya que todo el trafico esta mandandolo hacia nosotros y nosotros no lo estamos reenviando a ninguna parte).

iv. Por último, enviaremos dos mensajes al administrador de la Red: 1.) La notificación conforme se ha detectado un nuevo cliente, con sus datos. 2.) Un teclado con las 3 opciones posibles (verificar,denegar o cazar), es decir, un InlineKeyboardMarkup, que se codifica en un json y se manda mediante la petición a la API "sendMessage" (con el Teclado incluido).

Y así se mantendrá escaneando la red indefinidamente, notificando de cualquier acceso por parte de un nuevo cliente a la red, y tomando las medidas que el administrador determine. Los escaneos del segmento de red, tardan, según mis cálculos, una media de unos 4 o 5 segundos.

THEHUNTER_BOT.php

Un bot, es básicamente, un programa que realiza una serie de tareas de forma automática. Mandará y recibirá mensajes a traves de la API de Telegram. Pero…

¿Qué es la API de Telegram?

Tan sólo es una web nos permite obtener los mensajes que le han llegado al bot, al igual que mandar nuevos mensajes desde el bot haciendo una petición a la web junto con una serie de datos.

https://api.telegram.org/bot + (TOKEN) + ("/getUpdates") → nos devuelve un json con las actualizaciones, es decir, con los mensajes que ha recibido nuestro bot.

https://api.telegram.org/bot + (TOKEN) + ("/sendMessage") →nos permite enviar [junto con los parametros necesarios, el chat_id y el texto a enviar] un mensaje a un usuario desde el bot.

Teniendo claro esto, simplemente decir, que la función principal del bot, que es "run()" hace precisamente esto, va obteniendo actualizaciones de la API de Telegram, y comprobando el "update_id" que es el identificador del úlitmo mensaje recibido, sabe si le ha llegado un nuevo mensaje.

¿Y qué pasa si recibe un nuevo mensaje?

Pues hay 9 posibilidades, ya que tenemos implementados 8 comandos a los que el bot responderá:

I. */start : es el comando inicial, todo usuario que le habla a un bot por Telegram, que recuerdo que un bot se muestra como un usuario normal en el chat (lo unico que lo diferencia es que telegram te avisa que es un bot), le envía siempre el comando /start, y pues nuestro bot saluda.*

II. */help : este comando junto con el anterior suelen ser los de base para un bot, en este caso este nos muestra los comandos posibles a utilizar con una breve descripción.*

III. */whitelist : el bot muestra los datos de la Tabla "Friends" de la Base de Datos, en la que figuran los usuarios que el administrador de red ha verificado, es decir, los amigos.*

IV. */blocked : el bot muestra los datos de la Tabla "Blocks" de la Base de Datos, que es la tabla de Lista de Espera de Verficación, es decir, los clientes que no son intrusos pero no se han verificado.*

V. */blacklist : el bot muestra los datos de la Tabla "Intruders" de la Base de Datos, en esta tabla se encuentran todos los clientes que se ha denegado el acceso, siguen bajo arp spoofing.*

VI. */verificar <IP> : es uno de los 3 comandos completos, pasandole una dirección IP de un cliente que esta a espera de Verificación o Bloqueado, se realiza lo siguiente:*

1.) Se obtiene el PID de su arp spoofing (para matar el proceso posteriormente).
2.) Se mueve el cliente a la Tabla 'Friends' de verificados y se elimina de 'Blocks'.
3.) Detenemos el ARP Spoofing, matando el proceso identificado con PID extraido. Utilizamos el comando 'kill -9 <PID>', que manda una señal al proceso del tipo SIGKILL que hará que finalice el proceso inmediatamente. Y posteriormente notificamos al administrador de la Red.

VII. /denegar <IP> : mueve al cliente de la Tabla 'Blocks' a la tabla 'Intruders', es decir, se reconoce al cliente como intruso, y se le mantiene el bloqueo de conexión. De igual forma se notifica al administrador de Red.

VIII. /cazar <IP> : similar al anterior comando, la única diferencia es que lanza un escaneo personalizado sobre el intruso, desvelando tanto el Sistema Operativo, como los puertos que tiene abiertos, introduce dichos datos en la Base de Datos y los notifica al administrador. Dichos datos pueden aportar al administrador una idea más clara sobre el Intruso que se ha colado en la Red.

Y si se envia un mensaje que no figura entre los comandos anteriores, simplemente se responde con un mensaje de error por defecto, junto con el comando de ayuda.

INCONVENIENTES

Esta metodología presenta una serie de convenientes destacados que pueden tener mucha relevancia para decidir modificar según que aspectos de la implementación. Iremos por pasos:

- Método GetUpdates: hablamos de la forma en la que el bot obtiene las actualizaciones de la API de Telegram, que consiste basicamente en realizar peticiones continuamente, lo cual eleva el tráfico de red y te hace perder milisegundos de tiempo entre peticion y peticion, ya que has de mandar la petición, decodificar el json, comprobar los elementos del array que te dicen si hay un nuevo mensaje, y en el caso negativo, se frena el flujo en 2 segundos que reduce el número de peticiones a la web, pero aún así es un método muy agresivo. En su lugar se podría utilizar el método de Webhooks.
- Escaneo con Nmap: como bien decíamos antes, existe la posibilidad de que un equipo no responda a un ping, por ejemplo usando iptables para dropear esos paquetes o sino directamente al kernel pasandole 'kernel icmp_echo_ignore_all' (en linux). Y otros métodos de evasión.
- ArpSpoofing: hemos hablado anteriormente en la posibilidad de saturar la red, bueno, es dificil que suceda en una red pequeña utilizando este método, pero si incrementa el número de bloqueados, entonces, hay que pensar que estar mandando continuamente paquetes ARP falsos

a todos esos clientes, puede afectar al tráfico de red, por tanto es un punto que tener en cuenta. Pero ya no sólo el problema de saturación, antes hablabamos de caché arp, si las tablas arp son estáticas, adiós a este método.

- Dos Bloques, dos ejecuciones: evidentemente, por un lado tenemos al TheHunter.php y al TheHunter_Bot.php, con dos finalidades, esto dificulta la tarea, ya que, apesar de que los puedes dejar en segundo plano, o directamente con nohup, para cerrar las terminales, iría todo más estructurado en un sólo codigo, con 2 hilos por ejemplo, o dos procesos.

Y muchos otros más, no contemplados o quizás de menor grado, pero simpre surgen inconvenientes, de igual forma, siempre pueden repararse, pero eso conlleva tiempo.

FUTURAS MEJORAS EN VERSIONES POSTERIORES

Nos situamos en la versión 1.0, versión inicial, en las posteriores se pueden desarrollar una serie de implementaciones interesantes que comentamos:

1. Migrar código a C++, para utilizar forks, y obtener así dos procesos, comunicados mediante una tubería. O sino, la posibilidad que cabe es usar Threads en PHP, si se desea mantener el código.
2. Estudiar sustitutos del ARP Spoofing, como por ejemplo DHCP Spoofing, tanto por tema de rendimiento, como de mayor control y menor saturación. Aunque este método también presenta sus inconvenientes.
3. Implementación de tecnicas de denegación de servicio o desautenticación a los intrusos. De igual forma, el “/cazar” pretendía encaminar esto, un análisis dirigido a dicho cliente, y aprovechar para contraatacar, todo desde un punto de vista defensivo. Por ello también se podrían desarrollar métodos de análisis de vulnerabilidades incluso de Ingenieria Social al Intruso (contemplando los paquetes que manda, es decir, un cazito de Iptables).

Y seguro que se pueden desarrollar muchas otras mejoras, para evitar por ejemplo utilizar sudo, o similar. Pero eso ya lo dejamos para posteriores versiones.

BIBLIOGRAFÍA:

http://rm-rf.es/como-usar-el-comando-ip-en-linux-ejemplos-vs-ifconfig/

https://unix.stackexchange.com/questions/374581/difference-between-1-1-and-1-in-awk

https://www.cyberciti.biz/faq/howto-pingscan-icmp-ip-network-scanning/

http://php.net/manual/es/function.trim.php

http://php.net/manual/es/book.curl.php

https://es.wikipedia.org/wiki/ARP_Spoofing

http://rm-rf.es/nohup-mantiene-ejecucion-comando-pese-salir-terminal/

https://unix.stackexchange.com/questions/119648/redirecting-to-dev-null

https://stackoverflow.com/questions/11369104/linux-what-does-echo-in-linux

https://core.telegram.org/bots/api#inlinekeyboardmarkup

https://core.telegram.org/bots/api

https://es.wikipedia.org/wiki/Bot

https://www.reddit.com/r/linux/comments/1xvr25/linux_tip_dont_use_kill_9/

http://php.net/manual/es/function.sleep.php

https://core.telegram.org/bots/webhooks

https://hackiwis.es/cazadores-de-trolls/

*Apuntes de Redes de Computadores.

*Apuntes de Sistemas Operativos.

*Ayuda de los compañeros de BitUp.

*Ayuda de los compañeros del grupo de Telegram Fwhibbit.

By @Secury