HTTP-WS-AD: Detector de Anomalías en Aplicaciones Web y Web Services

José Giménez
Ingeniería Informática
Facultad Politécnica
Universidad Nacional de Asunción
Email: jdgimenez@pol.una.py

***Resumen*—Las aplicaciones web se han convertido en los sistemas que más demanda tienen para ser desarrollados en la actualidad. Esto es debido a numerosas ventajas que poseen con respecto a los modelos tradicionales de sistemas.**

Debido a la utilización masiva de los sistemas web, estos se han convertido en uno de los principales objetivos de ciberataques. Existen varios tipos de ataques conocidos, entre los cuales se destacan el ataque XSS (*Cross Site Scripting*) y el ataque SQL Injection, que siguen siendo los más populares debido a la efectividad de sus logros [8].

En este trabajo proponemos HTTP-WS-AD, que evalúa la eficacia de los diferentes modelos de detección de anomalías en la presencia de ataques nuevos como el HPP (*HTTP Parameter Pollution*) o mXSS (*mutated XSS*) e incluye nuevos modelos de anomalías para las peticiones HTTP basadas en los formatos XML o JSON que suelen corresponder a las peticiones en Web Services y AJAX [1] respectivamente.

I. INTRODUCCION

Entre las ventajas que poseen las aplicaciones web con respecto a los modelos tradicionales tenemos: procesamiento paralelo de las peticiones simultáneas de varios clientes; la mayoría de los usuarios están familiarizados con el manejo de los navegadores web, plataforma principal del lado del cliente; la lógica de negocios se ejecuta en el servidor web, haciendo que la actualización o corrección de esta solo se aplique en este lugar y los clientes automáticamente ven dicha actualización; todos los sistemas operativos tienen un navegador web por defecto, haciendo que no sea necesario la instalación de un software especial para el cliente; no se necesitan un gran poder de cómputo en los clientes, ya que el servidor web realiza toda la lógica de negocios. El navegador web se encarga principalmente de la interfaz gráfica presentada al usuario y del envío de las peticiones generadas por el usuario al servidor web.

Las ventajas de las aplicaciones web han causado que se propaguen rápidamente en todas las organizaciones y entidades. En este escenario, proteger estas aplicaciones de los ataques es un tema crítico. Para mitigar estas amenazas es necesario crear infraestructuras y aplicaciones seguras. En este sentido, los sistemas de detección de ataques para aplicaciones web (WAF - *Web Application Firewall*) son parte de la infraestructura de seguridad. Estos sistemas tienen la ventaja de que son independientes de los sistemas que protegen, y pueden proteger a los sistemas que son de código cerrado.

[1] AJAX: Asynchronous JavaScript And XML

Para detectar los ataques a las aplicaciones web existen 2 métodos:

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: posee una tasa de detección muy alta; su tasa de falsos positivos es muy baja; puede identificar el tipo de ataque de manera certera. Sus desventajas son: no es muy eficaz a la hora de detectar ataques nuevos o no conocidos; la generación de las firmas requieren una gran cantidad de esfuerzo y tiempo. Tienen que ser generadas por personas que tienen una vasta experiencia y dominio del tema. Es por esto que no existen herramientas que generen automáticamente estas firmas; su base de datos debe de ser actualizada frecuentemente para considerar los nuevos ataques o variaciones de los conocidos, lo que implica una tarea administrativa alta; 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: puede detectar ataques nuevos y variaciones de los conocidos; observa el comportamiento del sistema y genera automáticamente el modelo de comportamiento "normal" sin intervención humana y de manera automática; 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 principales desventajas son: posee una alta tasa de falsos positivos; se necesita realimentar los datos de normalidad cuando el sistema sufre algun cambio, de forma que el nuevo compartamiento no sea considerado como una anomalía; la identificación del tipo de ataque se hace más difícil con esta estrategia [7].

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 área de detección de anomalías se ha convertido en un método prometedor para identificar ataques y tratar de evitarlos. El enfoque anomalía 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 considerando nuevos tipos de ataques web (por ejemplo, el ataque HPP [1], el ataque mXSS[2], etc). Se implementan y analizan los modelos presentados en [5] y [6]. También el modelo basado en el análisis de *ngram's* presentado en [4] y el modelo de anomalía propuesto en [9]. 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 [2] (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 III se presentan los detalles del trabajo: los modelos implementados, el generador de modelos de anomalías y la arquitectura del detector/evaluador de anomalías.

A continucación se procede a explicar los modelos implementados.

II. Modelos Implementados

Se implementaron varios modelos que pueden ser categorizados en:

1. **Modelos de Parámetros:** representan el comportamiento de un parámetro a la vez. Entre los modelos en esta categoría se encuentran:
   - *a*) Modelos de Kruegel[5] [6]:
     1) Modelo de Token.
     2) Modelo de Longitud.
     3) Modelo de Distribución de Carácteres.
     4) Modelo de Inferencia de Estructura.
   - *b*) Modelos de Ingham[4]:
     1) Modelo de N-Gram.
     2) Modelo de Distancia de Mahalanobis[9].
2. **Modelos de "Query String":** representan la estructura de todos los parámetros que recibe un programa en una URL en particular. Se modelan que conjunto de parámetros son recibidos en esta URL y en que orden son recibidos. Entre los modelos en esta categoría se encuentran:
   - *a*) Modelos de Kruegel[5] [6]:
     1) Modelo de Prescencia o Ausencia de Atributos.
     2) Modelo de Orden de Atributos.
   - *b*) Viendo el auge que poseen la arquitectura REST en las aplicaciones WEB se implementaron 2 modelos nuevos:
     1) Modelo de JSON.
     2) Modelo de XML.
3. **Modelos de Sessión:** representan la interacción del usuario y los diferentes programas que existen en una aplicación web. Entre los modelos en esta categoría se encuentran:
   - *a*) Modelos de Kruegel[6]:
     1) Modelo de Frecuencia de Acceso.
     2) Modelo de Orden de Invocación.

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

A continuación se explican en detalle cada uno de los modelos.

II-A. Modelos de Parámetros

1) Modelo de 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 2 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 2 funciones se calcula la correlación que existen entre ellas, utilizando esta medida para determinar si las muestras fueron repeticiones o valores nuevos.

$$\rho = \frac{covar(f,g)}{\sqrt{var(f) * var(g)}} \tag{3}$$

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 rechaza.

2) Modelo de 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 muestra y se utiliza la inecuación de Chesbychev para calcular la distancia entre la longitud de una cadena y la media. La inecuación de Chesbychev coloca un límite superior en la probabilidad que la diferencia entre el valor de una variable x y μ exceda a un límite t.

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

En esta ecuación se reemplaza el valor de t por la diferencia entre la longitud de un valor l y la media, 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.

$$p(|x-\mu| > |l-\mu|) < p(l) = \frac{\sigma^2}{(l-\mu)^2} \tag{5}$$

3) Modelo de Distribución de Carácteres: se basa en la observación que los datos ingresados en los campos de los formularios son palabras pertenecientes a la lengua utilizada y por ende los carácteres que componen dichas palabras son "carácteres imprimibles", siendo la mayoría de ellos carácteres alfnuméricos y la minoría carácteres especiales.

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

Sea cuál sea la lengua utilizada existen carácteres cuya frecuencia es superior a las otras, pero si llega a calcularse la Distribución de Carácteres se notará que el descenso en la frecuencia de dichos carácteres se realiza de manera gradual.

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

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

Para determinar si la distribución de carácteres de un valor observado es una muestra perteneciente del ICD se utiliza el test χ^2 de Pearson, usando la confianza devuelta como la probabilidad de que la distribución de carácteres de un valor observado es una muestra perteneciente del ICD.

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

El test χ^2 se realiza de la siguiente manera:

1. 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 intérvalo. 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 intérvalos determinados para el ICD por la longitud de la cadena.
2. Calcular el valor de χ^2 de la siguiente manera: $\chi^2 = \sum_{i=0}^{i<6}(O_i - E_i)^2/E_i$
3. 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 intérvalos. La probabilidad que la muestra pertenezca a la distribución de carácteres ideal se obtiene de la tabla utilizando el valor χ^2 como índice.

4) Modelo de 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 Finítos no Determinístas (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 $p(t_i)$.

$$p(w) = \prod_{\text{(caminos p en w)}} \sum_{\text{(estados}\in\text{p)}} p_{s_i}(o_i) * p(t_i) \qquad (6)$$

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:

$$p(Modelo/DatosEntrenamiento) = \frac{p(DatosEntrenamiento/Modelo) * p(Modelo)}{p(DatosEntrenamiento)} \qquad (7)$$

La probabilidad de los datos de entrenamiento es considerado un factor de scala 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:

$$p(Model) = \frac{1}{\prod_{S\in Estados}(N+1)^{\sum_s transiciones} * (N+1)^{\sum_s emisiones}} \qquad (8)$$

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 multiplcación 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) Modelo de N-Gram: Un N-gram es una subcadena generada por una ventana deslizante de tamaño N a través de una cadena de carácteres. El resultado es un conjunto de carácteres 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 a calcular la cantidad de N-grams de esa cadena que se tienen registrado contra la cantidad total de N-Grams que posee la cadena:

$$p = \frac{\begin{array}{c}\text{\# de n-grams en el valor testeado que} \\ \text{también están en los datos de entrenamiento}\end{array}}{\text{\# de n-grams en el valor testeado}} \quad (9)$$

Dependiendo de la elección de N el modelo de N-Gram puede modelar desde carácteres 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) Modelo de Distancia de Mahalanobis: La distancia de Mahalanobis es una métrica de distancia estandar para comparar dos distribuciones estadísticas.

La distancia de Mahalanobis es pesado de calcular. Wang y Stolfo propusieron una distancia de Mahalanobis simplificada:

$$d(x, \bar{y}) = \sum_{i=0}^{n-1} \frac{(|x_i - \bar{y}_i|)}{\bar{\sigma} + \alpha} \quad (10)$$

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 estandar.

Wang y Stolfo también 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.

II-B. Modelos de Query String

1) Modelo de Prescencia 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.

2) Modelo de 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 siguiente fórmula:

$$O = \{(a_i, a_j) : a_i \text{ precede } a_j \text{ and } a_i, a_j \in (S_{q_j} : \forall j = 1, .., n)\} \quad (11)$$

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 precendencia 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.

El grafo generado es acíclico y se puede utilizar para controlar el orden de 2 atributos.

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

3) Modelo de JSON/XML: tienen un objetivo similar a los Modelos de Prescencia 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 Prescencia 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 la rechaza.

Similarmente se realiza esto para los mensajes que poseen el formato JSON.

II-C. Modelos de Sessión

1) Modelo de 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.

Kruegel 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.

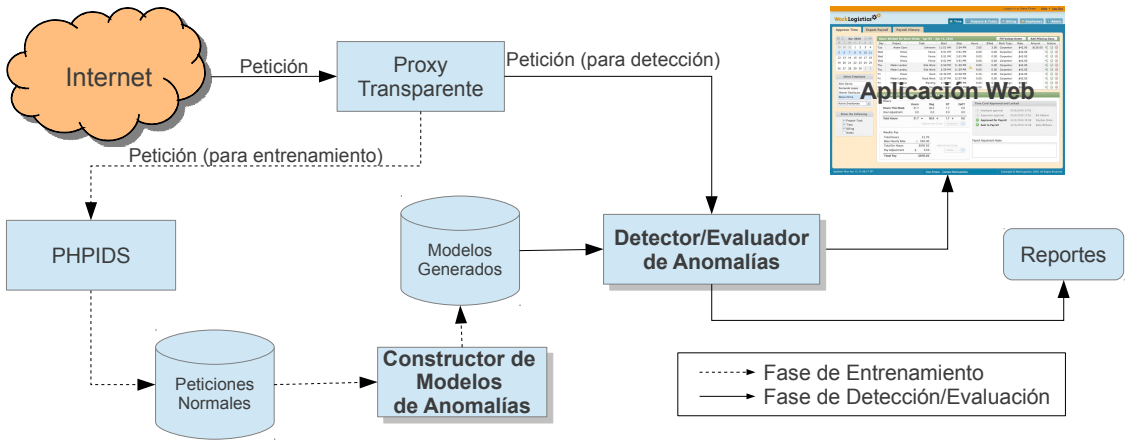

Figura 1. Arquitectura de HTTP-WS-AD

Para esto se dividen las peticiones registradas en intérvalos de 10 segundos. Dentro de cada uno de estos intérvalos 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 partcular en el mismo intérvalo utilizado en la etapa de aprendizaje. Con esto se calculan las probabilidades de Chesbychev para estas 2 distribuciones y se promedian.

2) Modelo de Orden de Invocación: modela la secuencia de URL que son accecidas por los clientes al utilizar un sistema. Para ello utiliza el valor de un Cookie configurado de antemano. Krugel 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.

III. Trabajo Propuesto

HTTP-WS-AD se basa en un proxy transparente que 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. El proxy transparente 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 1 muestra la arquitectura de nuestro detector de anomalía y evaluador. Las líneas punteadas indican la fase de entrenamiento o aprendizaje. En esta fase las peticiones son capturadas por el proxy transparente. Luego estas peticiones son analizadas por un WAF basado en firmas llamado PHPIDS [3]. 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 transparente 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. Para evaluar esta eficacia se incluyen peticiones HTTP sintéticos con ataques nuevos.

IV. Metodología y Estado Actual del Trabajo

Los objetivos del trabajo son los siguientes:

1. Explorar sobre los modelos de anomalías en tráfico web existentes para caracterizar los diferentes aspectos de los parámetros HTTP, las peticiones HTTP y el comportamiento de un usuario normal del sistema.
2. Implementar una plataforma que permita evaluar los modelos seleccionados sobre ataques actuales.
3. Caracterizar los ataques actuales a las aplicaciones web evaluando la eficacia de detectarlos con los modelos de anomalía implementados.
4. Proponer nuevos modelos adecuados para ataques incrustados en mensajes XML y JSON.

Se obtuvo una muestra de las peticiones de un servidor y se evaluaron los parámetros con PHPIDS. Actualmente se está generando los modelos para dichos parámetros para evaluar la eficacia de los modelos y compararlos con los resultados obtenidos con PHPIDS.

REFERENCIAS

[1] OWASP AppSec Europe 2009. HTTP parameter pollution. http://www.owasp.org/images/b/ba/AppsecEU09_CarettoniDiPaola_v0.8.pdf, May 2009.

[2] Mario Heiderich, Jörg Schwenk, Tilman Frosch, Jonas Magazinius, and Edward Z Yang. mXSS attacks: attacking well-secured web-applications by using innerhtml mutations. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 777–788. ACM, 2013.

[3] M. Heuderich. PHPIDS, web application security 2.0. http://phpids.org, 2014. Access: 01/03/2014.

[4] Kenneth L. Ingham and Hajime Inoue. Comparing anomaly detection techniques for HTTP. In *Proceedings of the 10th international conference on Recent advances in intrusion detection*, RAID'07, pages 42–62, Berlin, Heidelberg, 2007. Springer-Verlag.

[5] Christopher Kruegel and Giovanni Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM Conference on Computer and communications security*, CCS '03, pages 251–261, New York, NY, USA, 2003. ACM.

[6] Christopher Kruegel, Giovanni Vigna, and William Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks*, 48:717–738, Aug. 2005.

[7] W. K. Robertson. *Detecting and Preventing Attacks Against Web Applications*. PhD thesis, University of California, Santa Barbara, 2009.

[8] Symantec. Internet security threat report. Technical report, Symantec Corporation, Apr. 2013. Acesso em: 02/06/2013.

[9] K. Wang and S.J. Stolfo. Anomalous payload-based network intrusion detection. In Erland Jonsson, Alfonso Valdes, and Magnus Almgren, editors, *Recent Advances in Intrusion Detection*, volume 3224 of *LNCS*, pages 203–222. Springer, 2004.