

Apéndice

Figuras

Figura 1.1: Ley de Moore para microprocesadores y chips de memoria. La curva morada es la proyección de la Ley de Moore basada en los datos recopilados hasta el año 1975. Notar la corrección existente alrededor de 1980. Fuente: Intel Corporation.

Figura 1.2: Tasa de transferencia (throughput) y consumo de memoria (memory) del proceso grep frente al número de patrones usados. La longitud media de los patrones es de 29 caracteres. Fuente: Iulian Moraru and David G. Andersen. Fast Cache for Your Text: Accelerating Exact Pattern Matching with Feed-Forward Bloom Filters.

Figura 1.3: Proceso de búsqueda de patrones en Snort.

Figura 1.4: sigMatch como filtro previo al MPSE de Snort.

Figura 1.5: Fases del proyecto.

Figura 2.1: Ejemplo de máquina de estados

Figura 2.2: Autómata finito no determinista (izquierda) y determinista (derecha) usando los patrones 'halo', 'alto' y 'ala'. A L M I H A L A.

Figura 2.3: Ejemplo de aplicación de una función hash que devuelve, a partir del diccionario de la izquierda, un conjunto de resúmenes. Su rango de salida va desde 0x0000 a 0xFFFF. Se muestra una colisión entre 'alto' y 'tono'.

Figura 2.4: Fase offline, o creación de la estructura de búsqueda, de un filtro Bloom. En fondo amarillo se representa la colisión producida.

Figura 2.5: Fase online, o búsqueda de elementos, de un filtro Bloom. Los patrones 'alto' y 'meta' atraviesan el filtro (positivos), aunque 'meta' no forme parte del diccionario. El patrón 'bajo' es descartado (negativo).

Figura 3.1: Estructura de detección de sigMatch: sigTree. $b=2$.

Figura 3.2: Filtrado basado en prefijo. En este caso la cardinalidad de P es igual a la de F .

Figura 3.3: Filtrado basado en 2-gram. En este caso la cardinalidad de R es menor a la de F .

Figura 3.4: Esquema general correspondiente a un proceso de búsqueda de patrones.

Apéndice

Figura 3.5: Esquema general correspondiente a un proceso de búsqueda de patrones que implemente el filtro sigMatch antes de invocar a la Unidad de Verificación.

Figura 3.6: Obtención de los resúmenes R en función de las firmas originales F (fase offline).

Figura 3.7: Creación de la estructura sigTree a partir de los resúmenes R y las firmas originales F (fase offline).

Figura 3.8: Representación del paso de datos por la estructura sigTree (fase online).

Figura 3.9 Tiempos de ejecución y factores de mejora de los sistemas A y B.

Figura 3.10: Pruebas con ClamAV usando cuatro conjuntos de 100MB de diferentes tipos de ficheros. 89.903 firmas, $b=2$, $\beta=6$, tamaño del filtro Bloom = 2^{14} bits. Fuente sigMatch.

Figura 3.11: Pruebas con Bro usando una traza TCP. 1.200 firmas de Snort, $b=2$, $\beta=3$, tamaño del filtro Bloom = 2^{14} bits. Fuente sigMatch.

Figura 3.12: Pruebas con ClamAV usando una colección de 100MB de ficheros de tipo .exe y cuatro base de datos de firmas de diferentes tamaños. $b=2$, $\beta=6$, tamaño del filtro Bloom = 2^{14} bits. Fuente sigMatch.

Figura 4.1: Modelo OSI y captura de paquetes por parte de tcpdump a través de las librerías libpcap.

Figura 4.2: DAQ y Componentes de Snort. Flujo de datos desde la capa de enlace de datos hasta la salida de Snort.

Figura 4.3: Esquema representativo del camino seguido por un paquete en su descodificación.

Figura 4.4: Funcionamiento del preprocesador de Snort.

Figura 4.5: Ejemplo que demuestra la utilidad del componente preprocesador de Snort.

Figura 4.6: Esquema que muestra la utilidad del preprocesador Frag3.

Figura 4.7: Fin de conexión en el protocolo TCP. Fuente: Wikipedia.org.

Figura 4.8: Funcionamiento del motor de detección.

Figura 4.9: Esquema de lista enlazada en tres dimensiones.

Figura 4.10: Esquema de lista enlazada de reglas.

Figura 4.11: Diagramas de secuencia de las fases offline (izquierda) y online (derecha) llevadas a cabo en Snort.

Figura 4.12: Diagrama de secuencia de la fase offline de Snort.

Figura 4.13: Esquema representativo del camino seguido desde la función fpCreateFastPacketDetection hasta el motor de búsqueda multi-patrón.

Apéndice

Figura 4.14: Diagrama de secuencia de la fase online de Snort.

Figura 4.15: Esquema de la función Preprocess.

Figura 4.16: Esquema representativo del camino seguido desde la función fpEvalPacket hasta el motor de búsqueda multi-patrón (MPSE).

Figura 5.1: Implementación del filtro entre la salida del descodificador y la entrada del preprocesador.

Figura 5.2: Ejemplo en el que se coloca el filtro entre el descodificador y el preprocesador.

Figura 5.3: Implementación del filtro antes de la salida del preprocesador, considerándolo así como un módulo más del preprocesador.

Figura 5.4: Implementación del filtro como último módulo del preprocesador.

Figura 5.5: Implementación del filtro antes de cada llamada al motor de detección.

Figura 5.6: Implementación del filtro al principio del motor de detección.

Figura 5.7: Lista enlazada de reglas sin la opción split-any-any.

Figura 5.8: Lista enlazada de reglas construida con la opción split-any-any.

Figura 5.9: Diagrama de secuencia de la fase offline del motor de detección.

Figura 5.10: Diagrama de secuencia de la fase online del motor de detección.

Figura 5.11: Diagrama de secuencia de la fase offline del motor de detección modificado para implementar el filtrado previo.

Figura 5.12: Diagrama de secuencia de la fase online del motor de detección modificado para implementar el filtrado previo.

Figura 6.1: Diagrama de secuencia de la función redbCompile, correspondiente a la fase offline del filtrado previo introducido en el motor de detección de Snort.

Figura 6.2: Funcionamiento de la búsqueda dentro de la estructura sigTree.

Figura 6.3: Ejemplo de búsqueda en la estructura sigTree.

Figura 6.4: Supuesto práctico relacionado con la tercera implementación.

Figura 7.1: Operaciones realizadas en sigMatch.

Figura 7.2: Ejemplo de estimación de probabilidades. a) planteamiento del NFA; b) estimación basada en el número de nodos; c) y d) estimaciones basadas en distintas distribuciones de probabilidad de las bases nitrogenadas.

Gráficas

Gráfica 1.1: Evolución anual de la capacidad de las redes Ethernet (escala natural) y estimación en el año 2020, basado en el cálculo de la curva de regresión exponencial entre 1980 y 2010.

Gráfica 1.2: Evolución anual de la capacidad de las redes Ethernet (escala semi-logarítmica) y estimación en el año 2020, basado en el cálculo de la curva de regresión exponencial entre 1980 y 2010.

Gráfica 1.3: Evolución temporal del número de reglas. Recta de regresión evaluada desde 1999 hasta 2005. Recta de regresión evaluada desde 2005 hasta 2006 más el dato actual de número de reglas. Previsión de número de reglas en 2020 teniendo en cuenta la evolución desde 2005.

Gráfica 1.4: Relación, en escala semi-logarítmica, entre el número de patrones y la memoria usada (bytes) por una estructura DFA.

Gráfica 1.5: Estimación de la evolución anual del rendimiento de Snort (escala natural).

Gráfica 1.6: Estimación de la evolución anual del rendimiento de Snort (escala semi-logarítmica).

Gráfica 2.1: Comparativa del tiempo de ejecución de los algoritmos Aho-Corasick (AC) y Rabin-Karp (RK). En este último se muestran el mejor y el peor de los casos.

Gráfica 6.1: Distribución de firmas en función de la longitud.

Gráfica 6.2: Valores de los resúmenes correspondientes a la función RS.

Gráfica 6.3: Distribución de frecuencia de los resúmenes correspondientes a la función RS.

Gráfica 6.4: Valores de los resúmenes correspondientes a la función XOR.

Gráfica 6.5: Distribución de frecuencia de los resúmenes correspondientes a la función XOR.

Gráfica 6.6: Valores de los resúmenes correspondientes a la función SAX.

Gráfica 6.7: Distribución de frecuencia de los resúmenes correspondientes a la función SAX.

Gráfica 6.8: Valores de los resúmenes correspondientes a la función SDBM.

Gráfica 6.9: Distribución de frecuencia de los resúmenes correspondientes a la función SDBM.

Gráfica 6.10: Valores de los resúmenes correspondientes a la función AM.

Apéndice

Gráfica 6.11: Valores de los resúmenes correspondientes a la función AA.

Gráfica 6.12: Distribución de frecuencia de los resúmenes correspondientes a las funciones AM y AA.

Gráfica 6.13: Comparativa entre las diferentes funciones hash en cuanto a eficiencia.

Gráfica 6.14: Coeficientes de variación de las distintas funciones hash (con y sin dimensionamiento).

Gráfica 7.1: Representación del número de firmas que cuentan con una longitud determinada.

Gráfica 7.2: Distribución de patrones en las instancias creadas por Snort.

Gráfica 7.3: Distribución de patrones y del tráfico analizado por cada instancia creada en Snort.

Gráfica 7.4: Distribución del consumo memoria de las estructuras de búsqueda de Aho-Corasick y sigMatch por cada instancia creada por Snort.

Gráfica 7.5: Distribución del consumo memoria de las estructuras de búsqueda de Aho-Corasick y sigMatch por cada instancia creada por Snort (desde la undécima instancia en adelante).

Gráfica 7.6: Distribución del consumo memoria de las estructuras de búsqueda de Aho-Corasick y sigMatch por cada instancia creada por Snort (instancias ordenadas en función de la cantidad de tráfico que soportan).

Gráfica 7.7: Rango de consumo computacional de la implementación de los algoritmos Aho-Corasick y sigMatch: valores máximos y mínimos de cada uno.

Gráfica 7.8: Rango de consumo computacional de la implementación de los algoritmos Aho-Corasick y sigMatch: valores máximos y mínimos de cada uno. Consumo de sigMatch desglosado.

Gráfica 7.9: Rango de consumo computacional de la implementación de los algoritmos Aho-Corasick y sigMatch (sólo NFA de altura b): valores máximos y mínimos de cada uno.

Gráfica 7.10: Consumo computacional promedio de los algoritmos Aho-Corasick y sigMatch, usando probabilidades estimadas, teniendo en cuenta únicamente una base de firmas concreta.

Gráfica 7.11: Estudio del parámetro β .

Gráfica 7.12: Estudio de las funciones hash.

Gráfica 7.13: Estudio del tamaño de la matriz del filtro Bloom.

Gráfica 7.14: Estudio del tamaño de la matriz del filtro Bloom (segunda parte).

Gráfica 7.15: Estudio final.

Apéndice

Gráfica 7.16: Rendimiento: 1GB de tráfico benigno.

Gráfica 7.17: Rendimiento: 503MB de tráfico benigno.

Gráfica 7.18: Rendimiento: 4MB de tráfico agresivo.

Gráfica 7.19: Rendimiento: 76MB de tráfico agresivo.

Gráfica 7.20: Rendimiento: 612MB de tráfico agresivo.

Gráfica 7.21: Comparativa de las tasas de candidatos y de ocurrencias con los factores de mejora usando las cinco capturas de red disponibles hasta ahora. Línea azul discontinua: tasa de candidatos, Línea roja discontinua: tasa de ocurrencias. Línea verde continua: factor de mejora.

Gráfica 7.22: Relación obtenida entre tasa de candidatos y factor de mejora.

Gráfica 7.23: Consumo computacional promedio de los algoritmos Aho-Corasick y sigMatch, usando probabilidades calculadas teniendo en cuenta el entorno.

Gráfica 7.24: Tiempo ponderado empleado por los motores de búsqueda, usando y sin usar el filtrado.

Tablas

Tabla 1.1: Factores para el cálculo de la predicción del rendimiento de Snort

Tabla 1.2: Resultados de la evolución estimada anual del rendimiento de Snort.

Tabla 3.1: Datos de la memoria usada por sigTree con y sin filtros Bloom (BF), número de cache misses de nivel L1 y L2 y factor de velocidad (speedup). Resultados obtenidos usando, para ClamAV los parámetros $b=2$, $\beta=6$ y tamaño de BF = 2^{14} bits y 100MB de ficheros de tipo .exe, para Bro los parámetros $b=2$, $\beta=3$ y tamaño de BF = 2^{14} bits y un archivo que contiene un escaneo TCP realizado el 06/03/98 y para DBLIFE los parámetros $b=2$, $\beta=4$ y tamaño de BF = 2^{14} bits y un escaneo de una colección de 100MB de páginas web. Fuente sigMatch.

Tabla 3.2: Datos de la memoria usada por ClamAV y por sigMatch, número de cache misses de nivel L1 y L2, tasa de filtrado (filter rate) y factor de velocidad (speedup). Resultados obtenidos para cuatro tamaños diferentes de bases de datos de firmas, usando ClamAV, los parámetros $b=2$, $\beta=6$ y tamaño de BF = 2^{14} bits y 100MB de ficheros de tipo .exe. Fuente sigMatch.

Tabla 7.1: Instrucciones de lenguaje ensamblador implicadas en el método de cálculo teórico de rendimiento: Descripción.

Tabla 7.2: Instrucciones de lenguaje ensamblador implicadas en el método de cálculo teórico de rendimiento: Valores de CPI, Latencia y Latencia Adicional.

Tabla 7.3: Valores de tasa de candidatos, tasa de ocurrencias y factor de mejora para los cuatro grupos de reglas disponibles y las cinco capturas de tráfico.

Tabla 7.4: Probabilidades para el cálculo del promedio del coste computacional.