Framework para la generación automática de logs para el entrenamiento de modelos de aprendizaje automático [1]

Mikaela Pisani

Universidad ORT Uruguay

Agosto 2021

1 Introducción

Los modelos de aprendizaje automático podrían ser de relevancia en la detección de ataques. Pero utilizar este tipo de modelos para la detección de ataques posee ciertos obstáculos.

Debido a que los datos de logs generados por los sitios generalmente poseen información privativa, resulta complejo poder obtener datos masivos de estos sitios. A su vez, el conjunto de estos logs que refieran a ataques será reducido.

Ante la falta de datos para el entrenamiento de modelos de aprendizaje automático, tanto sobre ataques a sitios, como de tráfico normal, surge la necesidad de poder generar datos que puedan utilizarse como conjunto de entrenamiento.

Entonces, se propuso la creación de una plataforma para la generación automática de logs para el entrenamiento de modelos de aprendizaje automático para la detección de ataques. Dicha plataforma genera logs de ataques y de tráfico normal.

2 Herramientas investigadas

Se investigan diferentes herramientas útiles para la generación de logs. En esta sección se describen las mismas.

2.1 Sitio vulnerable

En primer lugar, es necesario contar con un sitio al cual invocar, el cuál generará los logs correspondientes. Los siguientes proyectos de código abierto fueron analizados:

2.1.1 DVWA Damn Vulnerable Web Application (DVWA)

https://dvwa.co.uk/

Es una aplicación web PHP / MySQL vulnerable. Este sitio es utilizado como una herramienta de aprendizaje por los profesionales de seguridad, de esta manera pueden poner en prueba sus habilidades en un entorno legal.

Provee separado por diferentes botones tipos de ataques, lo que permite separar fácilmente los mismos.

Brinda la posibilidad de ser instalado en un contendor Docker. De esta manera no requiere instalar ninguna dependencia y podrá ser instalado en cualquier plataforma que soporte contenedores.

2.1.2 Juice Shop

https://owasp.org/www-project-juice-shop/

[1] Documento de trabajo.

Esta herramienta también es un sitio web vulnerable. El mismo provee varios desafíos para vencer y es utilizado en para enseñar sobre seguridad en las aplicaciones web.

A diferencia de DVWA, este sitio web se asemeja más a un sitio real, en especial un "shop" online.

Provee una imagen Docker para la instalación.

2.1.3 Dvpwa

https://github.com/anxolerd/dvpwa

Este proyecto está inspirado en DVWA. Tiene como propósito la creación de una aplicación similar al mundo real. Se utiliza docker-compose para la instalación de los contenedores, la aplicación y la base de datos.

2.2 Herramientas para generación de ataques

En esta sección se mencionan diferentes herramientas investigadas para la generación automática de logs de ataques.

2.2.1 ZaProxy

https://www.zaproxy.org/

OWASP ZAP (Zed Attack Proxy) es un escáner de seguridad para aplicaciones web. Puede ser utilizado para test de penetración en seguridad informática. Los test de penetración son justamente para detectar problemas de seguridad en el sitio web.

2.2.2 PortSwigger

https://portswigger.net/

Es una herramienta para explorar la seguridad en la red y el tráfico. La versión comunitaria solamente proporciona la ejecución de tareas manualmente.

2.2.3 Sqlmap

https://sqlmap.org/

SQLMAP es una herramienta de prueba que automatiza el proceso de inyección SQL para diferentes sistemas de administración de bases de datos.

2.2.4 Postman

https://www.postman.com/

Esta herramienta permite la creación de un conjunto de peticiones HTTP.

2.2.5 Web Application Firewall (WAF)

WAF es un firewall que aplica filtros, monitoreo y puede bloquear el tráfico HTTP hacia y desde un servicio web. Mediante la inspección del tráfico HTTP, puede evitar ataques mediante la detección de los mismos en base a determinadas reglas.

2.3 Visualización

2.3.1 Threadfix

https://threadfix.it/

Mediante esta herramienta se pueden visualizar resultados y análisis, en particular se conecta con herramientas de escaneo de seguridad, para poder visualizar las vulnerabilidades encontradas.

3 Experimentos

En esta sección se describen los experimentos realizados con las herramientas mencionadas anteriormente.

3.1 Sitio vulnerable

Se realizaron pruebas con los 3 sitios vulnerables mencionados. Se decide utilizar DVWA debido a que es más genérica y simple. Por otro lado, Juice Shop es una herramienta interesante pero está orientada a resolver desafíos por parte del atacante, por lo que se requiere mayor conocimiento para la generación de ataques. DVPWA es un repositorio creado para unas pruebas puntuales y posee falta de mantenimiento.

Más adelante se sugiere analizar la incorporación de otros sitios vulnerables para complementar la generación de los logs y poder tener más variedad en los mismos.

3.2 WAF

Se realizaron las mismas pruebas de invocaciones al sitio vulnerable configurando el WAF y sin configurarlo, para notar diferencias en los logs.

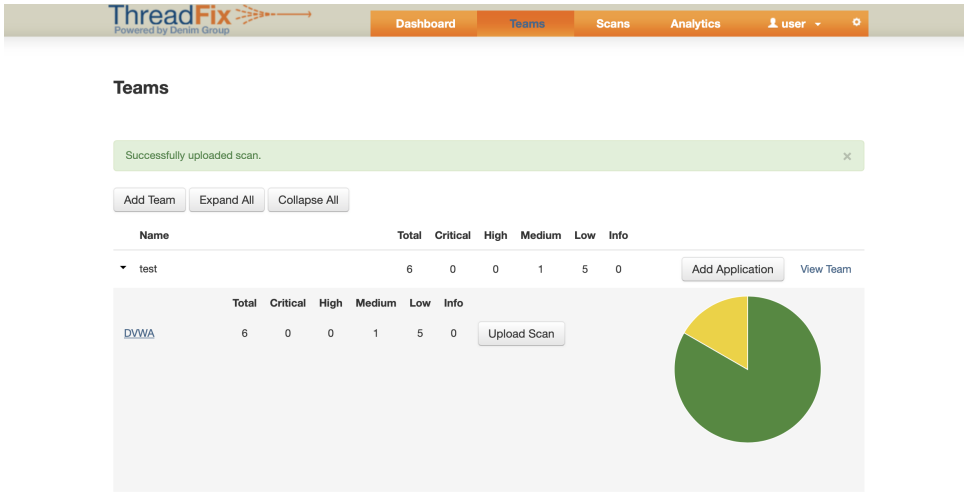
3.2 Experimento con ZaProxy

Se realizaron los siguientes experimentos con la herramienta ZaProxy.

1. Desde la interfaz de ZAP - Desktop UI: Desde la interfaz de usuario de ZAP Desktop, se pueden enviar ataques automatizados hacia el sitio vulnerable. Solamente requiere insertar la URL para atacar.

2. Desde el cliente ZAP: mediante un contenedor que posee instalado ZAP, un cliente escrito en Python proporciona una forma sencilla de escanear el sitio desde la línea de comandos.

3. Utilización de ThreadFix para visualización de reportes: el resultado de ZAP puede ser exportado en HTML. Luego, este reporte puede ser importado en ThreadFix para la visualización de datos y gráficas de forma centralizada para diferentes sitios.



3.3 Experimento inyección de ataques SQL

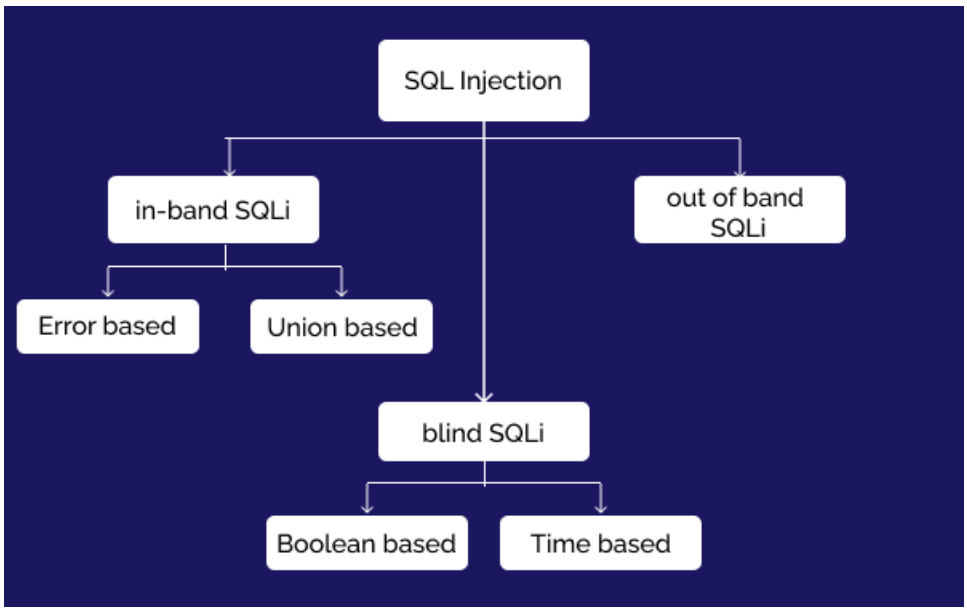
Se realizaron distintas pruebas, inyectando los ataques SQL de distintas formas, probando con o sin WAF para notar las diferencias en los logs generados.

Se prueba la inyección de ataques desde

1. Sqlmap
2. Inyectando las consultas directo desde el sitio web
3. Mediante peticiones HTTP a través de PostmanEstas 3 pruebas se realizaron con o sin WAF.

3.3.1 Tipos de ataques SQL

Hay diferentes tipos de ataques SQL. Los más comunes se pueden ver en la siguiente figura:



1. In-Band: El canal de comunicación utilizado por el atacante para enviar el ataque y recopilar los datos es el mismo. Dentro de estos ataques, puede ser basado en error o basado en unión.
    a. Error-based: el atacante provoca un error para a partir del mismo obtener información sobre la estructura de la base de datos.
    b. Union-based: utiliza la operación UNION de SQL para obtener información de la base de datos.
2. Out-Band: el atacante utiliza canales de comunicación diferentes
3. Blind SQL: El atacante envía cargas útiles de datos al servidor y observa la respuesta y el comportamiento del servidor para obtener más información sobre su estructura.
    a. Boolean: El atacante envía una consulta SQL a la base de datos y la aplicación devuelve un resultado, sobre el cual el atacante puede averiguar si el mensaje generó un resultado verdadero o falso.

b. Time-based: ante una consulta a la base de datos, la aplicación demora unos segundos para realizar la operación, por lo que el atacante puede descifrar en base al tiempo de ejecución de su acción si una consulta es verdadera o falsa.

3.3.2 Sqlamp

Mediante sqlmap se puede diferenciar entre los siguientes tipos de ataques:

Commando	Ataque
B	Boolean-based blind
E	Error-based
U	Union query-based
T	Time-based blind
Q	Inline queries

Por lo que se realizan diferentes baterías de ataques hacia distintas instancias del sitio vulnerable.

3.3.3 Experimento en Python

Se crea un script en Python que realiza peticiones HTTP que invocan a las diferentes secciones del sitio vulnerable DVWA para cada tipo de ataque. Los tipos de ataques soportados hasta el momento en este script son: Command Injection, File inclusion , XSS. Al ejecutar el script puede indicarse el tipo de ataque a ejecutar, url del sitio vulnerable y cantidad total de peticiones realizadas. De esta manera pueden los distintos tipos de ataques hacia diferentes instancias del sitio.

3.3.4 Experimento con Postman

Se prueba invocar al sitio mediante peticiones HTTP desde la herramienta Postman. Donde se pueden definir variables a reutilizar y se valida la idea de que puede ser una herramienta útil para la ejecución de ataques.

4 Generación de tráfico normal

También es necesario generar logs que refieran a tráfico normal del sitio. Estos pueden ser generados manualmente o mediante un script automático que realice invocaciones al sitio.

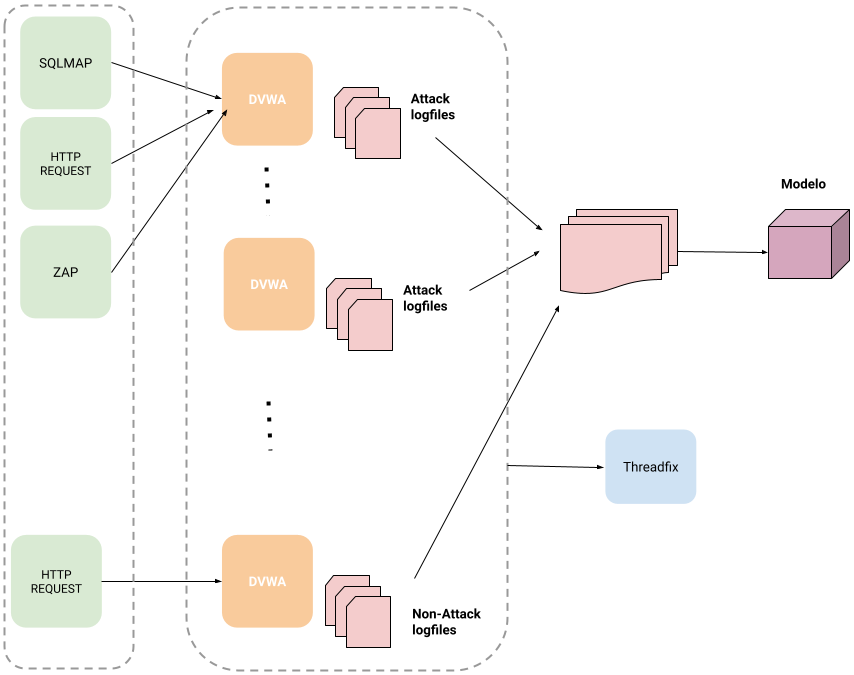
4.1 Arquitectura

Se diseña una arquitectura orientada a microservicios, donde cada componente corre en un contenedor independiente y cumple una funcionalidad especifica.

De esta forma, se logra escalabilidad tanto horizontal como vertical. A su vez, se logra independencia de la plataforma a utilizar, dado que los contenedores pueden correr en cualquier plataforma cloud o ser instalados localmente.

A continuación, se presenta un diagrama que muestra los componentes de la arquitectura.

En primer lugar, tenemos los contenedores generadores de ataques y de tráfico normal. Los mismos, realizan invocaciones hacia diferentes contenedores que corren el sitio vulnerable, uno por tipo de ataque y otro para tráfico normal. De esta forma, los logs generados en estos sitios pueden ser recopilados y utilizados para el entrenamiento de un modelo de aprendizaje automático.



Una vez que se tienen los datos recopilados, es necesario aplicar determinadas transformaciones a los datos para prepararlos para el modelo.

Se dispone de un modelo de aprendizaje automático entrenado con los datos generados que identifica entre ataque o no ataque. Este modelo es una prueba de concepto para validar el uso framework construido.

Los sitios vulnerables serán registrados en la herramienta Threadfix, para visualizar las vulnerabilidades encontradas.

4.2 Implementación

4.2.1 Sitio vulnerable DVWA

Para la ejecución del sitio, se utilizan diferentes instancias del contender 'aracloud/docker-dvwa' escuchando en diferentes puertos. Dichas instancias reciben las peticiones y generan el log correspondiente. Los logs son almacenados en volúmenes independientes configurados al correr cada contenedor.

4.2.2 Generación de ataques

1. ZaProxy: Se utiliza el contenedor owasp/zap2docker-stable apuntando contra el sitio vulnerable a atacar.

2. Sqlmap: Se utiliza un contenedor Docker con SQLMAP para generar ataques SQL Injection, utilizando una imagen custom, que apunta hacia el sitio vulnerable. Se puede elegir qué tipo de ataque utilizar (B,E,U,T,Q).
   En particular, se levanta una instancia para cada tipo, de esta forma se logra generar logs separados para cada ataque. Los cuales, luego, podrán ser utilizados para el entrenamiento de un modelo de aprendizaje automático que pueda identificar entre estos tipos de ataques SQL.

3. HTTP: Se crea scripts en Python para generar ataques mediante HTTP de forma automática. Se dispone de 4 scripts, uno para cada tipo de ataque: Command Injection Attack, File Inclusion Attack, Sql Injection, Xss Attack, Cross-Site Scripting (XSS). Los ataques son enviados hacia el sitio vulnerable.

4.2.3 Generación de tráfico normal

Una o varias instancias del sitio vulnerable se reservan para la generación de tráfico normal. Se dispone de un script en Python, que se encarga de realizar peticiones aleatorias hacia el sitio vulnerable. Se puede especificar la cantidad de peticiones a realizar.

4.2.4 Visualización

Threadfix es instalado en un contendor https://hub.docker.com/r/jmbmxer/threadfix

Y se pueden importar los reportes generados mediante ZaProxy siguiendo este tutorial: https://softwaretester.info/setup-learning-environment-for-security-testing/

4.2.5 Clasificador de ataques

Se utiliza Python y Jupyter Notebook para la implementación de un clasificador de ataques, entrenado con los datos generados.

4.3 Deploy

4.3.1 Contenedores

- En la nube: puede ser en cualquier plataforma cloud, las pruebas realizas fueron en EC2 y Azure
- Local: se pueden levantar los contenedores localmente, tiene como limitante la memoria y CPU que cuente el ordenador. Por lo que la escalabilidad va a estar restringida a estas características.
- Kubernetes: si se desea una arquitectura más robusta, se puede instalar un cluster de Kubernetes (ya sea de manera local como en una plataforma cloud).

4.3.2 Almacenamiento

Se puede utilizar cualquier tipo de almacenamiento para los logs, ya sea local o remoto.

5 Conclusiones

Se logra la independencia y escalabilidad independiente de los componentes en la arquitectura.

Para construir una arquitectura más robusta se requiere inversión de tiempo de DevOps y tomar decisiones en cuanto a monitoreo de los componentes,

Se valida el uso del framework mediante el entrenamiento de un modelo de aprendizaje automático capaz de identificar entre ataques y no ataques. Probablemente el modelo sufra de sobreajuste dado que solamente está entrenado con ataques para este sitio particular.

6 Próximos pasos

Se definen varias líneas de investigación y trabajo a futuro para mejorar los logs generados por el framework y potenciar el uso de las herramientas. A continuación, se enumeran los puntos identificados:

- Agregar más tipos de ataques: posee pocos tipos de ataques soportados, por lo cual se deberían agregar más ataques para extender las posibilidades de detectarlos.

- Utilizar otros sitios vulnerables: Al generar todos los ataques con el mismo sitio, el algoritmo se torna muy específico a este sitio, el cual no es real. Por lo tanto, agregar más sitios vulnerables, sobre todo agregando algún sitio que se asemeje más a la realidad sería imprescindible.

- Disponibilidad del framework: ahora que la arquitectura fue validada, el framework podría instalarse y estar disponible para aquellos que deseen generar logs de ataques y entrenar sus modelos.

- Generar logs mediante ataques más sofisticados de Hackers expertos: los ataques generados son básicos debido a que el equipo no posee los conocimientos necesarios para realizar otros ataques más complejos. Contar con gente más especializada resultaría beneficioso para que el modelo entrenado sea más robusto.

- Reentrenar el modelo con nuevos ataques: Se podría crear un ciclo de reentrenamiento del modelo ante nuevos ataques generados.

- ML Ops: La utilización de técnicas de ML Ops para automatizar los pasos del pipeline a la otra de entrenar el modelo ayudarían a la mejor gestión del framework, datos generados y registro de modelos entrenados.