HTTP-WS-AD: An Anomaly Detector oriented to web applications and web services

José Giménez
Polytechnical School
National University of Asunción
San Lorenzo, Paraguay
email: jdgimenez@pol.una.py

Cristian Cappo
Polytechnical School
National University of Asunción
San Lorenzo, Paraguay
email: ccappo@pol.una.py

***Abstract*—Web applications have become the most demanded systems to be developed today. This is because they have several advantages compared to "desktop" systems. Due to massive use of web applications they have become one of the main targets for cyberattacks. They are also used as the principal vectors for more sophisticated attacks.**

In this paper we present an online anomaly based detector for web applications which implements various detection models proposed in the literature. Our proposed detector includes new anomaly models for HTTP requests based on XML or JSON which are formats that usually used in web services and AJAX applications. We also present a framework to include and to evaluate new anomaly models in the detector.

Keywords: HTTP, Web Applications, Attacks, Vulnerabilities, Intrusion Detection, Anomalies

I. Introducción

Las aplicaciones web se han convertido en los sistemas más utilizados hoy en día. Según el sitio Internet Live Stats [1] actualmente el 40% de la población actual posee Internet, a diferencia de 1995 en donde solo lo poseía el 1%.

La Figura 1 ilustra el crecimiento en la utilización de Internet desde el año 1993.

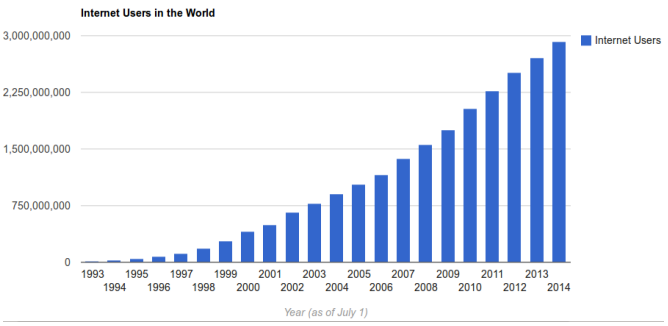

Fig. 1. Crecimiento en la utilización de Internet desde el año 1993

Debido a su alta utilización las aplicaciones web se han convertido en el blanco predilecto de los cyberataques. Según el Reporte de Amenazas en la Seguridad de Internet de Symantec del 2014[2] en el 2012 el 53% de los sitios escaneados tenían vulnerabilidades, de los cuáles el 24% eran críticas. En el año 2013 aumentó a 77%, de los cuáles el 16% eran críticas. El informe también manifiesta que 1 de cada 8 sitios web posee una vulnerabilidad crítica sin corregir.

Con respecto a la cantidad de nuevas vulnerabilidades en el 2012 se encontraron 5291 y en el 2013 un total de 6787. Ataques de día 0 (vulnerabilidades desconocidas por los desarrolladores de software y sin parches conocidos que los corrijan) fueron contados 14 en el 2012 mientras que en el 2013 totalizaron 23.

Las aplicaciones web se han vuelto de gran importancia con el auge de disponibilizar los servicios de software a través de la red y en particular a través de Internet. De esta manera las aplicaciones son accesibles a un mayor público pero también quedan expuestas a ser atacadas a fin de obtener de ellas, en forma ilícita, algún tipo de beneficio. En este sentido, proteger a las aplicaciones web de estos ataques se ha convertido en un requerimiento básico para los responsables de TI de las organizaciones. El desarrollo de aplicaciones seguras, etapa de prevención, es fundamental pero no suficiente. Por esta razón se han construido sistemas que permiten la detección de ataques y eventualmente la defensa contra ellos. Estos sistemas se denominan Sistemas de Detección de Intrusión y en el ámbito de las aplicaciones web son conocidos comúnmente como "Cortafuegos de Aplicaciones Web" o WAF (Web Application Firewall). Estas operan a nivel de la capa de aplicación sobre el protocolo HTTP.

De acuerdo al mecanismo utilizado para la detección, los detectores de intrusión pueden ser basados en firma o basados en anomalía. Los primeros utilizan una base de datos de firmas que corresponden a ataques conocidos y hacen una búsqueda de estas firmas en las peticiones evaluadas. Los segundos modelan el comportamiento usual de las peticiones o el conjunto de ellas, y una desviación significativa de los observado con respecto al comportamiento usual es considerada una anomalía y por tanto un ataque [3].

En este artículo presentamos un detector basado en anomalías para protección de aplicaciones web.

Usualmente los detectores de intrusión basados en anomalía utilizan como entrada para sus análisis los registros de logs de los servidores web (por lo general en formato CLF [1]). Este criterio solo permite analizar peticiones HTTP tipo GET por tanto no modelan las peticiones tipo POST. La importancia de tener un registro de las peticiones completas para el método

[1]Common Log Format

978-1-4673-9143-6/15/$31.00

POST radica en que las peticiones que lo utilizan son las que realizan usualmente un procesamiento de mayor importancia en las aplicaciones web, por ejemplo el procesamiento de los formularios.

La mayoría de los modelos de detección de intrusión basados en anomalías existentes son producto de investigaciones académicas. Esto es debido a que los modelos de intrusión basados en anomalías no son utilizados en sistemas en producción debido a la alta cantidad de falsos positivos que estos generan. Otra consecuencia de que los modelos de detección basados en anomalías son producto de investigaciones académicas es que estos están orientados a ser evaluados de forma "offline".

También, como los modelos de detección de intrusión basados en anomalías implementados están orientados a probar los modelos diseñados por el autor, ninguno de ellos fueron diseñados como frameworks o plataformas genéricas para poder utilizarse para incorporar y evaluar otros modelos de una manera sencilla.

En este trabajo presentamos un detector de anomalías, denominado HTTP-WS-AD, que implementa los modelos de anomalía en tráfico web más relevantes propuestos en la literatura. La implementación es realizada como un proxy reverso lo que permite proteger las aplicaciones sin necesidad de modificarlas y procesar las peticiones tipo GET y POST de modo "online". Además la implementación permite la incorporación y evaluación de nuevos modelos de anomalía. Además de los modelos usuales construidos con el URL del mensaje GET y el cuerpo del mensaje POST consideramos en este trabajo la evaluación de los mensajes de aplicaciones con arquitectura REST [2] considerando los formatos JSON y XML. Finalmente realizamos una evaluación de efectividad y rendimiento con respecto a ModSecurity [4], un WAF (*Web Application Firewall*) Open Source de uso extendido para protección de aplicaciones web.

Este artículo esta organizado de la siguiente manera: la Sección II presenta los principales trabajos relacionados. En la Sección III describimos a los detectores de intrusión y particularmente a los utilizados para protección de aplicaciones web. La arquitectura del detector y la descripción de los modelos implementados son presentados en la sección IV. En la Sección V presentamos los experimentos realizados, los resultados obtenidos tanto en efectividad de detección así como también en tiempo de respuesta. Finalmente presentamos las conclusiones en la Sección VI.

II. Trabajos Relacionados

Presentamos los trabajos relevantes sobre detectores de intrusión en aplicaciones web de acuerdo al modelo de algoritmo que implementan: detección por firmas y detección por anomalías.

[2]Transferencia de Estado Representacional: es un estilo de arquitectura para el diseño de aplicaciones basados en el protocolo HTTP

A. Detección por firmas

El detector más utilizado en este campo es ModSecurity[4]. Posee un motor de reglas flexible en el cuál se pueden configurar las reglas para la detección de ataques. Puede ejecutarse en 2 modos: embebido, en el cuál se ejecuta como un módulo del Servidor HTTP Apache[5] en el cuál se está ejecutando la aplicación web a ser protegida. Al ejecutarse como un módulo de Apache, ModSecurity actúa como un filtro y analiza la petición antes que llegue a la aplicación protegida, pudiendo evitar que pueda violar la seguridad de la aplicación en caso de identificarse como un ataque. El otro modo de ejecución es en un proxy reverso, en el cuál se coloca el proxy reverso delante de los servidores web a ser protegidos, pudiendo evitar que la petición llegue a la aplicación web protegida en caso de identificarse como un ataque.

Existe un Conjunto de Reglas Principales (CRS-Core Rule Set) elaboradas por OWASP [3] el cuál es lo suficientemente genérico para ser aplicado en cualquier aplicación web.

Otro detector popular utilizado para la detección de intrusiones en aplicaciones web basado en firma es PHPIDS[6] para la plataforma PHP. Este software es invocado de manera programática desde una aplicación PHP y puede obtener el impacto calculado para la petición completa o para cada uno de los parámetros y ver cuáles son los tipos de ataques detectados de acuerdo a las etiquetas que PHPIDS asigna (id,xss,csrf,rfe,sqli,lfi,dt,dos). Las acciones a ser realizadas dependen del usuario, pero algunas opciones son: mostrar una página de advertencia, registrar todos los datos de la petición y del cliente y terminar la sesión HTTP.

B. Detección por Anomalías

Para el método de detección de intrusiones en aplicaciones web por anomalía el trabajo pionero fue el presentado por Kruegel *et al.* en [7], propusieron un detector con seis modelos de anomalía: token, longitud, distribución de caracteres y de inferencia de estructura que corresponden a modelos de parámetros y el de presencia/ausencia de atributos y el de orden de los mismos que modelan el "Query String" del URL. Evaluaron los modelos con tres conjuntos de datos: Google y dos servidores web de sus instituciones académicas. Utilizaron un modelo lineal para incorporar el resultado de cada modelo individual pudiendo definirse el peso de cada uno. Demostraron que ningún modelo es capaz de detectar todos los ataques y la importancia de seleccionar y combinar diferentes características de las peticiones y sus atributos para cubrir la detección de la mayor cantidad de ataques. Para la determinación del limiar utilizaron el promedio de todos los modelos para cada parámetro y si alguno de los promedios superaba el 10% de la máxima calificación para ese parámetro con las muestras utilizadas en la etapa de aprendizaje identificaban a toda la petición como un ataque. Aunque la tasa de falsos positivos fue baja no indican la tasa de detección de ataques.

[3]The Open Web Application Security Project

Kruegel *et al.* volvieron a presentar en [8] una extensión de su trabajo con tres nuevos modelos, los cuales evalúan la sesión del usuario: frecuencia de acceso, tiempo entre peticiones y orden de invocación. Realizaron sus pruebas con el mismo conjunto de datos que el trabajo anterior. La tasa de falsos positivos no varió ni tampoco volvieron a demostrar la tasa de aciertos en la detección de ataques.

En [9] a partir del trabajo anterior añadieron un componente de agregación de anomalías, el cuál a partir de varias anomalías detectadas crea un cluster que engloba a todas las anomalías que sean similares entre ellas. El objetivo es presentar una de las anomalías pertenecientes al cluster al administrador del sistema para que categorice a la anomalía como falso positivo o como un ataque. Si el administrador categoriza a la anomalía como una petición normal todas las anomalías pertenecientes al cluster se pueden ingresar al modelo para que no se generen más falsos positivos a partir del cluster. Si el administrador categoriza a la anomalía como un ataque se identificaran a todas las anomalías pertenecientes al cluster como ataques sin que el administrador deba de inspeccionar individualmente cada una de las anomalías.

En [10] se propuso un detector para aplicaciones web utilizando una versión simplificada de la Distancia de Mahalanobis de modo que el cálculo sea rápido y no se convierta en un cuello de botella. El mismo modela todo el paquete de capa de aplicación como una cadena de bytes completa y alcanzaron tasas de detección del 60%.

El trabajo en [11] presentó una comparación entre diferentes modelos implementados para esa época. Entre ellos se encuentran los modelos implementados en [7] y [10]. También evaluaron el modelo de N-Gram que ellos habían propuesto con anterioridad. También compararon un modelo de Autómata Finito Determinista (Deterministic Finite Automatom-DFA) que habían diseñado con anterioridad. En la evaluación realizada los modelos de Longitud, Distribución de Caracteres y la combinación lineal de todos los modelos de Kruegel no alcanzaron la tasa de aciertos del 80%. El basado en la Distancia de Mahalanobis y el Modelo de Markov generan una cantidad alta de falsos positivos: 90 mil y 31 mil falsos positivos por día. Los modelos de DFA y de N-Gram de tamaño de ventana 6 fueron los modelos que mejor se comportaron en sus pruebas.

En [12] implementaron un sistema que modelaba los siguientes atributos de los parámetros HTTP: conjunto de caracteres especiales (no dígitos ni alfanuméricos) permitidos, longitud mínima, longitud máxima, porcentaje mínimo de letras, porcentaje máximo de letras, porcentaje mínimo de dígitos, porcentaje máximo de dígitos, porcentaje mínimo de caracteres especiales, máximo de caracteres especiales. Obtuvieron una alta tasas de detección pero con muchos falsos positivos. El mismo grupo presentan en [13] un sistema que modelaba solamente la longitud de los atributos y la secuencia de caracteres a través de una Cadena de Markov, en el cuál los estados agrupan a los dígitos, letras y caracteres especiales. Obtuvieron una alta tasa de detección pero también 20% de falsos positivos.

En [14] presentan un sistema en el cuál utilizaron dos medidas genéricas de selección para atributos y evaluaron 30 características de los atributos con el objetivo de disminuir el número de atributos que deben de ser modelados, disminuyendo el tiempo de detección sin disminuir de gran manera la tasa de detección del detector. Lograron eliminar el 63% de los atributos irrelevantes y redundantes disminuyendo la tasa de detección en 0.12%. Con ello alcanzaron tasas de detecciones entre el 75% y el 97% con tasas de falsos positivos entre 2.95% y 28%.

III. Detección de Intrusiones

Una intrusión se define como el conjunto de acciones que comprometen la integridad, confidencialidad o disponibilidad de un recurso. La detección de intrusión se define como el proceso de identificar y responder a actividades maliciosas orientadas a recursos computacionales y de red. [15]

Los sistemas de detección de intrusión (Intrusion Detection Systems-IDS) monitorean las actividades de los sistemas computacionales y/o de red para identificar eventos maliciosos o sospechosos. Cada vez que ocurre un evento el IDS genera una alerta. De acuerdo al dominio de monitoreo los sistemas de detección de intrusiones se categorizan en:

1) *Red*: el sistema de detección de intrusión es colocado en la red que se desea monitorear y este analiza los paquetes de red. Posee la ventaja que dicho sistema puede monitorear toda la red a la que tiene acceso. Como desventaja posee que se debe de mantener el estado de la conversación entre todos los pares de nodos para realizar un mejor análisis, lo que podría suponer un consumo de recursos bastante grande, sobre todo para redes amplias y/o con mucho tráfico. El hecho de que el dominio del IDS sea red no implica que solo puede monitorear la capa de red. También puede monitorear las capas de aplicación.
2) *Host*: el sistema de detección de intrusión monitorea los eventos generados por el sistema operativo y las llamadas a librerías. Es más difícil de evadirlo comparado con el basado en red por el mayor grado de integración que posee.
3) *Aplicación*: monitorean los eventos dentro de la aplicación. Posee un mayor grado de integración comparado con el basado en host, por ende es el más difícil de evadir.

A un detector de intrusión que se especializa en la capa de aplicación HTTP se lo denomina Cortafuego de Aplicación Web (Web Application Firewall-WAF)[16]. Los WAF son dispositivos, *plug-in* para servidores o filtros que aplican un conjunto de reglas a una conversación HTTP.

Para detectar los ataques a las aplicaciones web existen dos métodos usuales [3] :

La Detección basada en firmas o plantillas: consiste en generar un conjunto de "firmas" que representan las formas y características que tienen los ataques web conocidos. Sus principales ventajas son: a) posee una tasa de detección muy alta, b) la tasa de falsos positivos es baja y c) puede identificar

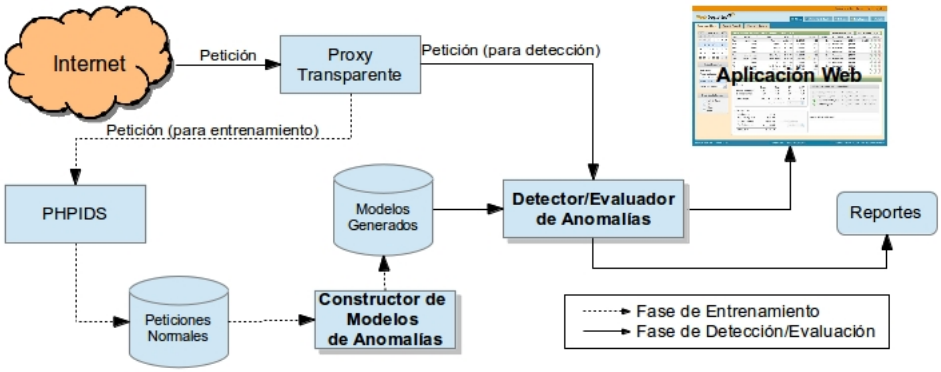

Fig. 2. Arquitectura del detector de anomalías como proxy reverso HTTP-WS-AD

el tipo de ataque de manera certera. Sus desventajas son: a) no es muy eficaz a la hora de detectar ataques nuevos o variantes de conocidos, b) la generación de las firmas requieren una gran cantidad de esfuerzo y tiempo debiéndose ser generadas por personas que tienen una vasta experiencia, dominio del tema y conocimiento del entorno de operación, c) la base de datos debe de ser actualizada frecuentemente que implica una tarea administrativa alta y d) no es escalable, ya que el tamaño de las firmas puede llegar a ser muy grande.

La Detección basada en anomalías: se modela el comportamiento normal de uso de la aplicación, mediante un entrenamiento previo. Cuando se detecta que el comportamiento se aleja considerablemente de lo usual se lo considera una anomalía y así, un ataque. Las principales ventajas que posee este método son: a) puede detectar ataques nuevos y variaciones de los conocidos, b) observa el comportamiento del sistema y genera automáticamente el modelo de comportamiento "normal" sin intervención humana y de manera automática y c) es escalable ya que solo se tiene que almacenar el comportamiento normal del sistema, el cuál es pequeño comparado con la base de datos de los ataques conocidos y sus variantes. Las desventajas del modelo por anomalía son: a) posee una alta tasa de falsos positivos, b) se necesita re-alimentar los datos de para reconstruir los modelos cuando el sistema sufre algún cambio, de forma que el nuevo comportamiento no sea considerado como una anomalía, c) la identificación del tipo de ataque se hace más difícil.

La variabilidad y la creación de nuevos tipos de ataques requieren una atención rápida de seguridad a fin de proteger las aplicaciones. De esta manera el abordaje de detección por anomalías se ha convertido en un método prometedor para identificar ataques y tratar de evitarlos. Este abordaje debe seleccionar las características de las aplicaciones web con el fin de modelar el comportamiento de ellos. Cada característica seleccionada es modelada generalmente de manera diferente y es combinado con otros con el fin de mejorar la detección y reducir el número de falsos positivos.

HTTP-WS-AD evalúa la eficacia de los diferentes modelos de anomalías. Se implementan y analizan los modelos presentados en [7] y [8]. También el modelo basado en el análisis de n-gram presentado en [11] y el modelo de anomalía propuesto en [10]. Estos modelos no tienen en cuenta el caso especial de las peticiones HTTP en formato JSON o XML. Estos dos formatos son generalmente utilizados en la arquitectura REST (por ejemplo, Web Services) y AJAX. Por esa razón se implementaron dos modelos de anomalías con especial énfasis en las peticiones HTTP en dichos formatos. En la sección IV se presentan los detalles del trabajo: los modelos implementados, el generador de modelos de anomalías y la arquitectura del detector/evaluador de anomalías.

IV. Propuesta

A. Arquitectura de HTTP-WS-AD

El modelo de HTTP-WS-AD es de un proxy reverso. Este recibe en forma transparente las peticiones de los clientes de las aplicaciones, las analiza y envía al servidor web que provee los recursos solicitados por el cliente. De este modo se puede colocar delante de los servidores web para analizar el tráfico, para registrar las peticiones y generar los modelos en la fase de entrenamiento o aprendizaje y para analizar las peticiones y evitar que lleguen al servidor web protegido en la fase de detección. Así permite proteger a cualquier aplicación web independientemente de la plataforma utilizada, y es más seguro ya que si el ataque tiene éxito sólo afectará al proxy y no a la aplicación protegida.

La figura 2 muestra la arquitectura del detector de anomalía. Las líneas punteadas indican la fase de entrenamiento o aprendizaje. En esta fase las peticiones son capturadas por

el proxy reverso. Luego estas peticiones son analizadas por un WAF basado en firmas llamado PHPIDS [6]. El objetivo de esta etapa es eliminar las peticiones que son ataques de modo a obtener una base de datos libre de ataques para el entrenamiento. Las peticiones sin ataques obtenidas de esta manera son el insumo para el constructor de los modelos de anomalías. Este módulo genera los modelos y los guarda en la base de datos de los modelos normales.

Las líneas continuas indican la fase de detección/evaluación. En esta fase, el proxy reverso utiliza al módulo de detección/evaluación de anomalías. En este módulo, las peticiones son comparadas con el modelo generado de comportamiento normal. Una desviación considerable es considerada un ataque. Además, este módulo genera informes sobre la efectividad de los modelos.

En la etapa de entrenamiento las peticiones son desagregadas en sus diversos componentes y guardadas en una base de datos relacional, para este trabajo consideramos el uso de PostreSQL[4]. A partir de estos datos registrados se construyen los modelos de anomalía que se utilizarán más tarde en el proceso de detección. Estos modelos también son registrados en la base de datos. El proceso de detección es iniciado por el administrador en forma selectiva por aplicaciones a proteger.

B. Modelos de anomalía implementados

Se implementaron varios modelos de anomalía. Clasificamos los modelos en tres categorías: modelo de parámetros, modelo de query string y modelo de sesión. A continuación describimos cada modelo implementado según la categoría.

Modelos de Parámetros: modelan un parámetro a la vez. No importa si la definición del parámetro está definido en el query string (GET) o en el cuerpo de la petición HTTP (POST). El URL de abajo contiene tres parámetros (modo,precio y B1), los que se modelarán en forma separada.

```
http://<host>/<app>?modo=insertar&precio=1254&B1=ok
```

Por cada parámetro del URL o del body del mensaje HTTP se construyen los siguientes modelos de parámetros:

1) Token: Tiene como propósito detectar cuando los valores que puede representar un parámetro proviene de un conjunto de valores pequeño. La detección lo realiza calculando dos funciones que se definen a continuación.

x toma los valores $1..n$, donde n es la cantidad de valores observados para el parámetro

$$f(x) = x, \tag{1}$$

$$g(x) = \begin{cases} g(x-1)+1 & \text{si el valor x es nuevo} \\ g(x-1)-1 & \text{si el valor x ya fue observado} \\ 0 & \text{si x=0.} \end{cases} \tag{2}$$

La función $f(x)$ es una función creciente y el comportamiento de la función $g(x)$ crece si el valor actual es nuevo y decrece si ya fue observado con anterioridad. A partir de estas dos funciones se calcula la correlación que existen entre ellas, utilizando esta medida para determinar si las muestras fueron repeticiones o valores nuevos por medio de la expresión $\rho = \frac{covar(f,g)}{\sqrt{var(f)*var(g)}}$. Si ρ es menor a 0 entonces las funciones están correlacionadas negativamente y se asume que es una enumeración. Si es una enumeración se registran todos los valores observados. Si el valor que se desea probar se encuentra registrada se acepta dicha entrada, sino se considera una anomalía.

2) Longitud: Registra las longitudes de los parámetros observados y forma una distribución a partir de ella. Se calcula la media μ y la varianza σ^2 de la muestras y se utiliza la inecuación de Chesbychev para determinar la probabilidad entre la longitud de una cadena y la media. La inecuación de Chesbychev dada por la ecuación 3 coloca un límite superior en la probabilidad que la diferencia entre el valor de una variable x y μ exceda a un límite t.

$$p(|x-\mu| > t) < \frac{\sigma^2}{t^2} \tag{3}$$

En la ecuación 3 se reemplaza el valor de t por la diferencia entre la longitud de un valor l y la media μ (ecuación 4, ya que se considera que si el valor de l está alejada de la media μ la probabilidad de que este valor sea válido debe ser pequeña. Para las cadenas con longitud menor o igual a μ la probabilidad $P(l) = 1$.

$$p(|x-\mu| > |l-\mu|) < p(l) = \frac{\sigma^2}{(l-\mu)^2} \tag{4}$$

3) Distribución de Caracteres: Se basa en la observación que los datos ingresados en los campos de los formularios son palabras pertenecientes a un idioma utilizado y por ende los caracteres que componen dichas palabras son "caracteres imprimibles", siendo la mayoría de ellos caracteres alfanuméricos y la minoría caracteres especiales.

Se define como "Distribución de Caracteres" el cálculo de la frecuencia relativa de aparición de cada carácter en una muestra y su posterior ordenación por dicha frecuencia.

Cualquiera sea el idioma utilizado, existen caracteres cuya frecuencia es superior a las otras, pero si llega a calcularse la Distribución de Caracteres se notará que el descenso en la frecuencia de dichos caracteres se realiza de manera gradual.

Se puede observar que existe un descenso brusco en la frecuencia relativa de aparición de caracteres en un ataque como el "Buffer Overflow" donde se utiliza repetidamente un carácter de relleno o la aparición en un gran numero de veces del carácter punto (.) en un ataque de tipo "Directory Traversal".

Se define como "Distribución de Caracteres Ideal" (ICD) la distribución de caracteres que posee una forma normal (no anómala). Para obtener la Distribución de Caracteres Ideal se calcula la distribución de caracteres de cada una de las muestras y se promedian. Esto es realizado estableciendo ICD(n) a la media de la enésima entrada de las distribuciones de caracteres de las muestras $\forall n : 0 \leq n \leq 255$..

Para determinar si la distribución de caracteres de una muestra observada pertenece al ICD se utiliza el test χ^2

[4]http://www.postgresql.org

de Pearson, usando el valor de confianza devuelto como la probabilidad de que la distribución de caracteres de un valor observado es una muestra perteneciente del ICD.

El test χ^2 requiere que el dominio de la función sea dividido en un número pequeño de intervalos. Para ello se agrupan las entradas del ICD en 6 intervalos de la siguiente manera: {[0],[1,3],[4,6],[7,11],[12,15],[16,255]}. Se realiza de la siguiente manera:

- Cuando se observa el valor de un atributo se calcula la cantidad repeticiones de cada carácter, se ordenan de manera descendente y son combinados en el mismo intervalo. A esto se lo denomina "Frecuencia Observada" O_i.
  La "Frecuencia Esperada" E_i es calculada multiplicando las frecuencias relativas de cada uno de los seis intervalos determinados para el ICD por la longitud de la cadena.
- Calcular el valor de χ^2 de la siguiente manera: $\chi^2 = \sum_{i=0}^{i<6}(O_i - E_i)^2/E_i$
- Determinar los grados de libertad y obtener la significancia. Los grados de libertad del test χ^2 es 5 debido a la utilización de 6 intervalos. La probabilidad que la muestra pertenezca a la distribución de caracteres ideal se obtiene de la tabla utilizando el valor χ^2 como índice.

4) Inferencia de Estructura: Intenta detectar los ataques que no son detectados por los modelos anteriores. Intenta inferir la gramática regular que describe todos los valores normales.

Para representar la gramática regular se considera que las muestras pertenecen a una "Gramática Probabilística". Una Gramática Probabilística es una gramática que asigna probabilidades a sus producciones.

Los modelos de Markov son Autómatas Finitos no Deterministas (NFA) que son adecuados para representar a una Gramática Probabilística. Cada estado del autómata posee un conjunto de símbolos que pueden ser emitidos, donde cada uno de estos símbolos posee una probabilidad asociada para ser emitidos . También posee un conjunto de transiciones que pueden ser utilizados con una probabilidad asociada.

La probabilidad de que una palabra w sea producida por el Modelo de Markov se calcula sumando las probabilidades de todos los caminos dentro del modelo que producen dicha palabra. La probabilidad de un camino es el producto de las probabilidades de los símbolos emitidos $p_{s_i}(o_i)$ y las transiciones tomadas $p(t_i)$.

$$p(w) = \prod_{\text{(caminos p en w)}} \sum_{\text{(estados}\in\text{p)}} p_{s_i}(o_i) * p(t_i) \tag{5}$$

El objetivo del proceso de inferencia de estructura es encontrar un AFN que posea la probabilidad más alta para los elementos de entrenamiento dados. Para eso se utiliza el Teorema de Bayes:

$$Pr(M|DE) = \frac{Pr(DE|M).Pr(M)}{Pr(DE)} \tag{6}$$

Donde M es el Modelo y DE son los Datos de Entrenamiento en la ecuación 6 y $Pr(D|DE)$ es la probabilidad de los datos dado el modelo M. La probabilidad de los datos de entrenamiento es considerado un factor de escala y es ignorado. La probabilidad de los datos de entrenamiento dado el modelo puede ser calculado sumando las probabilidades de cada uno de los de los elementos utilizados para entrenamiento.

La probabilidad a priori del modelo fue diseñado con el objetivo de demostrar que los modelos pequeños son priorizados. Dicha probabilidad fue calculado de manera heurística y utiliza el número total de estados N y la cantidad de transiciones y emisiones de cada estado S:

$$p(M) = \frac{1}{\prod_{S\in Estados}(N+1)^{\sum_s transiciones} * (N+1)^{\sum_s emisiones}} \tag{7}$$

Ambos factores crean un balance entre los modelos que representan exactamente los datos de entrenamiento pero son muy complejos y los modelos pequeños que generalizan demasiado.

La construcción del modelo inicia con un AFN que representa exactamente los datos de entrada. Luego procede a realizar la mezcla de estados hasta que la multiplicación de factores ya no aumente.

El costo de construcción del modelo es $O((n*l)^3)$, donde n es la cantidad de cadenas analizadas y l es la longitud de la mayor. Dicho costo es altamente prohibitivo.

Una vez construido el modelo se verifica si el valor que se desea probar es una producción del modelo. Si es una producción del modelo se acepta sino se rechaza.

5) N-Gram: Un N-Gram es una subcadena generada por una ventana deslizante de tamaño N a través de una cadena de caracteres [17]. El resultado es un conjunto de caracteres de tamaño N. Primero se define el tamaño de la ventana N y luego se procede a registrar todos los N-Grams para todos los valores de entrenamiento. Cuando se desea evaluar una cadena se procede a calcular los N-Grams de esa cadena y se calcula la relación entre el número de n-grams observados y los n-grams registrados en el entrenamiento de acuerdo a la expresión $p = \frac{\text{\#n-grams observados}}{\text{\#n-grams entrenamiento}}$. Un valor de p cercano a 1 indica que la cadena evaluada es normal.

Dependiendo de la elección de N el modelo de N-Gram puede modelar desde caracteres individuales hasta frases completas, siendo una alternativa de bajo costo comparada con el Modelo de Markov. Cabe señalar que la elección de N es crucial ya que para un valor de N grande puede dar lugar a muchos falsos positivos.

6) Distancia de Mahalanobis: La distancia de Mahalanobis es una métrica de distancia estándar para comparar dos distribuciones estadísticas. La misma se representa por la ecuación 8.

$$d^2(x,\overline{y}) = (x-\overline{y})^T C^{-1}(x-\overline{y}) \tag{8}$$

en donde x e $\overline{y}$ son 2 vectores de atributos y cada elemento del vector es una variable. x es el vector de atributos de

la nueva observación e $\overline{y}$ es el vector de atributos promedio calculado a partir de las muestras de entrenamiento. C^{-1} es la inversa de la matriz de covarianza, con $C_{ij} = Cov(y_i, y_j)$ e y_i y y_j son el i-ésimo y j-ésimo elementos de los vectores de entrenamiento.

La ventaja de esta métrica es que tiene en cuenta el valor promedio, la varianza y la covarianza de las variables medidas. En lugar de calcular la distancia de los promedios asigna un peso a cada variable utilizando su desviación estándar y su covarianza, logrando una medida estadística que califica si el nuevo ejemplo es consistente con las muestras de entrenamiento. Su desventaja es su requerimiento computacional. Por este motivo en [10] se propuso una distancia de Mahalanobis simplificada dada por la ecuación 9, en la cuál se asume los caracteres son estadísticamente independientes. De esta manera la matriz de covarianza se vuelve diagonal y los elementos en la diagonal son la varianza de cada carácter.

$$d(x, \bar{y}) = \sum_{i=0}^{n-1} \frac{(|x_i - \bar{y}_i|)}{\bar{\sigma} + \alpha} \tag{9}$$

donde x es el valor que se desea comparar, y es el promedio de los valores observados, n es 256 ya que con un byte (8 bits) se pueden representar 256 símbolos diferentes y σ es una desviación estándar. Los autores propusieron como umbral 256, lo cuál equivale a una desviación estándar. Usando este umbral se pueden categorizar los valores de prueba para detectar una anomalía.

Modelos de Query String

Los modelos de query string caracterizan la relación existente entre los diferentes parámetros enviados a una URL, sin tener en cuenta sus valores. No importa si los parámetros fueron enviados en el query string o en el cuerpo de la petición.

Como ejemplo se resaltan las partes del la URL que se tienen en cuenta.

```
http://<host>/<app>?modo=insertar&precio=1254&B1=ok
```

7) Presencia o Ausencia de Atributos: Modela los conjuntos de parámetros que son vistos durante la etapa de entrenamiento y en la etapa de detección acepta un conjunto de parámetros si estos fueron vistos en la etapa de entrenamiento.

8) Orden de Atributos: Esta orientado a los formularios que son muy dinámicos en los cuáles los parámetros que se envían al servidor son seleccionados por alguna elección realizada por el usuario.

Es más flexible que el anterior debido a que no tuvo que haber visto el conjunto de parámetros exacto en la etapa de entrenamiento para que sea aceptado, sino que comprueba que todos los parámetros enviados tengan un orden relativo entre ellos que sea consistente con el orden relativo observado en la etapa de entrenamiento.

Para ello genera un conjunto, cuyos elementos son vértices de un grafo que cumple con la ecuación 10

$$O = \{(a_i, a_j) : a_i \text{ precede } a_j \text{ y } a_i, a_j \in (S_{q_j} : \forall j = 1, .., n)\} \tag{10}$$

Esto es, el parámetro a_i precede el parámetro a_j si en todas las peticiones en las que aparecen ambos parámetros a_i estaba siempre antes que a_j.

Este conjunto se genera añadiendo los atributos (a_s, a_{s+1}) al conjunto O, donde $1 \leq s \leq (i-1)$ donde i es la cantidad de atributos que posee la petición que está siendo procesada actualmente.

Luego de este proceso el conjunto O posee todas las restricciones de precedencia entre todos los atributos vistos, ya sea por una precedencia directa o por una secuencia de precedencias. Pero dicho conjunto puede llegar a contener ciclos. Para eliminar los ciclos se identifican los componentes fuertemente conectados mediante el algoritmo de Tarjan y se eliminan todos los vértices que pertenecen al mismo componente fuertemente conectado.

En la etapa de detección se comprueba si la arista (a_{s+1}, a_s) se encuentra dentro del conjunto O. Si se encuentra se detectó una violación de orden.

C. Modelo de JSON/XML

Tienen un objetivo similar a los Modelos de Presencia o Ausencia de Atributos y Orden de Atributos: representar la estructura del mensaje HTTP completo sin tener en cuenta los valores de cada parámetro en particular.

Pero los Modelos de Presencia o Ausencia de Atributos y Orden de Atributos fueron diseñados para los mensajes HTTP que posee el formato "application/x-www-form-urlencoded".

Estos modelos identifican si los mensajes son del tipo XML y registran el orden de los elementos junto con sus atributos. En la etapa de entrenamiento registra todas las estructuras observadas y en la etapa de detección comprueba que esa estructura haya sido observada. Si no ha sido observada se considera una anomalía. De modo similar el proceso se realiza para mensajes en formato JSON.

D. Modelos de Sesión

Los modelos de sesión modelan la secuencia de URL's que son accedidas por un usuario. En el conjunto de URL de abajo se subrayan los datos tenidos en cuenta por el modelo:

```
http://localhost:8080/tienda1/index.jsp?modo=ins
http://localhost:8080/tienda1/publico/autenticar.jsp?modo=ins
http://localhost:8080/tienda1/miembros/index.jsp?modo=ins
http://localhost:8080/tienda1/publico/productos.jsp?modo=ins
http://localhost:8080/tienda1/publico/carrito.jsp?modo=ins
http://localhost:8080/tienda1/publico/anadir.jsp?modo=ins
http://localhost:8080/tienda1/publico/pagar.jsp?modo=ins
http://localhost:8080/tienda1/miembros/salir.jsp?modo=ins
```

1) Frecuencia de Acceso: Registra la Frecuencia de Acceso a un programa en particular. Para ello toma en cuenta dos frecuencias. Una de ellas es la frecuencia de acceso de todos los clientes a dicho programa y la otra es la frecuencia de acceso de un cliente en particular.

En [8] utilizaba las direcciones IP para asociar las peticiones a un cliente particular. Nosotros decidimos no utilizar las direcciones IP debido a la alta utilización de Network Address Translation (NAT) y por ende utilizamos el valor de un cookie

para asociar las peticiones a un cliente. El valor del cookie debe ser configurado de antemano.

Para esto se dividen las peticiones registradas en intervalos de 10 segundos. Dentro de cada uno de estos intervalos se calcula la cantidad de accesos de todos los clientes y la cantidad de acceso de los distintos clientes. Estas cantidades forman 2 distribuciones: una para todos los clientes y otra para un solo cliente.

Con estas 2 distribuciones se utiliza el Modelo de Chesbychev utilizado por el Modelo de Longitud para comparar la frecuencia de acceso registrada con la que se desea probar.

En la etapa de detección se calculan los valores actuales de acceso para todos los clientes y para el cliente en particular en el mismo intervalo utilizado en la etapa de aprendizaje. Con esto se calculan las probabilidades de Chesbychev para estas 2 distribuciones y se promedian.

2) Orden de Invocación: Modela la secuencia de URL que son accedidas por los clientes al utilizar un sistema. Para ello utiliza el valor de un cookie configurado de antemano. Kruegel utilizó de vuelta las dirección IP de los clientes.

Utiliza como base el Modelo de Markov utilizado por el Modelo de Inferencia de Estructura, pero a diferencia de este Modela secuencia de URL's y no la estructura de un parámetro.

Se realiza el mismo proceso que el modelo de Inferencia de Estructura en la mezcla, reducción y generalización del modelo. También realiza el mismo proceso en la etapa de detección.

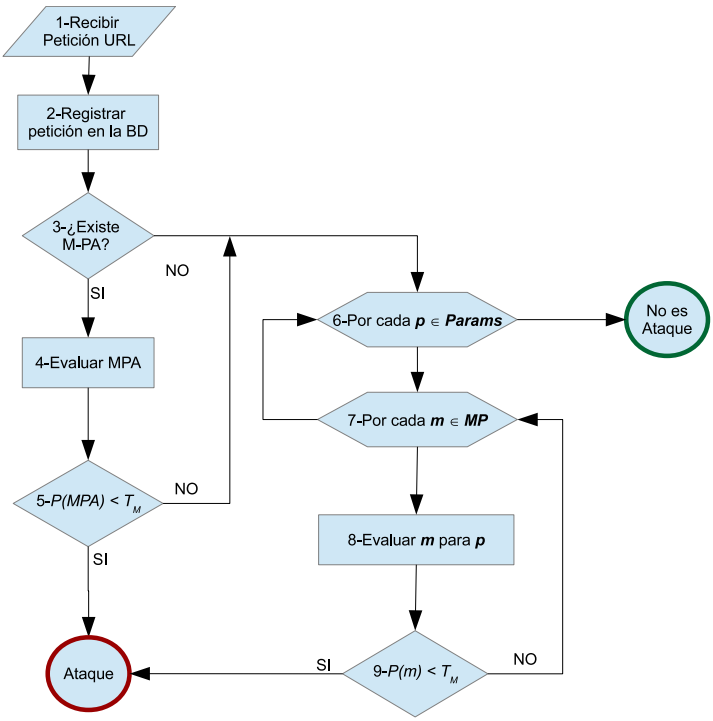

Fig. 3. Diagrama de detección de HTTP-WS-AD

E. Modelos de Agregación

Los Modelos de Agregación utilizan las evaluaciones de varios modelos para generar una probabilidad como resultado de dicha agregación.

1) Modelo de Kruegel: En [8], [8] se realizaba una sumatoria ponderada de las probabilidades calculadas por todos los modelos utilizados para otorgar un puntaje de anomalía a una petición de acuerdo a la ecuación 11.

$$\text{Puntaje de Anomalia} = \sum_{m \in Modelos} w_m * (1 - p_m) \qquad (11)$$

En esta ecuación w_m representa el peso asignado al modelo m y p_m es la probabilidad calculada por dicho modelo. En los experimentos realizados por los autores el peso para todos los modelos es 1.

2) Modelo de Cascada: Propuesto en [18]. Dicha agregación esta representada por un diagrama de flujo en el cuál en un nodo se realizan ciertas verificaciones y de acuerdo a ello se elige el siguiente nodo a procesar. Un nodo puede ser un estado final o un estado intermedio.

F. Funcionamiento del Detector

En la figura 3 se observa el proceso de detección de HTTP-WS-AD. Por cada petición que llega al detector (1) se procede registrarla eventualmente en la base de datos(2). El primer paso que se realiza para verificar si la petición es aceptable es observar entre los modelos guardados si se generó el modelo de Presencia o Ausencia (3). En caso de que exista tal modelo para el URL y el tipo de método HTTP (POST o GET) entonces se evalúa lo observado con respecto al modelo generado en el entrenamiento. Si la probabilidad de que lo observado es menor a una probabilidad threshold T_M entonces se considera un ataque(5). Si la probabilidad es mayor o igual a T_M entonces en (6) iteramos por cada parámetro p que posee la petición analizada. Iteramos por cada modelo de parámetros MP (Token, longitud de caracteres, distancia de Mahalanobis, N-Gram-1) (7). Verificamos si existe generado el modelo m de MP para el parámetro p (8). Si la probabilidad de que el parámetro p observado corresponda al modelo m de MP es menor a un threshold T_M entonces se considera un ataque (9), de lo contrario se continua evaluando el siguiente modelo y luego el siguiente parámetro.

De forma a administrar mejor el funcionamiento del detector se construyó una interfaz de administración en donde se pueden visualizar el estado del detector, los datos registrados, las estadísticas y cambiar de estado a los modelos a la etapa de detección.

G. Inserción de nuevos modelos a HTTP-WS-AD

HTTP-WS-AD permite agregar nuevos modelos de anomalía, generarlos, agregarlos a la base de datos de modelos y luego utilizarlos para la evaluación de las peticiones. De esta forma nuestro detector se convierte en un framework de trabajo. Esta facilidad permite, especialmente a los que trabajan en el área de modelos de anomalías, tener un entorno

real de pruebas pudiendo concentrarse en el modelo mismo y no perder tiempo en cuestiones técnicas de implementación. Para agregar un nuevo modelo se crea una clase PHP con algunos métodos definidos que son de fácil comprensión. Debido a un tema de espacio no colocamos un ejemplo completo de definición de un nuevo modelo para el detector.

V. Experimentos y Resultados

A. Conjuntos de Datos de Prueba

Para el desarrollo de los experimentos fue utilizado el *dataset* CSIC 2010[19]. El mismo contiene datos de un sistema de e-commerce que se encuentra en idioma español. Por ende posee caracteres acentuados, los cuáles pueden afectar a detectores basados en firmas que buscan caracteres especiales. Es importante resaltar que este *dataset* es uno de los pocos disponibles con datos marcados y utilizados en varios trabajos por ejemplo [20].

Posee tres subconjuntos de datos: el conjunto de entrenamiento que posee 36000 peticiones sin ataques(con 84000 muestras de parámetros) y dos conjuntos de pruebas: el conjunto de ataques que poseen más de 24028peticiones (con 96670 muestras de parámetros) y el de peticiones sin ataques que posee 35877 peticiones (con 73882 muestras de parámetros).

Entre las peticiones con ataques se encuentran los ataques de SQLInjection, buffer overflow, obtención de información, divulgación de archivos, inyección CRLF, XSS, server side include, manipulación de parámetros entre otros.

Para registrar las peticiones en la base de datos se implementó una aplicación que lea los archivos del CSIC y lo persista en la base de datos, de la misma manera como lo hubiera realizado el proxy reverso.

En la base de datos de ataques del CSIC había un total de 24425 peticiones, de los cuáles 397 peticiones utilizaban el método HTTP PUT. Como no se observó ninguna petición con ese método en la etapa de aprendizaje dichas peticiones no fueron introducidas a la base de datos.

B. Experimentos

En primera instancia se evaluó la eficacia de los detectores por firma para el dataset. En primer lugar PHPIDS y luego Modsecurity. Para ello se implementó una aplicación que lea los archivos que poseen los conjuntos de datos de prueba y los reenvíe a un servidor Apache que posee instalado Modsecurity configurado con las reglas por defecto obtenidas del sitio de OWASP. Posteriormente evaluamos los modelos de anomalía implementados. Las alternativas en que se determinaba si un parámetro es un ataque son las siguientes:

1) Al menos uno de los modelos evaluó al parámetro como un ataque (probabilidad menor a T_M).
2) Al menos dos de los modelos evaluó al parámetro como un ataque (probabilidad menor a T_M).
3) Se calculó el promedio entre las probabilidades de los 18 modelos.

Se diseñaron los modelos de cascada, en los cuáles se van evaluando los modelos en un orden definido. Si el modelo evaluado actualmente brindó una probabilidad menor a T_M se considera un ataque, sino se evalúa el siguiente modelo.

Se diseñaron e implementaron 4 modelos de cascada:

1) Cascada 1 con promedio: Presencia o Ausencia de Atributos, Token, Longitud, Distancia de Mahalanobis, N-Gram-1, promedio de los modelos de N-Gram con tamaño de ventana desde 2 hasta 15.
2) Cascada 1 sin promedio: Presencia o Ausencia de Atributos, Token, Longitud, Distancia de Mahalanobis, N-Gram-1.
3) Cascada 2 con promedio: Presencia o Ausencia de Atributos, Token, Longitud, Distancia de Mahalanobis, N-Gram-1, N-Gram-2, de los modelos de N-Gram con tamaño de ventana desde 3 hasta 15.
4) Cascada 2 sin promedio: Presencia o Ausencia de Atributos, Token, Longitud, Distancia de Mahalanobis, N-Gram-1, N-Gram-2.

Luego se creó un Modelo de Longitud en el cuál se registran las longitudes de las muestras observadas y en etapa de detección se aceptan las muestras con longitudes menores o iguales a la máxima longitud observada. Con este modelo de Longitud modificado se volvieron a ejecutar las pruebas de los modelos de cascada.

Para todas las pruebas consideramos el valor de probabilidad $T_M = 0.9$, es decir consideramos un valor normal si la probabilidad de que pertenezca a un modelo en particular es mayor o igual al 90%.

Para las evaluaciones consideramos las siguientes métricas: TP(True positive) es la cantidad de ataques verdaderos detectados, FP(False positives) es la cantidad de ataques detectados que no lo son, FN(False negative) es la cantidad de ataques que no han sido detectados, TN(True Negative) es la cantidad de no ataques detectados, TPR(True Positive Rate) es el ratio de aciertos dado por $TPR = TP/(TP + FN)$, FPR(False Positive Rate) es el ratio de aciertos falsos dado por $FPR = FP/(FP + RN)$ y finalmente el valor de F que relaciona los falsos positivos con los verdaderos positivos y esta dada por $F = 2TP/(2TP + FP + FN)$. Un detector tiene un buen resultado cuando la medida F se acerca al valor 1.

C. Resultados

1) Detectores por firmas: La tabla II muestra los resultados de utilizar los detectores por firma. Puede notarse un valor de F para PHPIDS es bastante pobre ya que la detección de los ataques fue bastante baja. En el caso de ModSecurity el resultado fue mucho mejor que PHPIDS aunque todavía no se logra resultados competitivos, probablemente por el hecho de que se tiene que realizar una ajuste en las reglas.

Tabla II
Resultados de detectores por firmas

Detector	TP	FN	FP	TN	TPR	FPR	F
PHPIDS	5562	13457	3272	28728	0,292	0,102	0,399
ModSecurity	13355	10673	0	71877	0,555	0	0,714

Modelo de Cascada

Tabla I
COMPARACIÓN DE AGRUPACIONES DE MODELOS

Medida	1M	2M	P	C2SP	C2CP	C1SP	C1CP	1MLM	2MLM	PML	C1SPLM	C1CPLM
TP	24028	19002	24000	24028	24020	23970	24028	24028	19002	24028	23755	24028
FN	0	5026	28	0	8	58	0	0	5026	0	273	0
FP	12932	7804	6636	12932	10130	9038	12932	7804	7804	7804	116	7804
TN	58945	64073	65241	58945	61747	62839	58945	64073	64073	64073	71761	64073
TPR	1	0,790	0,998	1	0,999	0,997	1	1	0,790	1	0,988	1
FPR	0,179	0,108	0,092	0,179	0,140	0,125	0,179	0,108	0,108	0,108	0,001	0,108
F	0,787	0,747	0,878	0,787	0,825	0,840	0,787	0,860	0,747	0,860	0,991	0,860

2) Evaluación por agrupación de modelos: La tabla I presenta los resultados de los diferentes modelos de agregación diseñados, implementados y evaluados. Consideramos para los experimentos el valor de 0.9 (90%) para T_M Las leyendas utilizadas

- 1M: Un modelo menor a una probabilidad T_M.
- 2M: Dos modelos menores a una probabilidad T_M.
- P: Promedio de todos los modelos.
- C2SP: Modelo de Cascada 2 sin Promedio.
- C2CP: Modelo de Cascada 2 con Promedio.
- C1SP: Modelo de Cascada 1 sin Promedio.
- C1CP: Modelo de Cascada 1 con Promedio.
- 1MLM: Un modelo menor a una probabilidad T_M con Longitud Modificada.
- 2MLM: Dos modelos menores a una probabilidad T_M con Longitud Modificada.
- PLM: Promedio de todos los modelos con Longitud Modificada.
- C1SPLM: Modelo de Cascada 1 sin Promedio con Longitud Modificada.
- C1CPLM: Modelo de Cascada 1 con Promedio con Longitud Modificada.

D. Discusión de los resultados

1) Modelos de Kruegel: Realizando pruebas sobre los Modelos de Kruegel implementados se notó lo siguiente:

1) El Modelo de Distribución de Caracteres detectó a todas las peticiones como ataques para este conjunto de datos de prueba. Analizando la causa se detectó que el Modelo de Distribución de Caracteres necesita que todos los intervalos posean una cantidad mínima de elementos, que según la literatura debería de ser 5.
   Como el último intervalo posee los caracteres en el rango [16-255] en las muestras se debería de observar 21 caracteres diferentes. Esto no se cumple para la mayoría de los parámetros de este conjunto de datos, ya que estos poseen valores que son poco extensos. Por ende no es confiable la probabilidad calculada por dicho modelo y su aplicabilidad se reduce a aquellos parámetros extensos.
2) El Modelo de Inferencia de Estructura de Kruegel tiene un costo de aprendizaje prohibitivo, así como se planteó en su explicación. Realizando pruebas se notó esto y se decidió continuar las pruebas sin utilizar dicho modelo.

2) Detectores por firma: Con PHPIDS (Tabla II) se obtuvieron resultados muy bajos. Esto nos motivó a realizar las pruebas también con Modsecurity. Con este detector se obtuvieron mejores resultados comparado con los obtenidos con PHPIDS. A pesar de ello Modsecurity solo ha detectado alrededor de un 56% de los ataques.

3) Evaluación por Agregación de Modelos: Primero se diseñaron y probaron los siguientes Modelos de Agregación:

1) Al menos uno de los modelos evaluó al parámetro como un ataque (probabilidad menor a T_M).
2) Al menos dos de los modelos evaluó al parámetro como un ataque (probabilidad menor a T_M).
3) Se calculó el promedio entre las probabilidades de los 18 modelos.

Al observar la alta Tasa de Falsos Positivos obtenidas por estas agrupaciones se analizó las evaluaciones realizadas por cada uno de los modelos de manera individual a fin de determinar cuáles son los modelos que influyen de manera negativa. Para ello se analizó los resultados obtenidos por los modelos para el grupo de peticiones normales para pruebas. A continuación se presentan los resultados obtenidos para el grupo de peticiones normales para pruebas:

Tabla III
PETICIONES NORMALES PARA PRUEBAS

Modelo	**FP**	**TN**
Presencia o Ausencia Atributos	0	35877
Token	4	73878
Longitud	2914	70968
Distancia de Mahalanobis	6	73876
N-Gram-1	4	73878
N-Gram-2	2648	71234
N-Gram-3	14122	59760
N-Gram-4	29522	44360
N-Gram-15	38706	35176
N-Gram-14	38706	35176
N-Gram-13	38702	35180
N-Gram-12	38702	35180
N-Gram-11	38700	35182
N-Gram-10	38694	35188
N-Gram-9	38678	35204
N-Gram-8	38648	35234
N-Gram-7	38586	35296
N-Gram-6	38500	35382
N-Gram-5	38326	35556

Al realizar el análisis de los datos de la tabla III se notó que los únicos modelos que no generan una gran cantidad de Falsos Positivos son los siguientes: Presencia o Ausencia

de Atributos, Token, Longitud, Distancia de Mahalanobis, N-Gram de tamaño de ventana 1 y N-Gram de tamaño de ventana 2. El resto de los modelos poseen una Tasa de Falsos Positivos muy altas como para ser tenidos en cuenta en la Agregación de Modelos.

El orden de evaluación de los modelos en el Modelo de Cascada se definió teniendo como objetivo que primero se evalúen los modelos que generan una menor cantidad de Falsos Positivos. Es por esto que los modelos que no generan una gran cantidad de Falsos Positivos son evaluados primero.

Teniendo como siguiente objetivo la eficiencia del Modelo de Cascada se definió que el primer modelo a ser evaluado sea el Modelo de Presencia o Ausencia de Atributos, ya que este modelo detectará si el conjunto de parámetros fue observado en la etapa de entrenamiento. Si esto no es así se calificará a la petición como un ataque y no hará falta examinar el resto de los parámetros.

El resto de los modelos son todos Modelos de Parámetros. Entre estos se encuentra el Modelo de Token. Si este modelo determina que los valores de un parámetro pertenece a un conjunto finito de valores posibles entonces se debe de respetar lo que indica este modelo y ya no es necesario continuar evaluando el resto de los modelos. Es por esto que el Modelo de Token es el primer modelo a ser evaluado para un parámetro.

Si el Modelo de Token no determinó que un parámetro pertenece a un conjunto finito de valores posibles entonces el siguiente modelo a ser evaluado es el Modelo de Longitud. Esto es debido a que la mayoría de los ataques radican en la introducción de una cadena relativamente larga para la ejecución de un comando.

Si la longitud del parámetro no indica una anomalía hay que revisar la distribución de los caracteres. Entre los 2 modelos restantes, que modelan la distribución de caracteres, ambos modelan un carácter a la vez. Entre ambos se optó por evaluar primero la Distancia de Mahalanobis ya que este tiene en cuenta más atributos que el Modelo de N-Gram de tamaño de ventana 1. Esto es debido a que el Modelo de N-Gram de tamaño de ventana 1 solo modela que caracteres observó en la etapa de entrenamiento sin tener en cuenta su frecuencia de aparición. La Distancia de Mahalanobis sí tiene en cuenta la frecuencia ya que en su cálculo utiliza la media y desviación estándar de los caracteres.

El Modelo de N-Gram-2 (N=2) posee una tasa de falsos positivos alta comparada con el resto de los modelos que poseen una tasa de falsos positivos baja. Es por ello que cuando es utilizado se realiza luego de evaluar el resto de los modelos citados anteriormente. Hay variaciones en donde es utilizado dicho modelo y otras en la que no.

Como el resto de los modelos obtuvieron una Tasa de Falsos Positivos muy alta se optó agregar dichos modelos mediante un promedio. Hay variaciones en las cuáles se utiliza dicho modelo y otras en la que no.

De acuerdo al análisis realizado se diseñaron los siguientes Modelos de Cascada:

1) Cascada 1 con promedio: Presencia o Ausencia de Atributos, Token, Longitud, Distancia de Mahalanobis, N-Gram-1, promedio del resto.
2) Cascada 1 sin promedio: Presencia o Ausencia de Atributos, Token, Longitud, Distancia de Mahalanobis, N-Gram-1.
3) Cascada 2 con promedio: Presencia o Ausencia de Atributos, Token, Longitud, Distancia de Mahalanobis, N-Gram-1, N-Gram-2, promedio del resto.
4) Cascada 2 sin promedio: Presencia o Ausencia de Atributos, Token, Longitud, Distancia de Mahalanobis, N-Gram-1, N-Gram-2.

A pesar de haber analizado los modelos de manera individual y haber determinado cuales son los modelos que poseen una alta Tasa de Falsos Positivos y diseñado los Modelos de Cascada se volvieron a obtener resultados muy bajos.

El único modelo que es utilizado en el modelo de cascada y que posee muchos falsos positivos es el Modelo de Longitud de Atributos.

Analizando dicho modelo se notó que el modelo brinda mucho peso a la media de las longitudes calculadas y que los valores extremos, a pesar de ser válidos, tienen asignado una probabilidad muy baja.

Realizando las pruebas nuevamente pero utilizando el Modelo de Longitud modificada se obtuvieron resultados mejores. Entre estos el que obtuvo los mejores resultados es el modelo de cascada 1 sin promedio, con los cuáles se obtienen una tasa de aciertos del 98,86% y una tasa de fallos del 0,16%.

4) Pruebas de tiempo de respuesta: Para evaluar si el detector de intrusiones basado en anomalías puede ser utilizado en un ambiente de producción se realizaron pruebas de rendimiento utilizando Modsecurity como referencia.

Tabla IV
COMPARACIÓN DE RENDIMIENTO DE MODSECURITY VS HTTP-WS-AD

Quartiles	Modsecurity	HTTP-WS-AD
Mínimo	0,769	2,060
Q1(25%)	2,872	3,651
Mediana(50%)	76,639	4,997
Q3(75%)	260,436	39,520
Máximo	1064,25	199,353

Para ello se compararon los tiempos que invierten cada uno de los detectores en realizar la inspección de la petición sin tener en cuenta los tiempos entre el navegador y el servidor web (Apache) en el caso de Modsecurity y los tiempos entre el navegador, el servidor web, PHP y el framework Zend en el caso de HTTP-WS-AD. Para estas pruebas se utilizó una Plataforma con Sistema Operativo GNU/Linux Ubuntu 12.04.1 LTS, servidor Web Apache 2.2.22, PHP 5.3.10, Zend Framework 1.11.11, PostgreSQL 8.4.22, ModSecurity 2.6.3 y ModSecurity CRS 2.2.0-1.

En la tabla IV se puede observar los valores en milisegundos arrojados por nuestro detector y ModSecurity. Estos valores estan dados en quartiles (25%, 50%(mediana) y 75%) además se indican los valores máximo y mínimo obtenidos. Como puede notarse existe una notoria diferencia en tiempo de detección a favor de nuestro detector.

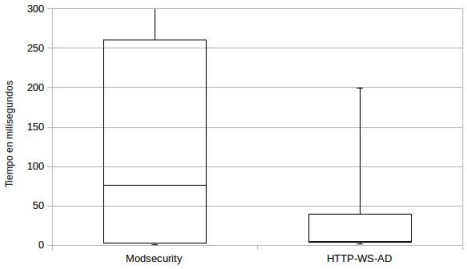

Fig. 4. Gráfico de cajas y bigotes de las pruebas de rendimiento de Modsecurity y HTTP-WS-AD

En la figura 4 se puede visualizar el Gráfico de Cajas de los tiempos obtenidos con el detector de anomalía propuesto y ModSecurity. Se puede notar que el tiempo de detección obtenido por HTTP-WS-AD es mucho menor al tiempo de utilizado por Modsecurity.

VI. Conclusiones

Este artículo presenta un detector de anomalías *online* para aplicaciones web implementado como un proxy reverso. Con esta arquitectura es posible obtener un registro completo de las peticiones tanto GET como POST. También puede evitar que una petición determinada como ataque alcance a una aplicación web que se encuentra detrás del proxy reverso actuando como un WAF.

El detector propuesto esta construido como un framework donde puede fácilmente agregarse otros modelos de anomalía y evaluarlos teniendo todo el mensaje HTTP a disposición. Además permite una fácil administración y control de los modelos y datos registrados en el detector. El detector utiliza una base de datos relacional para registrar las peticiones y los modelos generados. De esta manera los datos son persistidos de manera estructurada, es robusta a cualquier falla del sistema y soporta operaciones concurrentes sobre ella.

Fueron implementados y evaluados los principales modelos de anomalía en aplicaciones web. Se demostró que el Modelo de Inferencia de Estructura planteado por Kruegel requiere de un tiempo de aprendizaje muy superior a la de los otros modelos y que existen otros modelos que realizan una caracterización similar y que poseen un tiempo de aprendizaje lineal, como es el caso del Modelo N-Gram. Comprobamos que el modelo de Distribución de Caracteres detecta a todas las muestras como ataque, inclusive a aquellas que fueron utilizadas en la etapa de entrenamiento.

Según los experimentos un solo modelo no es capaz de detectar todos los ataques, una cuestión que ya fue observada en otros trabajos. Se obtuvo mejor rendimiento con la combinación de varios modelos siendo un aporte importante de este trabajo la forma de seleccionar los modelos adecuados para elevar el nivel de detección y disminuir los falsos positivos. De esta manera obtuvimos un resultado de 98,86% de tasa de acierto utilizando una selección de modelos y un orden determinado. Los experimentos fueron realizados sobre un dataset utilizado por otros trabajos obteniendo nuestro detector resultados auspiciosos.

Con las pruebas de rendimiento se pudo comprobar la factibilidad de utilizar el detector en producción siendo su rendimiento superior a ModSecurity, un detector basado en firmas cuyo uso es bastante popular para la protección de aplicaciones web.

Como trabajo futuro pretendemos realizar más experimentos sobre otras base de datos con diferentes características y realizar pruebas extensivas de los modelo de JSON y XML las que no pudimos realizar por no encontrar una base de datos pública sobre la cual realizar los experimentos.

Referencias

[1] I. L. Stats, "Internet users in the world," http://www.internetlivestats.com/internet-users/, acceso: 4/04/2015.
[2] Symantec, "Internet security threat report," http://www.symantec.com/security_response/publications/threatreport.jsp, Symantec Corporation, Tech. Rep., Apr. 2014, acceso en: 02/02/2015.
[3] A. Kozakevicius, C. Cappo, B. A. Mozzaquatro, R. C. Nunes, and C. E. Schaerer, "Url query string anomaly sensor designed with the bidimensional haar wavelet transform," *International Journal of Information Security*, pp. 1–21, 2015.
[4] (2015) Modsecurity. [Online]. Available: http://www.modsecurity.org
[5] (2015) Http server project. [Online]. Available: http://httpd.apache.org/
[6] M. Heuderich, "PHPIDS, web application security 2.0," http://phpids.org, 2014, access: 01/03/2014.
[7] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of the 10th ACM Conference on Computer and communications security*, ser. CCS '03. New York, NY, USA: ACM, 2003, pp. 251–261.
[8] C. Kruegel, G. Vigna, and W. Robertson, "A multi-model approach to the detection of web-based attacks," *Computer Networks*, vol. 48, pp. 717–738, Aug. 2005.
[9] W. Robertson, G. Vigna, C. Kruegel, R. A. Kemmerer *et al.*, "Using generalization and characterization techniques in the anomaly-based detection of web attacks," in *NDSS*, 2006.
[10] K. Wang and S. Stolfo, "Anomalous payload-based network intrusion detection," in *Recent Advances in Intrusion Detection*, ser. LNCS, E. Jonsson, A. Valdes, and M. Almgren, Eds., vol. 3224. Springer, 2004, pp. 203–222.
[11] K. L. Ingham and H. Inoue, "Comparing anomaly detection techniques for HTTP," in *Proceedings of the 10th international conference on Recent advances in intrusion detection*, ser. RAID'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 42–62.
[12] C. Torrano-Giménez, A. Perez-Villegas, G. Álvarez Marañón *et al.*, "An anomaly-based approach for intrusion detection in web traffic," *Journal of Information Assurance and Security*, vol. 5, pp. 446–454, 2010.
[13] C. T.-G. A. Perez-Villegas and G. Alvarez, "Applying markov chains to web intrusion detection," in *Reunión Española sobre Criptología y Seguridad de la Información*, 2010, pp. 361–366.
[14] H. T. Nguyen, C. Torrano-Gimenez, G. Alvarez, S. Petrović, and K. Franke, "Application of the generic feature selection measure in detection of web attacks," in *Computational Intelligence in Security for Information Systems*. Springer, 2011, pp. 25–32.
[15] C. Kruegel, F. Valeur, and G. Vigna, *Intrusion Detection and Correlation Challenges and Solutions*, 1st ed. Springer-Verlag TELOS, 2005.
[16] https://www.owasp.org/index.php/Web_Application_Firewall.
[17] W. Cavnar and J. Trenkle, "N-gram-based text categorization," in *SDAIR*, 1994, pp. 161–175.
[18] T. Krueger, C. Gehl, K. Rieck, and P. Laskov, "Tokdoc: A self-healing web application firewall," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 1846–1853.
[19] A. Perez-Villegas, C. Torrano-Gimenez, and G. Álvarez, "Http dataset csic 2010," http://www.isi.csic.es/dataset/, 2010.
[20] D. Atienza, l. Herrero, and E. Corchado, "Neural analysis of http traffic for web attack detection," in *International Joint Conference*, ser. Advances in Intelligent Systems and Computing. Springer International Publishing, 2015, vol. 369, pp. 201–212. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-19713-5_18