El Spyware como amenaza contra navegadores web

Sergio Castillo-Pérez*, José Alfredo Múrcia Andrés‡, Joaquin Garcia-Alfaro†‡

* Universitat Autònoma de Barcelona, Edifici Q, 08193, Bellaterra
† Institut Telecom, Telecom Bretagne, 35576, Cesson-Sevigne, France
‡ Universitat Oberta de Catalunya, Rambla de Poblenou 156, 08018, Barcelona

***Resumen*—En la última década se ha realizado un progreso sustancial en Internet y en las tecnologías basadas en el paradigma web. Aplicaciones relacionadas con educación, salud, banca, o incluso con relaciones entre individuos o grupos sociales, pueden beneficiarse mediante el uso de dichas tecnologías. Sin embargo, los ataques a sistemas pueden comprometer drásticamente la privacidad de los usuarios que hacen uso de las tecnologías web. En este contexto, la infección de sistemas mediante Spyware es un claro ejemplo. En este artículo analizamos la amenaza del Spyware como vía para comprometer la seguridad y privacidad de los recursos de los navegadores web.**

I. Introducción

El uso del paradigma web en todos los modelos de negocios y organizaciones está convirtiéndose en un aspecto omnipresente. De hecho, su uso aparece como una estrategia emergente en todos los tipos de aplicaciones software de las compañías [1]. Éste permite el diseño de aplicaciones totalmente interactivas que pueden potencialmente ser usadas por miles de usuarios alrededor del mundo. La existencia de nuevas tecnologías para mejorar las características del paradigma web tradicional permite a los ingenieros del software concebir nuevos servicios, que no están restringidos a un sistema operativo específico. Sistemas tradicionales de información relacionados con la educación, salud, banca o incluso sistemas de emergencia, pueden beneficiarse de esta tecnología.

La complejidad actual del paradigma web tiene, sin embargo, un impacto en la seguridad de los navegadores web y, de forma más precisa, en el tratamiento de sus recursos. Los ataques contra navegadores web pueden comprometer la seguridad y la privacidad de los usuarios. Esto puede tener serias consecuencias dada la omnipresencia de software malicioso, como el Spyware. El Spyware puede ser instalado de forma secreta en los navegadores web y robar datos sensibles, tales como identificaciones de usuarios, contraseñas o datos financieros [7]. Los navegadores web deben, por tanto, incluir mecanismos confiables para garantizar la seguridad y privacidad de sus usuarios. En este artículo damos una visión general de algunas técnicas usadas por el Spyware, y que pueden ser usadas por entidades maliciosas para violar la privacidad de los usuarios. Presentamos un escenario que muestra cómo la privacidad de un usuario accediendo a un servicio web puede ser violada por Spyware vinculado al navegador web. Seguidamente discutimos algunos mecanismos de defensa que pueden reducir el riesgo representado por la amenaza del Spyware.

Organización del artículo — La sección II presenta la amenaza del Spyware y desarrolla nuestro escenario de motivación. La sección III presenta una visión general sobre mecanismos de defensa para reducir el riesgo de la amenaza del Spyware. La sección IV concluye el artículo.

II. Spyware

El concepto Spyware (o software espía) es un término utilizado para catalogar al software malicioso (*Malware*) [11] que registra información de usuarios de forma no consentida, violando la privacidad de éstos. La información recolectada por este tipo de aplicaciones suele ser de distinta índole, tal como datos personales, números de tarjeta de crédito, hábitos de navegación web, contraseñas, pulsaciones de teclas, o captura de pantalla. Tal información es transmitida a terceras partes con finalidades como son el fraude electrónico [6], el marketing a través de publicidad web no consentida, u otras actividades normalmente maliciosas. Con la finalidad de conseguir su propósito, dicho software provoca diversos efectos en el comportamiento del sistema afectado, como la aparición de ventanas de navegación con publicidad no deseada, el secuestro del navegador web, instalación de puertas traseras (*backdoors*), modificación de los números de conexión a ISPs a otros con tarifas elevadas, etc. Asimismo, y con motivo de llevar a cabo su finalidad, la ejecución de estas aplicaciones suele conllevar la degradación del rendimiento del sistema, incrementando el uso de CPU, del espacio utilizado en disco, o del ancho de banda de red.

A lo largo del tiempo, el Spyware ha evolucionado incorporando sofisticados mecanismos propios de los rootkits. Estos mecanismos permiten al Spyware esconder su presencia a administradores o a software destinado a su detección. Así, estrategias como la ocultación de procesos, archivos o conexiones de red, o el uso de técnicas antidebugging o de criptografía, suelen ser características habituales en el Spyware de hoy en día. De forma análoga, y con el objetivo de no ser eliminados del sistema infectado, la inclusión de estrategias que dificultan su desinstalación son propiedades comunes en este tipo de software. Este conjunto de metodologías evasivas, junto a los mecanismos para la recopilación de información en si, suele conllevar la utilización de técnicas de programación que, en ocasiones, provocan cierta inestabilidad del sistema, dando lugar a un comportamiento inesperado de algunas aplicaciones.

En ocasiones, el concepto de Spyware es confundido con el de virus o gusano. A pesar de que ambos comparten una finalidad maliciosa, son radicalmente distintos en su modo de funcionamiento. Una de las diferencias primordiales que los distingue radica en el hecho de que el Spyware no suele auto-replicarse, es decir, un sistema afectado no propagará la infección a otros ordenadores. En el proceso de infección del Spyware podemos distinguir básicamente dos tipos de estrategia en función de si se requiere una interacción directa por parte del usuario, o si, por contra, la instalación se realiza de forma transparente.

En el primer caso, el Spyware suele venir camuflado en programas supuestamente legítimos y que, sin el consentimiento del usuario, es instalado junto a dicho software. Así, programas de tipo P2P, complementos para los navegadores web, software descargado de forma ilegal, o cracks, son ejemplos de programas que pueden esconder Spyware. En otras ocasiones, la vía de infección se basa en el acceso a una determinada web con componentes ActiveX o Applets Java especialmente preparados que, tras la autorización de su ejecución por parte del usuario, conllevan la instalación del Spyware. En cualquier caso, el proceso de infección requiere una aprobación de ejecución tácita por parte del usuario. La utilización de la ingeniería social suele estar presente en esta metodología, pudiéndose considerarse incluso un factor decisivo en el consecución del éxito para los atacantes. De esta manera, los atacantes utilizan estrategias para persuadir a los usuarios a descargar determinado software, o realizar determinadas acciones que conlleven la instalación del Spyware.

En el segundo caso, los mecanismos de infección del Spyware se basan en la explotación de vulnerabilidades en el software. De esta manera, la visita a una determinada web preparada haciendo uso de un navegador vulnerable, o la apertura de un archivo especialmente preparado mediante software que incluya deficiencias de seguridad, pueden provocar la ejecución de código arbitrario que conduzca a la instalación del Spyware. En este proceso de infección, la instalación del software espía suele pasar totalmente desapercibida para el usuario afectado, sin requerirse ningún tipo de acción a realizar que puede considerarse motivo de sospecha. Comúnmente, esta estrategia suele estar ligada a otros ataques previos como son la redirección mediante XSS o el DNS Spoofing entre otros. Una vez el proceso de infección ha tenido éxito, el software espía capturará la información de su interés, y la utilizará según la finalidad para la que fue programado. En base a los mecanismos que se usan para interceptar esta información, y de las estrategias utilizadas para evadir la detección y/o desinstalación, distinguimos entre Spyware en modo usuario y Spyware en modo de kernel.

Spyware en modo usuario — A esta categoría pertenece la mayor parte del Spyware que encontramos hoy en día, dado que su programación es más sencilla. Adicionalmente, dado que se ejecutan en modo de usuario, su detección y eliminación desentraña menos complejidad. Las estrategias empleadas en esta categoría para interceptar información, o los mecanismos para evadir su detección/desinstalación se basan principalmente en desviar el flujo de ejecución de las aplicaciones, cediendo el control a determinado código del Spyware. Para esto, existen básicamente dos técnicas. La primera se basa en modificar las direcciones de memoria donde son mapeadas las funciones de las librerías compartidas y utilizadas por las aplicaciones. En el segundo caso, la metodología se sustenta en aprovechar mecanismos de extensión que proporcionan determinados programas. Estos mecanismos permiten ceder el control a cierto código al producirse eventos asociados a dicha aplicación, permitiendo extender las funcionalidades del software de forma sencilla. Esta idea, que dota de flexibilidad a los programas, puede ser usada de forma maliciosa por parte del Spyware.

Spyware en modo de kernel — El Spyware en modo de kernel es conceptualmente igual que el Spyware en modo usuario, en el sentido que su funcionamiento se basa en desviar el flujo de ejecución de las aplicaciones. De nuevo, la ejecución es desviada a código del Spyware, el cual captura u oculta información en función de sus necesidades. La diferencia principal entre las dos categorías se basa en que en esta segunda el proceso de intercepción de información o evasión se realiza con un mayor nivel de privilegios. Concretamente, el código del software espía es cargado y ejecutado en el espacio de direcciones del núcleo del sistema operativo. Precisamente, esta característica de ejecución con privilegios de sistema les proporciona una mayor resistencia a ser detectados o eliminados, ya que las herramientas de detección o eliminación suelen ejecutarse en espacio de usuario. Esto implica una mayor complejidad en su código, lo que los hace menos comunes. La forma más habitual de desviar el flujo de ejecución por este tipo de Spyware se basa en modificar las tablas que contienen las direcciones asociadas a las llamadas al núcleo (*syscalls*).

II-A. Ejemplo de infección de un navegador web

Con el objetivo de ilustrar cómo la privacidad de un usuario puede ser vulnerada mediante Spyware, describimos en esta sección un hipotético escenario donde se pone de manifiesto este hecho. La técnica utilizada por el Spyware para interceptar la información se basará en explotar los mecanismos de extensión de ciertos programas — en nuestro caso, del navegador web. Esta característica está presente en la mayoría de los navegadores, y permite facilitar la adición de mejoras – comúnmente llamadas *extensiones* o *complementos* – por terceros programadores. Para conseguir esto, la rutina principal del navegador delega el flujo de ejecución a las nuevas funciones de las extensiones ante ciertos eventos, permitiéndoles la lectura y la modificación del DOM (*Document Object Model*). Así, por ejemplo, se puede ceder el control del navegador durante la carga de una web, la carga/descarga de archivos, etc. De la misma manera que las extensiones legítimas, esta tecnología puede ser explotada por el Spyware para introducir sus rutinas de captura de información.

Supongamos que un usuario malicioso desea obtener números de tarjetas de crédito válidos con el objetivo de perpetrar fraude electrónico. Asumamos también que dicho usuario malicioso descubre una vulnerabilidad asociada a una versión particular de un navegador web que permite la ejecución de código arbitrario. Analizando la vulnerabilidad, el usuario malintencionado podría preparar varios servidores web que contengan el código necesario para explotarla, así como forzaría a usuarios a visitar dichos servidores (e.g., mediante el uso de ingeniería social [6]). El acceso de las víctimas a estos servidores conllevaría la instalación del Spyware mediante la explotación de la vulnerabilidad. El objetivo del código malicioso instalado sería capturar y enviar hacia un sistema remoto los números de tarjeta de crédito introducidos por los usuarios durante sus transacciones electrónicas. En este contexto, el Spyware podría emplear los mecanismos de extensión del navegador con el propósito de conseguir su objetivo.

Para garantizar el éxito del escenario anterior, el atacante debe persuadir a un número elevado de potenciales víctimas (i.e., usuarios con un navegador vulnerable) a visitar los servidores preparados. A mayor número de víctimas persuadidas, mayor será el éxito del atacante. El factor tiempo también es un elemento decisivo en este proceso, dado que tan pronto como la vulnerabilidad sea reportada y corregida por los usuarios, menor será la probabilidad de éxito. Con la finalidad de incrementar la velocidad del proceso de infección, otras técnicas pueden ser utilizadas, tales como en envenenamiento de DNS, el envío de correo no solicitado (*spam*), o el cross-site scripting, redirigiendo a los usuarios a los servidores preparados.

Después de la infección, el Spyware — visto ahora como una extensión del navegador — permanece activo mientras analiza los datos que el navegador recibe o envía, a la espera de información de transacciones electrónicas (i.e., números de tarjetas de crédito, fechas de expiración, contraseñas, etc.). Toda la información capturada por el Spyware será enviada una vez post-procesada y protegida hacia un sistema remoto controlado por el atacante. Debemos notar que, a pesar de que el navegador puede usar criptografía para garantizar la autenticidad, integridad y confidencialidad de la información intercambiada con los servidores de comercio electrónico (e.g., usando el protocolo SSL/TLS), esto no protege la información robada por el Spyware.

Si analizamos el uso de SSL/TLS en el modelo de capas de TCP/IP, podemos observar que la información permanece protegida entre la capa de aplicación (HTTPS) y la capa de transporte (TCP). Sin embargo, dado que el Spyware que infectó el navegador es ejecutado en la capa de aplicación (i.e., HTTPS), éste intercepta la información antes de ser protegida por SSL/TLS. Asimismo, cabe destacar que el Spyware no provocará ningún mensaje sospechoso asociado a los certificados digitales de SSL/TLS, ya que el proceso de verificación no se ve afectado por la infección. De forma similar, el uso de códigos de verificación, tales como CVC2, CVV o CID, no ayudan tampoco a detectar o prevenir actividades maliciosas asociadas a un posible Spyware.

III. Técnicas de prevención

III-A. Mecanismos genéricos

Podemos considerar tres métodos principales para prevenir nuestros sistemas de las infecciones de Spyware. En primer lugar, fomentar las buenas prácticas por parte de los usuarios, manteniendo actualizado tanto el sistema operativo como las aplicaciones, impedir las descargas de software de fuentes no fiables o ignorar los correos y contenidos adjuntos de remitentes desconocidos. Desafortunadamente, estas prácticas no son cumplidas muchas veces por los usuarios, ni tampoco son completamente efectivas.

Otras formas de prevención, más técnicas, se basan en el diseño de patrones de protección. El objetivo de estos patrones es impedir la infección de un sistema o bien reducir el daño de la infección. Podemos agrupar dentro de esta categoría la utilización de anillos de protección. Dichos anillos establecen una estructura de confianza por capas en el sistema operativo y se complementan con hardware específico para poder realizar una separación efectiva entre procesos de confianza y procesos sospechosos. A través de esta solución, se pueden ofrecer distintos niveles de acceso a los recursos del sistema. De hecho, los anillos se organizan de forma jerárquica, estructurando aquellos dominios más privilegiados y confiables hasta los de menores privilegios y nivel de confiabilidad. De este modo, se reduce el riesgo de que procesos de tipo Spyware ataquen al núcleo del sistema operativo (lo que les permitiría obtener el control del sistema al completo [4]). Estos métodos han demostrado que, aunque reducen las consecuencias de una infección, no son totalmente efectivos.

El tercer mecanismo genérico, pensado especialmente para complementos o extensiones de software, consiste en verificar la autenticidad del código que se está ejecutando. Una primera solución consiste en la utilización de firmas digitales. Estas firmas se asociarán al código que ha de ser ejecutado y permiten verificar su autenticidad [17]. Para aquel código que no haya sido firmado, los usuarios deberán decidir entre: (1) no usar sus funcionalidades, o (2) exponerse a ciertos niveles de riesgo si dicho código es ejecutado. Con esta finalidad, se han dedicado esfuerzos a la investigación de mecanismos de tipo PCC (*Proof-Carrying Code*) y MCC (*Model-Carrying Code*). Ambas soluciones ofrecen una infraestructura para probar que el código se comportará de manera segura antes de su ejecución. Estas técnicas resultan eficientes en general a su aplicación al código móvil, pero ineficientes en el caso navegadores web [13].

III-B. Mecanismos específicos

Ante la ineficiencia de los métodos genéricos, las tendencias actuales se centran en la investigación de aplicaciones automáticas capaces de detectar y aislar código de tipo Spyware. Según la estrategia de análisis que utilicemos para la detección, podemos clasificar estos métodos en dos categorías principales: detección basada en análisis sintáctico (también conocida como detección de patrones o firmas) y detección basada en análisis semántico (detección por comportamiento).

Análisis Sintáctico — Consiste en la comparación del código a ejecutar contra una base de datos de firmas asociadas a código malicioso. El mayor inconveniente asociado a esta primera solución, al igual que ocurre con la totalidad de sistemas de detección basados en firmas, es la dependencia de una actualización periódica de la base de firmas. Por otro lado, la detección efectuada por este tipo de propuestas es fácil de evadir mediante técnicas de ofuscación de código [10], tales como el polimorfismo y el metamorfismo.

Análisis Semántico — Consiste en modelar la interacción de código binario con el sistema — a partir del cual se determina si se trata de software malicioso. Esta estrategia supera las deficiencias del análisis sintáctico, dado que las mutaciones no afectan al comportamiento final del Spyware. Por esto, esta estrategia es conocida también como detección basada en comportamiento. En función de cómo la información para el modelado es obtenida podemos encontrar dos categorías [5]: (1) análisis dinámico y (2) análisis estático.

Dentro de la primera categoría, encontramos, por ejemplo, el trabajo presentado por Vogt *et al.* en [12]. La propuesta consiste en un mecanismo de prevención de ataques XSS orientados al robo de información de usuario. Más concretamente, el mecanismo consiste en la utilización de un marcado de datos para la construcción y análisis de modelos dinámicos. Este mecanismo sufre, sin embargo, grandes deficiencias de rendimiento que lo hacen impracticable para la protección contra el Spyware en forma de extensiones de navegador. De hecho, el gran número de extensiones utilizadas actualmente en navegadores tales como Mozilla Firefox [13], hace que la solución propuesta por Vogt *et al.*, basada en la pre-evaluación e interpretación de cualquier código ejecutable que trate de interactuar con los recursos del navegador, resulte a la práctica muy ineficiente. Una mejora propuesta por Russo *et al.* en [14] trata de mejorar dicho rendimiento a partir de una reducción de las operaciones a monitorizar. En este caso, se propone la monitorización de tan sólo aquellas operaciones que interactúan con el DOM (*Document Object Model*) asociado a cada ejecución, redefiniendo su semántica. La propuesta no se centra únicamente en ataques de tipo Spyware mediante inyección de código malicioso, sino que más bien ofrece un mecanismo general para prevenir cualquier flujo considerado como de robo de información.

En [16], Moshchuk *et al.* proponen una estrategia de análisis basada en la utilización de dispositivos de tipo *proxy*. Estos dispositivos interceptan y analizan el contenido dirigido desde las aplicaciones web hacia los recursos del navegador. En el supuesto que dicho contenido no pueda ser analizado de manera estática, por motivos de rendimiento, por ejemplo, se propone realizar un renderizado a través de máquinas virtuales. Éstas se encargarán de supervisar los efectos de la ejecución del contenido interceptado, y determinarán si debe ser aceptado o rechazado. A diferencia de otros métodos que tratan de prevenir la intercepción de información confidencial, la propuesta se centra en evitar la instalación dentro del navegador de Malware procedente de la web analizada. Ciertas limitaciones relacionadas con el indeterminismo del paradigma web actual, ya que el flujo de ejecución puede variar en función del contexto y la interacción con los usuarios, podrían hacer impracticable este tipo de propuestas, debido al alto número de falsos positivos y negativos que se pueden llegar a generar.

Otras aproximaciones para la supervisión en tiempo de ejecución presentadas por Kirda *et al.* en [8], plantean un mecanismo de detección de Spyware aplicado específicamente a los componentes internos del navegador Internet Explorer (concretamente, las bibliotecas BHO — *Browser Helper Object* — asociadas al navegador). Esta solución propone una supervisión híbrida a partir de un análisis tanto dinámico como estático. Mediante el análisis dinámico, se supervisa la interacción de los componentes con el navegador, registrando las funciones COM (*Component Object Model*) que son invocadas como respuesta a los eventos de un usuario. Este registro permite identificar eventos específicos, que son posteriormente analizados más en detalle, a partir de un modelo estático. Este segundo análisis concluye con la creación de un grafo de flujos que caracteriza las reacciones asociadas a cada evento.

Nos gustaría destacar, por último, la propuesta presentada en [13] por Ter *et al.*. Esta propuesta ofrece protección de confidencialidad e integridad de los datos del usuario mediante la supervisión del acceso a datos y servicios realizados por extensiones de navegadores Mozilla Firefox a través de componentes XPCOM (*Cross Platform Component Object Model*) de Mozilla. La novedad de esta propuesta reside en la utilización de firmas del usuario asociadas a las extensiones del navegador. Estas firmas serán supervisadas tanto en el proceso de instalación de las extensiones, como durante la carga y ejecución de las extensiones. El mayor inconveniente relacionado con la propuesta, al igual que sucede con los mecanismo de prevención genéricos tratados en la sección III-A, es la dependencia del buen comportamiento por parte de los usuarios, lo que en general comporta el fracaso de la medida de prevención.

IV. Conclusiones

En este artículo se ha presentado un análisis de soluciones existentes contra la infección de sistemas mediante código malicioso (Malware) de tipo Spyware. El concepto de Spyware (o software espía) ha sido presentado como una evolución de Malware tradicional, cuyo objetivo es la interceptación no consentida de información perteneciente a los usuarios de los recursos de un navegador web. Dicha interceptación pretende violar, por lo tanto, la privacidad de usuarios de aplicaciones web relacionadas con educación, salud, banca, o redes sociales. Se han analizado a continuación algunas de las técnicas que pueden ser utilizadas por código Spyware para infectar los recursos de un sistema. Finalmente, se ha concluido el análisis con un resumen de soluciones generales para la prevención de este tipo de infecciones. Una presentación más elaborada de nuestro estudio así como los resultados iniciales de una propuesta propia serán tratados en un informe futuro.

Agradecimientos — Este trabajo ha sido financiado por el Ministerio de Ciencia y Educación, a través de los proyectos TSI2007-65406-C03-03 E-AEGIS y CONSOLIDER-INGENIO CSD2007-00004 ARES.

REFERENCIAS

[1] Cary, C., Wen, H. J., Mahatanankoon, P. (2004). A viable solution to enterprise development and systems integration: a case study of web services implementation. International Journal of Management and Enterprise Development, 1(2):164–175, Inderscience.

[2] Christodorescu, M., Jha, S., Seshia, S. A., Song, D., Bryant, R. E. (2005). Semantics-Aware Malware Detection. IEEE Symposium on Security and Privacy (S&P'05), pp.32-46.

[3] Egele M., Kruegel C., Kirda, E., Yin, H., Song, H. (2007). Dynamic Spyware Analysis. *USENIX Annual Technical Conference*.

[4] Embleton, S., Sparks, S., Zou, C. (2008). SMM Rootkits: a New Breed of OS Independent Malware. In *SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication netowrks*, pp. 1–12, ACM.

[5] Jacob, G., Debar, H., Filiol, E. (2008). Behavioral Detection of Malware: From a Survey Towards an Established Taxonomy. *Journal in Computer Virology*, 4(3):251–266.

[6] Garcia-Alfaro, J., Cuppens, F., Autrel, F., Castella-Roca, J., Borrell, J., Navarro, G., Ortega-Ruiz, J. (2005). Protecting On-line Casinos against Fraudulent Player Drop-out. IEEE International Conference on Information Technology. IEEE Computer Society.

[7] Hu, Q., Dinev, T. (2005). Is Spyware an Internet Nuisance or Public Menace?. *Communications of the ACM*, SPECIAL ISSUE: Spyware, 48(8):61-66.

[8] Kirda, E., Kruegel, C., Banks, G., Vigna, G., Kemmerer, R. A. (2006). Behavior-Based Spyware Detection. *USENIX Security '06*, Vancouver, Canada.

[9] Lee, Y., Kozar, K.A. (2005). Investigating Factors Affecting the Adoption of Anti-Spyware Systems.. *Communications of the ACM*, SPECIAL ISSUE: Spyware, 48(8):72-77.

[10] Moser, A., Kruegel C., Kirda E. (2007). Limits of Static Analysis for Malware Detection. *Computer Security Applications Conference*, ACSAC 2007, pp. 421-430.

[11] Skoudis, E., Zeltser, L. (2004). Malware: Fighting Malicious Code. *Prentice Hall PTR*.

[12] Vogt, P., Nentwich, F., Jovanovic, N., Kirda, E., Kruegel, C., Vigna, G. (2007). Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis.. *Proceeding of the Network and Distributed System Security Symposium (NDSS?07)*.

[13] Ter Louw M., Soon Lim, J., Venkatakrishnan, V.N. (2007). Extebsible Web Browser Security. *Lecture Notes in Computer Science*. Springer.

[14] Russo, A., Sabelfeld, A., Chudnov, A. (2009). Tracking Information Flow in Dynamic Tree Structures. *M. Backes an P. Ning (Eds.): ESORICS 2009*, LNCS pp. 86-103, Springer.

[15] Hallaraker, O., Vigna, G. (2005). Detecting Malicious JavaScript Code in Mozilla. *10th IEEE International Conference on Engineering of Complex Computer Systems*, pp 85-94.

[16] Moshchuk, A., Bragin, T., Deville, D., Gribble, S.D., Levy H.M. (2007). SpyProxy: Execution-based Detection of Malicious Web Content. *16h USENIX Security Symposium*.

[17] R. Sekar, V.N. Venkatakrishnan, Samik Basu, Sandeep Bhatkar, Daniel C. Du Varney. (2003). Model-Carrying Code: A Practical Approach for Safe Execution of Untrusted Applications.. *19th ACM symposium on Operating systems principles*, ACM.