View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by El Servicio de Difusión de la Creación Intelectual

Automatizando el descubrimiento de portales de autenticación y evaluación de la seguridad mediante ataques de fuerza bruta en el marco de una auditoría de seguridad

Gastón Traberg, Lía Molinari, Paula Venosa, Nicolás Macia, and Einar Lanfranco

LINTI, Facultad de Informática,
Universidad Nacional de La Plata,Buenos Aires, Argentina
{gtraberg}@cert.unlp.edu.ar
{lmolinari,pvenosa,nmacia,einar}@info.unlp.edu.ar
http://www.linti.unlp.edu.ar

Abstract. Este trabajo, derivado de la tesina de grado del alumno Gastón Traberg, plantea el desarrollo de la aplicación Swarming[1], la cual surge como una herramienta que encara de manera novedosa la automatización de algunas tareas que forman parte de la auditoría de seguridad de un servicio, mediante el descubrimiento de portales de autenticación y la obtención de credenciales de acceso mediante ataques de fuerza bruta, contemplando en este proceso los vacíos dejados por otras aplicaciones e incorporando tecnologías para las que existe escaso o ningún soporte.

Keywords: seguridad, auditoría, fuerza bruta, swarming, HTTP

1 Motivación

La utilización de ataques de fuerza bruta en el ámbito de las auditorías de sistemas de información, habitualmente presentan dos situaciones adversas:

- En cuanto a la modalidad, pues se realizan de una manera que desaprovecha gran parte de la información expuesta por los servicios que la red posee y que es relevante acerca de revelar cuáles son las conveniencias en la realización del ataque.
- En cuanto a la dificultad, ya que que se vuelven muy complejas en el momento de la configuración, sobre todo en servicios que ofrecen recursos de gran canttidad de alternativas en su utilización, como es el caso de HTTP.

En la práctica ocurre que la fuerza bruta durante una auditoría es poco practicada o se efectúa sin aprovechar su potencial, ya sea por cuestiones de tiempo o por las situaciones antes citadas, dejando así fuera del análisis importantes vulnerabilidades en los servicios que son objeto de auditoría.

2 Estado del arte

La Fuerza Bruta[2] es una modalidad de ataque mediante la cual se intenta obtener acceso no autorizado realizando intentos reiterados y sistemáticos de login sobre un servicio de red, a través de la utilización de credenciales potencialmente válidas. Las credenciales utilizadas durante la realización de este tipo de ataques, suelen provenir de conjuntos de credenciales que existen por defecto para ciertos dispositivos o aplicaciones, de credenciales comúnmente utilizadas por los usuarios o de credenciales que fueron comprometidas previamente en otros ataques[3]. En el contexto de las auditorías de seguridad, mediante la fuerza bruta se detectan todos o la mayoría de los casos en los cuales las credenciales utilizadas para el acceso a un servicio presentan alguna debilidad, las cuales potencialmente permitirían a un atacante acceder de forma no autorizada al sistema de información que está siendo evaluado.

3 Introducción

Entre las tres herramientas más destacadas que se utilizan en la comunidad de la seguridad informática para realizar ataques de fuerza bruta podemos mencionar: Hydra[4], Medusa[5] y Ncrack[6].

Sobre estas opciones es que se construyó el análisis que expone ventajas y limitaciones de cada una, constituyendo la base del desarrollo planteado.

Hydra es una de las aplicaciones de fuerza bruta más antiguas que existen en el contexto de la seguridad siendo desarrollada en sus inicios por el Alemán van Hauser en 2001. Hydra es Software Libre, y es popularmente, la primera opción al momento de hablar de herramientas de fuerza bruta sobre redes. Esta aplicación es activamente mantenida, habiendo sido actualizada la rama estable por última vez el 12 Mayo del 2014, mientras continúa su desarrollo utilizando github[7] como repositorio de código fuente. Entre las ventajas que se pueden destacar de Hydra se destaca:

- el gran número de protocolos que soporta para atacar
- el número de plataformas sobre las que se indica que funciona.

Por otro lado Hydra posee algunas características que le restan popularidad al momento de utilizar o de extender en funcionalidad, entre las que se destacan:

- Suele suceder que la aplicación se corrompe durante su ejecución u omite resultados producto de los errores que se generan. Y si bien algunos de estos problemas se fueron solucionando con el tiempo, otros aún persisten.
- Al estar implementada en lenguaje C y con un estilo de desarrollo muy particular de los autores del código, es difícil tener un entendimiento claro de lo que ocurre dentro de la aplicación, dificultando mucho la realización de extensiones por parte de gente que no forme parte del equipo principal de desarrolladores del proyecto.

Realizando estas mismas observaciones, es que miembros del grupo Foofus.net se avocaron al desarrollo de Medusa, subsanando de en este nuevo producto la mayoría de los problemas que Hydra presentaba. Comparándolos:

- Medusa posee un código fuente mucho más organizado y prolijo, además de otras características que lo hacen una interesante opción,
- Pero a diferencia de Hydra, Medusa tiene soporte sobre un número algo menor de protocolos .

Como tercera opción popular existe nCrack que es un proyecto nacido en el "Google Summer of Code" de 2009 pero que aún no es considerado como un programa completo ni terminado ni siquiera por los propios autores. En conclusión es válido decir que no es una opción muy competitiva contra las dos primeras:

- en principio porque no ofrece nuevas funcionalidades
- y en segundo lugar su desempeño es inferior con respecto a Hydra y a Medusa.

Concluyendo podemos decir que estas tres herramientas en conjunto poseen soporte para una gran cantidad de protocolos, buena velocidad en la realización de ataques de fuerza bruta y relativa simplicidad de uso[8][9].

Sin embargo, más allá de estas características positivas, lo que hace que estas herramientas sean la primer opción al momento de hablar de fuerza bruta sobre redes, es la inexistencia de alternativas completas. Esto se comprueba mediante la realización de simples casos de uso de estas aplicaciones sobre redes y servicios comúnmente encontrados hoy en día en cualquier red, quedando rápidamente en evidencia los muchos problemas que éstas presentan a lo largo de su utilización. Pudiendo mencionar:

- De existir un Firewall como iptables[10] o un IPS como fail2ban[11], los servicios solo se pueden acceder de a intervalos de tiempo regulares o un cierto número de veces por cantidad de tiempo. Esto genera situaciones que no siempre son bien manejadas por las herramientas, provocando que estas ignoren algunas credenciales o que las mismas sean rechazadas, dejando en desconocimiento si dichas credenciales son válidas o no.
- Existen protocolos como HTTP que presentan una gran complejidad en los procesos de login, ya que por ejemplo suelen utilizar para el manejo de dichos procesos Cookies de sesión o Tokens anti CSRF[12]. Estos mecanismos que no se encuentran considerados y soportados en forma completa en las en las herramientas antes mencionadas, sino que, o se lo hace de manera parcial o directamente ni se los considera. Esta ausencia de soporte lográ que para ellas HTTP sea un ejemplo de protocolo que queda excluido, aunque sea extremadamente utilizado hoy en día.
- Siguiendo con HTTP, a diferencia de otros servicios, en la actualidad este tiene un rol extremadamente destacado dentro de una red, en muchos de los aspectos posibles. Se ha vuelto un servicio extremadamente complejo, y aparejado a esa complejidad creció el número de potenciales problemas de seguridad existentes. Todas las herramientas mencionadas se limitan a

auditar sólo lo explícitamente indicado por su ejecutor, generando que la auditoría de un servicio con estas características sea una tarea imposible, debido a la enorme cantidad de contenido a analizar antes de poder generar una ejecución válida de las mismas.
- La velocidad con la que se realiza un número de intentos de login no es relevante al momento de un ataque. Este objetivo parece ser el buscado en principio por las aplicaciones listadas, no siendo el caso de la propuesta de Swarming. Si algo deja como experiencia la auditoría de redes y sus servicios es que en la mayoría de los casos, las credenciales que uno busca están al alcance de la mano o suelen ser de fácil inferencia, quedando la velocidad de ataque rezagada a un segundo plano ya que el problema radica en la obtención de las credenciales y no en la velocidad con la que se las utiliza.

Si bien se mencionan las expuestas anteriormente como las mas destacables, existen infinidad de características más que hacen de estas tres herramientas algo poco práctico al momento de hablar de procesos de fuerza bruta, más cuando se trata de auditorías de redes grandes.

4 Swarming

4.1 Los objetivos planteados

Partiendo de todos los problemas anteriormente analizados como así también de experiencias e ideas propias y sugeridas, es que se elaboró una lista de requerimientos que sirvieron como base para el diseño de la aplicación solución, entre los cuales se destacan:

- Realizar procesos de fuerza bruta, también conocidos como Cracking.
- Realizar un recorrido automático del servicio auditado, proceso conocido como Crawling.
- Crecer dinámicamente en el número de procesos que realizan los trabajos.
- Reconocer aplicaciones, dispositivos y cualquier característica que signifique un ahorro de trabajo en la auditoría.
- Reutilizar lo descubierto, iniciando de manera automática procesos de Crawling sobre recursos a los que se tuvo acceso, repitiendo el ciclo todas las veces que sea posible.
- Brindar una interfaz de usuario simple, basada en una API que permita la interacción de otras herramienta con la aplicación de manera sencilla.
- Seguir un modelo en el cual cada módulo encargado de procesar un protocolo en particular, funcione de la manera más aislada posible, permitiendo la carga y descarga de los mismos además de una fácil y flexible implementación.

De esta manera se establece un escenario que a futuro facilitará la tarea de agregar nuevas funciones, entre otras, los protocolos a soportar.

Si bien es imprescindible la implementación de la mayor cantidad de protocolos posible para un aprovechamiento máximo de la capacidad de la aplicación, en la primer instancia del desarrollo se apuntó a HTTP como primer protocolo a

contemplar. Esta decisión se justifica porque HTTP es uno de los protocolos con peor soporte por parte de las aplicaciones antes mencionadas, además de ser el protocolo que más problemas puede presentar debido a su complejidad y al gran número de diferentes situaciones existentes en su utilización.

4.2 Características del desarrollo

Swarming nace como una aplicación cuyo objetivo principal es el desarrollo automático de todas las actividades posibles en torno a los procesos de fuerza bruta, buscando con esto aprovechar al máximo toda la información obtenida del contexto, sin que se requiera intervención humana.

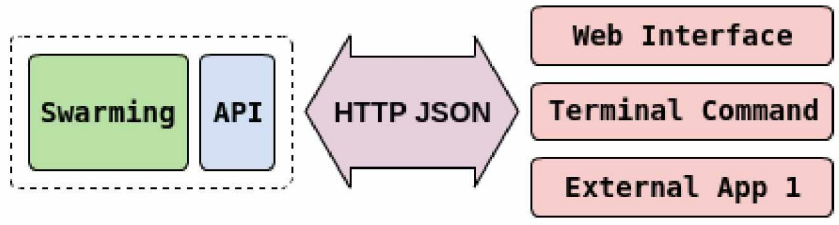

Fig. 1. Arquitectura de SWARMING

Para lograr esto, se siguió un modelo que presenta una única interfaz a través de la cual no solo el usuario controla la aplicación, sino que también permite que otras aplicaciones puedan acceder a los datos y controles, logrando de esta manera una fácil extensión en el uso del Swarming por otros programas. La figura 1 muestra una representación general de cómo el único medio de acceso, ya sea desde una interfaz de usuario o desde una aplicación externa, es mediante la API del programa, la cual es accedida a través del envío de mensajes en formato JSON[13].

Para la implementación, si bien existe un espectro interesante de lenguajes, se optó por la utilización del lenguaje de programación Python[14] en su versión 3, el cual presenta muchas cualidades dentro de las que podemos destacar:

- Lenguaje Interpretado. Esta característica es una gran ventaja al momento de desarrollar, ya que las pruebas pueden realizarse inmediatamente después que los cambios fueron realizados. Si bien el hecho de que sea un lenguaje interpretado no es óptimo con respecto a la performance, existe en Python la posibilidad de realizar módulos en algún lenguaje compilado (C++ por ejemplo), los cuales pueden ser importados y utilizados desde Python, característica que será aprovechada para dar soporte a protocolos como MySQL, SSH2, y otros.
- Soporte nativo de Unicode. Hoy en día, la capacidad de soportar la codificación Unicode en las comunicaciones es más un requerimiento que una opción. Es

por esto que se optó por Python 3 para la implementación del Swarming, ya que Python en sus versiones anteriores, como la 2.7, no soporta Unicode de manera simple.
– Gran Número de Librerías existentes que se pueden utilizar. Algo que siempre caracterizó a Python es el hecho de que posee un espectro muy interesante de librerías para distintas actividades, desde ORMs para el modelado de bases de datos siguiendo una orientación a objetos, pasando por Frameworks para el desarrollo de interfaces Web y utilización de JSON, hasta módulos para el manejo de protocolos como HTTP de forma muy simple pero no por eso menos potente.

5 Experiencia realizada

Para evaluar la implementación actual de SWARMING, se configuró un servidor WEB con el objeto de simular ser el portal de una organización real, accesible vía: http://www.miportal.unlp.edu.ar/.

Existe en el portal una sección http://www.miportal.unlp.edu.ar/admin/ cuyo ingreso está restringido con una autenticación tipo HTTP Basic, y que al autenticarse exitosamente permite acceder a un PHPMyAdmin, el cuál posee una autenticación propia que utiliza un formulario HTTP.

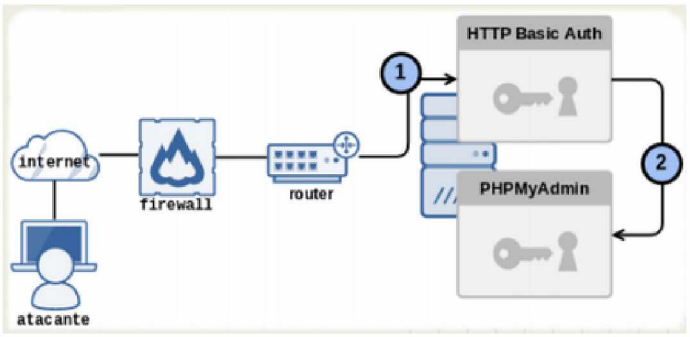

Fig. 2. Entorno de prueba

Se ejecutó la herramienta especificando la url del sitio web y los diccionarios de usuarios y contraseñas a utilizar a fin de:

– Descubrir posibles portales de autenticación como ser: HTTP Basic, HTTP Digest, Formularios HTML u otro.
– Ejecutar ataques de fuerza bruta sobre los mismos para una posterior evaluación de la información encontrada en los mismo.

Como resultado de la etapa (1), SWARMING encontró el portal *http://www.miportal.unlp.edu.ar/admin/* el cual estaba protegido con autenticación HTTP tipo Basic y creó una "tarea"

para la ejecución del ataque de fuerza bruta utilizando los diccionarios previamente cargados (etapa 2).

Finalizada la etapa (2), en caso de haber logrado el acceso SWARMING se encarga de iniciar nuevamente el ciclo de ejecución (descubrimiento/ataque) dentro de la sección a la que antes no se tenía acceso.

En un segundo ciclo de ejecución SWARMING detectó la aplicación WEB PHPMyAdmin, la cual restringe el acceso mediante un formulario HTML sobre el cual luego de descubierto se inicia nuevamente el ataque.

Para lograr lo anteriormente descripto, antes de la implementación de SWARMING hubiera sido necesario que el analista de seguridad realizara en forma manual:

- El descubrimiento de portales de autenticación utilizando alguna herramienta adicional que permita enumerar los recursos en la red, no sólo una herramienta de fuerza bruta, por ejemplo usando Burp/Firefox[15].
- Una vez descubierto el portal si realizar la ejecución del ataque de fuerza bruta, usando alguna de las herramientas preexistentes.

Dicho proceso el analista debería volver a iniciarlo de forma manual, luego de obtener algún nuevo acceso. En cambio SWARMING, automatiza lo anteriormente descrito utilizando unos pocos pasos de configuración los cuales implican:

- Iniciar la aplicación
- Cargar los diccionarios
- Especificar la URL

Una de las ventajas que vale la pena destacar de SWARMING es que permite interrumpir el proceso de auditoría en cualquier momento, como si se estuviera pausando, ya que luego va a continuar a partir del punto en que fue interrumpido sin necesidad de arrancar desde cero. Otra aspecto interesante es que a medida que avanza el análisis el usuario puede acceder a resultados parciales, ya sea a la información sobre portales de autenticación encontrados o accesos a los mismos logrados sin necesidad de que la herramienta finalice completamente su ejecución.

6 Conclusiones

Mediante lo experimentado se pudo observar el grado de automatización que el Swarming ofrece, no sólo en el descubrimiento de recursos a auditar, sino también retroalimentandose a partir de la generación de nuevas tareas a desarrollar sobre el servicio a partir de lo parcialmente descubierto, con el fin de lograr el máximo nivel de acceso posible.

De haberse ejecutado a través de aplicaciones como Hydra o Medusa cada etapa analizada a lo largo de la ejecución del Swarming, hubiese resultado en una repetida ejecución y análisis de resultados de forma manual, desperdiciando así todo el tiempo que la buena performance de estas herramientas podrían llegar a ofrecer al usuario.

Esta aplicación logra sus objetivos al implementar de forma completa mecanismos de autenticación HTTP, brindando un soporte robusto de formularios HTML y manejo de sesiones HTTP. Si bien la herramienta aún no presenta todas las capacidades que se pretenden, está diseñada de manera extensible permitiendo en el futuro la incorporación de nuevas características o funcionalidades.

El desarrollo de soporte de otros protocolos se propone como trabajo a futuro.

References

1. Swarming - Repositorio GIT de código del proyecto https://github.com/samelat/swarming/
2. Penetration Testing and Network Defense by Andrew Whitaker,Daniel P. Newman - Capítulo 9 - Página 209 - Brute Force Attacks - Cisco Press
3. Hacking: The Next Generation by Nitesh Dhanjani, Billy Rios & Brett Hardin - Capítulo 3 - Página 74 - Brute-Forcing Your Way In.
4. Hydra - Sitio Oficial: https://www.thc.org/thc-hydra/
5. Medusa - Sitio Oficial: http://foofus.net/goons/jmk/medusa/medusa.html
6. nCrack - Sitio Oficial: https://nmap.org/ncrack/
7. https://github.com/vanhauser-thc/thc-hydra
8. Análisis de performance comparativo - Realizado por el proyecto Hydra: https://www.thc.org/thc-hydra/network_password_cracker_comparison.html
9. Análisis de performance comparativo - Realizado por el proyecto Medusa: http://foofus.net/goons/jmk/medusa/medusa-compare.html
10. The netfilter.org project - IPTABLES http://www.netfilter.org/projects/iptables/index.html
11. Fail2ban HomePage http://www.fail2ban.org/wiki/index.php/Main_Page
12. Cross-Site Request Forgery. Web Security: A WhiteHat Perspective by Hanqing Wu y Liz Zhao. Capítulo 4, pag. 123.
13. Ajax, Data Formats, JSON. High Performance JavaScript by Nicholas C. Zakas - Capítulo 7, pag. 138.
14. Mark Lutz. Learning Python. 5ta edición.
15. BURP Proxy - https://portswigger.net/burp/proxy.html