Gestión de recursos de un navegador Web para prevenir ataques contra la privacidad en Tor

G. Navarro-Arribas[†], J. Garcia-Alfaro[†,‡], O. Mula-Valls[†] y J. Herrera-Joancomartí[†]

***Resumen*—En este trabajo presentamos una propuesta para el despliegue de políticas de autorización contextuales sobre navegadores Web. El objetivo de nuestra propuesta es la automatización del proceso de control de recursos de un navegador Web cuando el contexto de navegación varía. Cuando esto sucede, los requisitos de seguridad del navegador cambian igualmente. La utilización de nuestra propuesta está orientada a facilitar esta adaptación en los requerimientos de seguridad del nuevo contexto y desplegarlos en el navegador sin necesidad de interacción por parte del usuario. Presentamos una aplicación concreta de nuestro trabajo en forma de *plug-in* para la adaptación de los requerimientos de seguridad de navegadores Mozilla/Firefox cuando un contexto de navegación anónima es activado.**

***Palabras Clave*—Seguridad en redes informáticas (*Computer network security*); políticas de seguridad (*security policies*; protección de aplicaciones (*sofware protection*).**

I. Introducción

La Web se está convirtiendo en una interfaz universal para el desarrollo de aplicaciones de todo tipo: desde la tradicional banca electrónica, pasando por el correo electrónico, los procesadores de texto e incluso completas redes sociales. A medida que la Web se transforma, las tecnologías que le rodean se hacen más complejas. En especial, esta transformación afecta de manera decisiva a la aplicación que hace posible la interacción con la Web desde el lado del cliente: el navegador Web. La complejidad actual de la Web repercute directamente en los aspectos de seguridad de estas aplicaciones y, en especial, al tratamiento de sus recursos.

Actualmente estamos trabajando en la implementación de un gestor de políticas contextuales en XACML [1] para navegadores Web. El objetivo de nuestro trabajo es poder automatizar la gestión de recursos asociados al navegador de manera flexible y dinámica. La utilización de distintos contextos de seguridad ayudará además a que el navegador pueda adaptarse a las necesidades de seguridad requeridas por el entorno de trabajo definido por el usuario. En este aspecto, presentamos en este artículo una aplicación concreta de nuestra propuesta para adaptar los requerimientos de seguridad del navegador cuando un contexto de navegación anónima es activado. Discutimos también en este trabajo el desarrollo actual de nuestra propuesta en forma de *plug-in* para la familia de navegadores Web Mozilla/Firefox.

La organización del artículo es la siguiente. En la sección II presentamos el uso de la nomenclatura AAA (*Authentication, Authorization, and Accounting*), discutimos sobre el uso del lenguaje XACML, e introducimos el desarrollo de nuestra propuesta en forma de *plug-in* para navegadores Web Mozilla/Firefox. En la sección III mostramos una aplicación concreta de nuestra propuesta para adaptar los requerimientos de seguridad de Mozilla/Firefox para su navegación anónima a través de la infraestructura de anonimato del proyecto Tor. Cerramos por último el artículo en la sección IV con una serie de conclusiones.

II. Antecedentes

A. La nomenclatura AAA

El marco de trabajo AAA (*Authentication, Authorization, and Accounting*) [2], [3] presenta un esquema común orientado a los servicios de Internet que ofrecen autenticación, autorización y contabilidad para asegurar la compatibilidad e interoperabilidad de diferentes dominios. Dicho esquema proporciona una base común que se ha utilizado mucho en productos y sistemas distribuidos.

Un escenario típico de AAA presenta una arquitectura capaz de autenticar usuarios, gestionar peticiones de autorización (o control de acceso) y recoger datos de contabilidad (*accounting*). Respecto a la parte de autorización [4], AAA extiende los elementos de gestión de políticas introducidos en [5]. Estos elementos son el punto de decisión de políticas PDP (*Policy Decision Point*), y el punto de aplicación de política PEP (*Policy Enforcement Point*).

En general, el esquema de autorización de un sistema estará baso en una política de seguridad que rige lo que se puede o no hacer en el sistema. Dicha política puede estar distribuida entre diversos componentes o localizaciones, o incluso distribuida en forma de certificados de atributo [4]. Las tareas relacionadas con autorización las lleva a cabo el servidor AAA, que puede estar compuesto de diversos módulos, algunos de los cuales introducimos a continuación.

De manera genérica definimos el PDP como el módulo del sistema encargado de tomar la decisión de autorización. Es decir, dada una petición en la que un usuario quiere realizar una determinada acción, el PDP será el encargado de decidir

[†]Universitat Autònoma de Barcelona.
[‡]Universitat Oberta de Catalunya.
Este trabajo está financiado por el Ministerio de Ciencia y Educación, a través de los proyectos *CONSOLIDER CSD2007-00004* y *TSI2006-03481*, y el programa de becas de la Fundación “la Caixa”.

si dicha acción puede o no ser permitida. Por otra parte, el PEP será el modulo encargado de hacer efectiva la aplicación de la política. El PEP es quien controla el acceso al recurso, trata directamente con el usuario, y consulta al PDP para saber si debe permitir el acceso del usuario o no.

Generalmente se suelen definir otros módulos adicionales. Cabe destacar el punto de información de políticas PIP (*Policy Information Point*). El PIP es el módulo encargado de recoger información contextual sobre la política del sistema, para que el PDP pueda usarla en la toma de decisión. Otro módulo a destacar es el punto de recuperación de políticas PRP (*Policy Retrieval Point*). El PRP es el módulo encargado de obtener la política a aplicar de un repositorio de políticas.

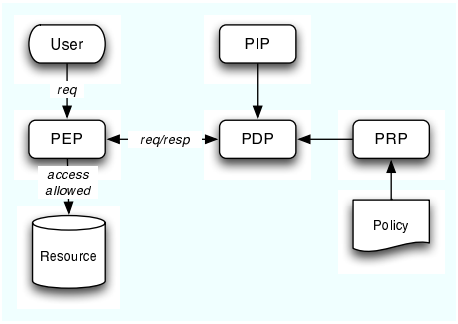


Fig. 1. Componentes principales en un esquema de autorización.

B. XACML

XACML (*eXtensible Access Control Markup Language*) es un lenguaje estándar basado en XML que permite la especificación tanto de políticas de control de acceso genéricas así como de los mensajes de petición/respuesta. La versión 2.0 de XACML [1] fue estandarizada por OASIS en 2005 y está siendo implementada y utilizada por numerosos proyectos y productos comerciales. XACML proporciona un lenguaje suficientemente rico y flexible como para adaptarse a la mayoría de situaciones y escenarios de control de acceso con un especial énfasis en sistemas distribuidos.

Una política XACML presenta un formato especifico, siendo su elemento básico la regla (*rule*). Cada regla tiene asociada un *target*, que indica a que (o quien) se aplica la regla, un *effect*, que suele ser *permit* o *deny*, y una condición. Si la evaluación de la condición es favorable, el resultado de evaluar la regla al completo será el indicado por el campo *effect*. Como se puede ver en la Fig. 2, una o más reglas están asociadas a una política que, a su vez, también puede especificar un *target* y unas obligaciones (*obligations*). Dichas obligaciones especifican acciones a realizar por parte del verificador de la política a la vez que se aplica dicha política [6]. Generalmente será el PEP correspondiente el encargado de realizar dichas acciones. El resultado de evaluar una política queda determinado por el resultado de evaluar todas las reglas que contiene. Finalmente, una o más políticas están incluidas en un *PolicySet* (conjunto de políticas) que también puede tener asociado un *target* y unas obligaciones. El resultado de evaluar un *PolicySet* queda determinado análogamente por el resultado de evaluar todas sus políticas.

La manera en que en XACML se combina el resultado de la evaluación de todas las reglas incluidas en una misma política y de todas las políticas en un mismo conjunto de políticas (*PolicySet*) viene determinada por los algoritmos de combinación. Dichos algoritmos, no sólo se utilizan para combinar políticas y reglas sino que además sirven en la resolución de conflictos, ya que se aplican en el caso en el que más de una regla o política haga referencia a un mismo *target*. Existen un conjunto de algoritmos estándar aplicables tanto a la combinación de reglas como a la de políticas:

- *deny-overrides*: una evaluación con efecto *deny* tiene preferencia sobre el resto.
- *ordered-deny-overrides*: igual pero la evaluación de reglas/políticas se tiene que hacer en el orden en el que aparecen en la política.
- *permit-overrides*: una evaluación con efecto *permit* tiene preferencia sobre el resto.
- *ordered-permit-overrides*: igual pero la evaluación de reglas/políticas se tiene que hacer en el orden en el que aparecen en la política.
- *first-applicable*: el resultado es el efecto de la primera regla/política que se puede aplicar (cuyo *target* corresponde al de la petición).
- *only-one-applicable*: (únicamente para la combinación de políticas) el resultado es el efecto de la evaluación de la única política aplicable. Si hay más de una, el resultado es indeterminado.

En nuestro caso, de manera muy resumida, mediante el uso de XACML podemos especificar la tripleta tradicional 'sujeto-recurso-acción' adaptada a nuestro problema y contexto descrito. Por ejemplo, en el caso de querer evaluar si un script debe o no ser ejecutado por un navegador Web, mediante XACML especificaremos si un *script* (sujeto) tiene permiso para acceder y/o modificar (acción) un recurso del navegador (objeto). En la sección III-C mostramos con más detalle como se definen las políticas de nuestra propuesta.

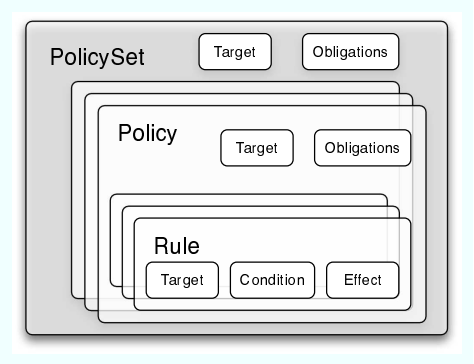


Fig. 2. Elementos principales de XACML.

C. Plug-in de nuestra propuesta para Firefox

La implementación específica de nuestra propuesta de control de acceso, en adelante, XAPO (*XAcml Policy Officer*), se basa en el marco de desarrollo de Mozilla para la implementación de extensiones en el navegador Web Firefox (i.e., plug-ins). El desarrollo de XAPO se basa en la utilización de los lenguajes Java, JavaScript y XUL (*XML User Interface Language*) [7]. El plug-in se ejecuta dentro del navegador mediante la interfaz *chrome* utilizada por las aplicaciones de Mozilla [8]. Desde esta interfaz, XAPO, al igual que cualquier otro código ejecutado en modo *chrome*, puede realizar las acciones de nuestra propuesta, tales como acceso a las opciones de configuración, almacenamiento y lectura de preferencias, activación y desactivación de componentes del navegador (por ejemplo, Java, JavaScript y Shockwave/Flash), etc. La activación/desactivación de componentes se realiza a través de la interfaz XPCOM del navegador Mozilla/Firefox. Por el momento, esta opción está únicamente disponible en la versión 3 del navegador, la cual está aún en fase de desarrollo.

La implementación de XAPO consiste actualmente de unas 400 líneas de código JavaScript y unas 100 líneas de código XUL para implementar la interfaz de usuario. Para la implementación del punto de decisión de políticas PDP (*Policy Decision Point*) y del punto de aplicación de política PEP (*Policy Enforcement Point*) hemos utilizado el marco de trabajo *SunXACML* [9]. *SunXACML* es una implementación *open source* del estándar XACML de OASIS. Dicha implementación, desarrollada mediante lenguaje Java, se ejecuta dentro de XAPO gracias a la interfaz *LiveConnect* proporcionada por Mozilla. La instalación del conjunto de componentes de XAPO se realiza de manera única a través de un paquete *xpi*. En el siguiente apartado presentamos la utilización de XAPO para la adaptación de los requerimientos de seguridad de navegadores Mozilla/Firefox cuando un contexto de navegación anónima es activado.

III. Prevención de ataques de privacidad en un contexto de navegación anónima

Presentamos en esta sección una aplicación concreta de nuestra propuesta para adaptar los requerimientos de seguridad de un navegador Web cuando un contexto de navegación anónima es activado. Nuestro ejemplo de navegación anónima se basa en la utilización de la infraestructura de anonimato del proyecto Tor [10]. Introducimos a continuación alguna de las características de Tor y el ataque concreto que pretendemos evitar a través del uso de nuestra propuesta.

A. La infraestructura de comunicaciones del proyecto Tor

Multitud de infraestructuras orientadas a reforzar el anonimato del tráfico dirigido hacia y por Internet han sido propuestas en la literatura. El principal objetivo de estas infraestructuras es la ocultación de la identidad de sus usuarios. Desde simples redes de *proxies* hasta complejos sistemas criptográficos, éstas infraestructuras ayudan a reforzar tanto el anonimato de servicios de alta latencia (por ejemplo, correo electrónico) como de baja latencia (por ejemplo, aplicaciones y servicios Web). Una de las infraestructuras más utilizadas en la actualidad para navegar de forma anónima a través de la Web es la infraestructura del proyecto Tor (*The second generation Onion Router*) [10]. Basada en la utilización de un esquema criptográfico conocido como *onion routing* [11], los diferentes componentes del proyecto Tor se distribuyen actualmente en modo de software libre y disponibles para gran multitud de plataformas y sistemas operativos.

El objetivo principal de Tor es proteger la privacidad de los usuarios que redirigen tráfico a través de sus componentes. Por ello, Tor construye circuitos protegidos criptográficamente a través de los cuales los mensajes son redirigidos. Por cada circuito, los componentes involucrados en la redirección de los mensajes tratan de realizar su reenvío de manera impredecible. El contenido de cada mensaje es protegido mediante un cifrado independiente para cada nodo de la red de Tor, de manera que la existencia de adversarios controlando de forma parcial componentes de la red no pueda comprometer el origen de los mensajes. Tan pronto como un componente de la red recibe un nuevo mensaje, éste descifrará la capa que le corresponda a través de las claves que se han establecido durante la construcción del circuito. Una vez descifrada, el componente comprobará si ha de entregar el mensaje al exterior, o si ha de redirigirlo a otro componente del circuito. Los diferentes caminos por los que los mensajes de un usuario serán redirigidos los establece su propio proceso local, en el lado del cliente. Ningún otro componente conoce la ruta completa. Tan sólo el siguiente componente al que ha de redirigir el mensaje, en el caso de un nodo intermedio; o el destinatario final, en el caso de un nodo de salida.

La madurez del proyecto y su bajo impacto en el rendimiento de servicios *on-line* posicionan a esta infraestructura como plataforma ideal para navegar por la Web de forma anónima. Aunque algunos ataques reportados en la literatura demuestran que el nivel de anonimato de Tor podría llegar a ser altamente degradado (por ejemplo, ataques reportados en [12], [13], [14]), multitud de personas lo utilizan a diario con el objetivo de mejorar en lo posible algunas de sus actividades de navegación en las que desean que su identidad o localización permanezcan ocultas. Para ello, algunas condiciones han de cumplirse. Aparte de la instalación de los componentes necesarios del proyecto Tor en la red del usuario final, ciertos recursos del navegador deberían ser adaptados al nuevo contexto. Desde el propio proyecto Tor se nos advierte que su utilización para navegación anónima requiere no tan sólo un cambio de hábitos, sino también la reconfiguración de ciertas características del navegador utilizado. Es necesario, por ejemplo, la desactivación de ciertos recursos del navegador (por ejemplo, plug-ins de Java, Flash, JavaScript, etc.) así como vaciar de información otros componentes (por ejemplo, cookies del navegador asociadas con sitios Web previamente visitados) ya que de lo contrario, es relativamente sencillo manipular el comportamiento de estos componentes para

obtener la identidad o localización del usuario a través de la obtención de su dirección IP. En el apartado siguiente mostramos con un ejemplo práctico la obtención de la dirección IP de un navegador (previamente configurado para navegar de forma anónima mediante Tor) a través de la ejecución de código Java embebido en una Web de ejemplo.

B. Traspasando el anonimato de Tor a través de ataques Web

Para navegar de forma anónima a través de la red de Tor, un usuario deberá primero configurar su navegador Web para salir al exterior a través de un proxy HTTP como, por ejemplo, Privoxy [15]. De esta manera, cualquier petición HTTP realizada por el navegador es desviada hacia la red de Tor, en lugar de conectarse directamente a la Web a través de Internet. En realidad, no sólo las peticiones HTTP son desviadas hacia la red de Tor. También las peticiones de DNS realizadas por el navegador son dirigidas por Privoxy a través de la utilización del protocolo SOCKS [16]. Sin embargo, la riqueza en opciones adicionales de navegadores actuales complica la tarea de ofrecer anonimato a los usuarios de Tor. La ejecución de aplicaciones Flash, Java, ActiveX, etc., ofrece un gran dinamismo e interactividad entre usuarios y servicios Web, pero también abre un gran número de posibles agujeros de seguridad. De hecho, dichas aplicaciones pueden perfectamente dirigir su tráfico al exterior sin pasar a través del proxy configurado en el navegador.

En [17], los autores describen la utilización de este tipo de ataques Web ejecutados en el navegador para vulnerar el anonimato de los usuarios de Tor. A través de un engaño recibido, por ejemplo, mediante ingeniería social, *spam*, *phising*, etc., el navegador de un usuario es forzado a visitar el servicio Web de un posible atacante. Al hacerlo, un código malicioso contenido en el servidor Web del atacante es ejecutado en el navegador del usuario y abre un canal oculto entre usuario y dominio del atacante. Posteriormente, a través de un análisis del tráfico intercambiado, el atacante recibirá la información recogida por el código ejecutado en el entorno del navegador — por ejemplo, la dirección IP asociada con la identidad del usuario, así como el sistema operativo, características internas del navegador, etc. Es importante hacer notar que la obtención de la identidad del usuario no es en sí mismo un ataque contra la red de Tor. Lo que el ataque explota en realidad son las herramientas y el entorno exterior de la red de Tor, en particular, el navegador Web y su configuración de recursos para redirigir tráfico a través de Privoxy.

El ataque es simplemente una demostración sobre como comprometer el anonimato de ciertos usuarios de la red de Tor debido a un error en la configuración de su navegador Web. En [18], los autores describen como extender este sencillo ataque para identificar al usuario de Tor sin necesidad de tener el control sobre el servicio Web visitado. El atacante tiene bajo su control simplemente nodos de salida de la red de Tor. Desde estos nodos, se encarga de modificar tráfico HTTP y realizar de este modo un ataque de tipo *man-in-the-middle*. Más concretamente, el atacante modifica el tráfico HTTP para incluir elementos invisibles de tipo *iframe* con una referencia a otros servicios Web maliciosos, así como un identificador único para cada petición contenido en una *cookie* asociada al ataque. Al obtener la página, el navegador del usuario acabará descargando un código malicioso asociado al atacante — por ejemplo, una animación Flash. Si el plug-in adecuado se encuentra activado en el navegador del atacante, la animación Flash empezará a ejecutarse de manera invisible en el navegador. Dicha animación enviará la cookie del navegador directamente al servidor Web del atacante, vulnerando nuevamente el anonimato ofrecido por la red de Tor. A través de un análisis del tráfico recibido, el atacante acabará por descubrir la identidad y actividades asociadas a los usuarios que pasan a través de los nodos de salida de Tor que están bajo su control y tienen el plug-in Flash de su navegador activado. Los autores en [17] muestran como extender este y otros ataques similares con el objetivo de simplificar el análisis necesario y aumentar las probabilidades de comprometer usuarios y servicios escondidos tras la red del proyecto Tor.

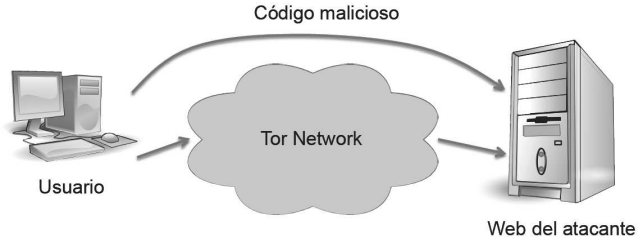


Fig. 3. Ejemplo de ataque Web para traspasar el anonimato de Tor. A través de ingeniería social, *phising*, o un ataque *man-in-the-middle*, un adversario consigue que el usuario descargue y ejecute un código malicioso. Al hacerlo, se abre un canal secreto entre el navegador y el servidor del atacante. El intercambio de mensajes no pasará por los nodos de la red de Tor. Al contrario, cualquier comunicación TCP entre ambos será dirigida directamente hacia el dominio del atacante, vulnerando así pues el anonimato ofrecido por Tor.

C. Utilización de XAPO y políticas XACML para evitar ataques contra la privacidad de Tor

Para evitar posibles ataques que traspasen el anonimato de Tor como los descritos en la sección III-B, nuestra propuesta utiliza un tipo de política concreta que permite no sólo evitar dichos ataques sino que además introduce suficiente flexibilidad para poder adaptarse a diferentes entornos y grados de privacidad.

La política XACML que proponemos se divide a su vez en dos políticas concretas. Por una parte, hay una política general que de forma explícita determina los recursos del navegador que tienen que ser protegidos: plugin Java, JavaScript, Flash, etc. y por otra se incluye una política tipo *whitelist* (lista blanca) que permite hacer un control detallado de aquellos dominios de confianza en los que se permite la activación y/o acceso a recursos concretos.

La primera política es la *generic-tor-policy*. Se compone de un elemento *Policy* que contiene una regla para cada recurso a

proteger. El efecto de cada regla es siempre *deny*, lo que indica que dicho recurso no puede ser accedido cuando la política es aplicada (ver Fig. 4).

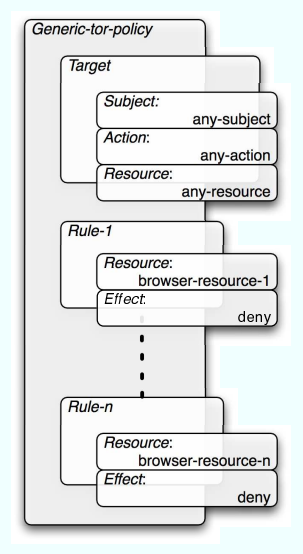


Fig. 4. Política genérica Tor.

La principal función de la *generic-tor-policy* es evitar de manera global problemas como el descrito en la sección III-B. Para ello se desactiva o se deniega explícitamente el acceso a todos los recursos sensibles cuando se esta utilizando Tor. Entre estos los más importantes son:

- Plugins del navegador como: Java, Flash, ActiveX, RealPlayer, Quicktime, Adobe PDF, etc. En la política se pueden especificar uno por uno o utilizar el recurso *all-plugins*. Con esta última referencia, XAPO mira todos los plugins presentes en el navegador y los desactiva.
- Cookies: es necesario proteger el acceso a cookies que puedan haber sido creadas con anterioridad a la activación de navegación anónima.

Aparte de plugins y cookies, la política puede incluir otros recursos que se consideren necesarios.

Como se puede ver, esta política es muy restrictiva y puede limitar la funcionalidad de aplicaciones a las que accede el usuario. Para mejorar la experiencia del usuario, consideramos importante la inclusión de una política de tipo *whitelist* que permita definir dominios de confianza a los que se permita acceder a recursos concretos del navegador. Esto evita la situación habitual en la que un usuario utiliza dos navegadores, uno para navegación anónima y con funcionalidad mínima, y otro para navegación normal y con una funcionalidad más completa o extendida. Es decir, el usuario puede determinar algunas aplicaciones en las que confía y a las que permite acceder a ciertos recursos sin tener que sacrificar las medidas de anonimato aportadas por Tor y XAPO en el resto de dominios por los que navega.

La *tor-whitelist-policy* define aquellos dominios a los que se les permite acceder a ciertos recursos concretos. Para cada dominio de confianza hay una política específica. Dicha política contiene reglas donde se detalla que acciones se permiten hacer sobre que recursos. El efecto de estas reglas será *Permit* y tendrá preferencia sobre la evaluación de la política genérica (ver Fig. 5). Mediante XAPO, el usuario puede escoger aquellos dominios de confianza y activar las opciones, recursos, etc. que prefiera. Estos cambios se guardan en la política *whitelist* correspondiente y tendrá efecto para sucesivas ejecuciones del navegador.

Tanto la política genérica como las políticas *whitelist* se combinan en un *PolicySet* mediante el algoritmo de combinación de políticas *permit-overrides* (ver sección II-B). De esta manera la política de *whitelist* toma precedencia sobre la genérica. Dicho de otra manera, las políticas *whitelist* expresan excepciones de la política genérica.

D. Ejemplo de políticas para Tor en XAPO

A continuación mostramos un ejemplo sencillo de políticas para Tor que se utiliza en XAPO para garantizar un nivel de anonimato adicional y poder prevenir ataques contra la privacidad y anonimato del usuario aun cuando este esté navegando utilizando Tor. El ejemplo, así como la nomenclatura de las políticas utilizadas, ha sido simplificado para mejorar su legibilidad. Aún así, el ejemplo muestra el funcionamiento de las políticas de forma concisa y clara.

En el siguiente listado (Listing 1) se muestra una posible política genérica de Tor. En ella podemos ver que se incluyen tres reglas: *java-plugin*, *javascript-plugin*, y *cookies*. La primera regla hace que XAPO desactive el plugin de Java, la segunda que desactive el interprete de JavaScript, y la tercera impide que la lectura de cookies a cualquier dominio.

```
<Policy PolicyId="tor−generic:default−tor−firefox"
        RuleCombiningAlgId="deny−overrides">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>

  <Rule RuleId="java−plugin" Effect="Deny">
    <Target>
      <Subjects><AnySubject/></Subjects>
      <Actions><AnyAction/></Actions>
      <Resources>
        <Resource>
          <ResourceMatch
              MatchId="function:anyURI−equal">
            <AttributeValue DataType="XMLSchema#anyURI">
              urn:browser:plugin:java
            </AttributeValue>
            <ResourceAttributeDesignator
                DataType="XMLSchema#anyURI"
                AttributeId ="resource:resource−id"/>
          </ResourceMatch>
        </Resource>
      </Resources>
    </Target>
  </Rule>

  <Rule RuleId="javascrip−plugin" Effect="Deny">
    <Target>
```

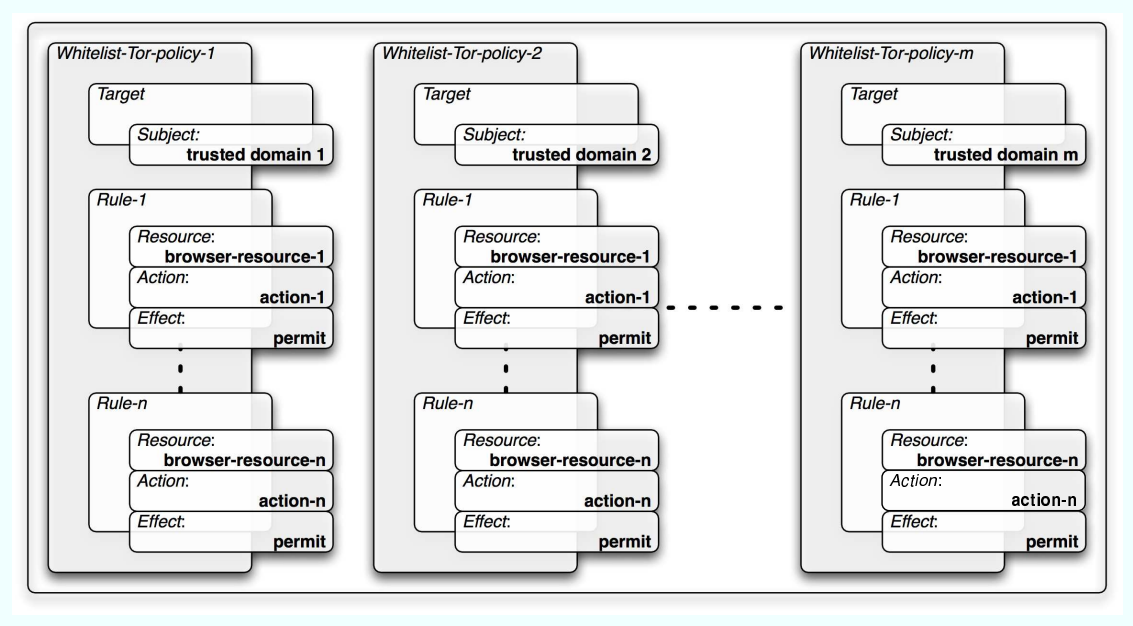


Fig. 5. Política whitelist de Tor.

```
    <Subjects><AnySubject/></Subjects>
    <Actions><AnyAction/></Actions>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="function:anyURI-equal">
          <AttributeValue DataType="XMLSchema#anyURI">
            urn:browser:plugin:javascript
          </AttributeValue>
          <ResourceAttributeDesignator
              DataType="XMLSchema#anyURI"
              AttributeId ="resource:resource-id"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
</Rule>

<Rule RuleId="cookies" Effect="Deny">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Actions>
      <Attribute AttributeId ="action:action-id"
                 DataType="XMLSchema#string">
        <AttributeValue>read</AttributeValue>
      </ Attribute >
    </Actions>
    <Resources>
      <Resource>
        <ResourceMatch
            MatchId="function:anyURI-equal">
          <AttributeValue
              DataType="XMLSchema#anyURI">
            urn:browser:document.cookie
          </AttributeValue>
          <ResourceAttributeDesignator
              DataType="XMLSchema#anyURI"
              AttributeId ="resource:resource-id"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
</Rule>
</Policy>
```

Listing 1. Ejemplo de política genérica de Tor.

Siguiendo con el ejemplo, el usuario puede decidir activar el interprete de JavaScript y el plugin de Java pero sólo para una aplicación Web de correo electrónico (`mail.trusted.domain.org`) a la que accede mediante HTTPS y en la que confía. En vez de tener que cambiar la política genérica de Tor, desactivarla, o iniciar una sesión sin Tor, el usuario puede incluir una política de tipo *whitelist* para hacer que la extensión XAPO permita ejecutar código JavaScript desde el dominio `https://mail.trusted.domain.org`. Como se muestra en el siguiente listado (Listing 2), hay una política aplicada al dominio correspondiente donde se incluyen dos reglas: una que activa el plugin de Java y otra que activa el plugin de JavaScript.

```
<Policy PolicyId="tor-whitelist:mail"
        RuleCombiningAlgId="permit-overrides">
  <Target>
    <Subjects>
      <Attribute AttributeId ="subject:subject-id"
                 DataType="XMLSchema#anyURI">
        <AttributeValue>
          https://mail.trusted.domain.org
        </AttributeValue>
      </ Attribute >
    </Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>

  <Rule RuleId="java-rule" Effect="Permit">
    <Target>
      <Subjects><AnySubject/></Subjects>
      <Actions><AnyAction/></Actions>
      <Resources>
        <Resource>
          <ResourceMatch
              MatchId="function:anyURI-equal">
            <AttributeValue DataType="XMLSchema#anyURI">
              urn:browser:plugin:java
            </AttributeValue>
```

```
            <ResourceAttributeDesignator
                DataType="XMLSchema#anyURI"
                 AttributeId ="resource:resource-id"/>
          </ResourceMatch>
        </Resource>
      </Resources>
    </Target>
  </Rule>

  <Rule RuleId="javascrip-rule" Effect="Permit">
    <Target>
      <Subjects><AnySubject/></Subjects>
      <Actions><AnyAction/></Actions>
      <Resources>
        <Resource>
          <ResourceMatch
              MatchId="function:anyURI-equal">
            <AttributeValue  DataType="XMLSchema#anyURI">
              urn:browser:plugin:javascript
            </AttributeValue>
            <ResourceAttributeDesignator
                DataType="XMLSchema#anyURI"
                 AttributeId ="resource:resource-id"/>
          </ResourceMatch>
        </Resource>
      </Resources>
    </Target>
  </Rule>
</Policy>
```

Listing 2. Ejemplo de política Tor-whitelist para el dominio trusted.domain.org.

En el siguiente listado (Listing 3) mostramos otra política *whitelist*, con la que el usuario activa JavaScript pero sólo para el dominio `trusted-bank.org`.

```
<Policy  PolicyId="tor-whitelist:bank"
        RuleCombiningAlgId="permit-overrides">
  <Target>
    <Subjects>
      <Attribute   AttributeId ="subject:subject-id"
                   DataType="XMLSchema#anyURI">
        <AttributeValue>trusted-bank.org</AttributeValue>
      </Attribute>
    </Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>

  <Rule RuleId="javascrip-rule" Effect="Permit">
    <Target>
      <Subjects><AnySubject/></Subjects>
      <Actions><AnyAction/></Actions>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="function:anyURI-equal">
            <AttributeValue  DataType="XMLSchema#anyURI">
              browser:plugin:javascript
            </AttributeValue>
            <ResourceAttributeDesignator
                DataType="XMLSchema#anyURI"
                 AttributeId ="resource:resource-id"/>
          </ResourceMatch>
        </Resource>
      </Resources>
    </Target>
  </Rule>
</Policy>
```

Listing 3. Ejemplo de política Tor-whitelist para el dominio trusted-bank.org.

Como se puede ver, la política permite la activación del intérprete de JavaScript para un dominio en concreto mediante la regla correspondiente.

Las tres políticas que se han visto en el ejemplo se combinan en un *PolicySet*. Un ejemplo simplificado de dicho *PolicySet* se puede ver en el siguiente listado (Listing 4).

```
<PolicySet PolicySetId="xapo:tor-policyset"
           PolicyCombiningAlgId="permit-overrides">
  <Description>
    Tor  policyset.
  </Description>
  <Target />

  <PolicyIdReference>
    tor-generic:default-tor-firefox
  </PolicyIdReference>
  <PolicyIdReference>tor-whitelist:mail</PolicyIdReference>
  <PolicyIdReference>tor-whitelist:bank</PolicyIdReference>

</PolicySet>
```

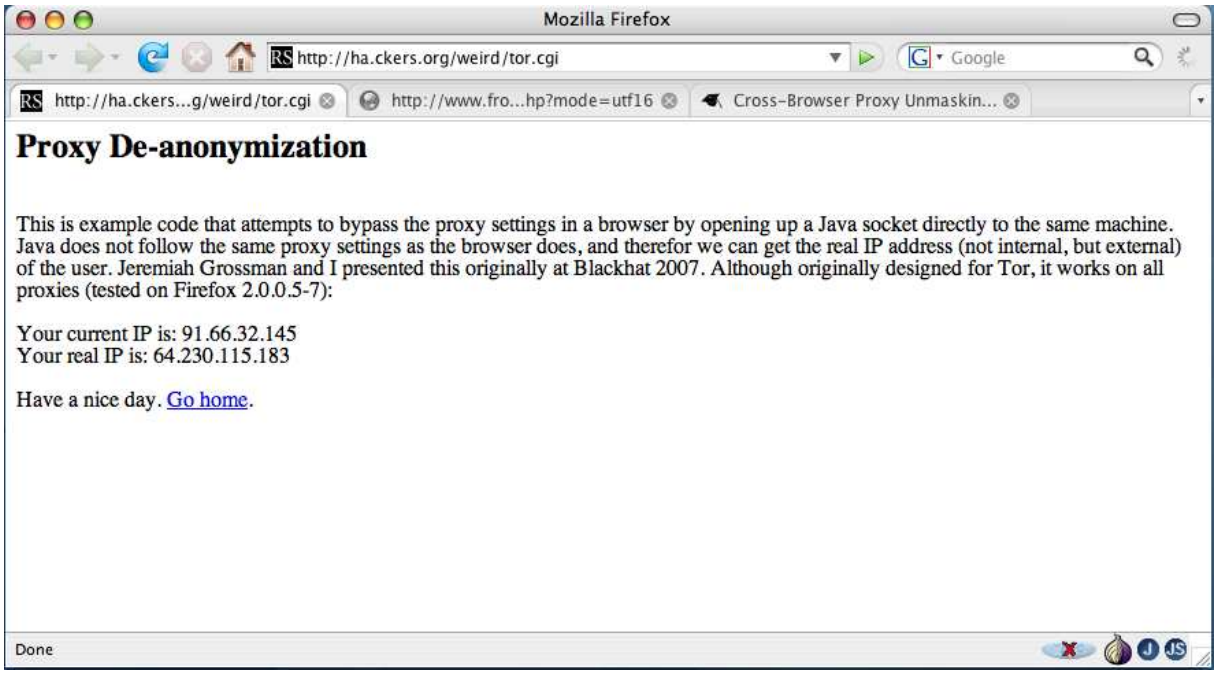
Listing 4. Ejemplo de policyset para Tor.

Para concluir esta sección, mostramos con un ejemplo práctico (ver Fig. 6(a) y Fig. 6(b)) la manera en la cual la activación de XAPO en un navegador Mozilla/Firefox previene ataques como los presentados en la sección III-B. El navegador utilizado es Mozilla/Firefox 3 Beta 2, configurado con XAPO y le extensión Torbutton [19] (para la configuración automática de Privoxy en las preferencias de navegación de Mozilla/Firefox). Como ya se ha discutido en la sección III-B, la gran cantidad de opciones disponibles en un navegador Mozilla/Firefox hacen que, sin las medidas apropiadas, una tercera parte pueda vulnerar la construcción de canales anónimos a través de Tor y resolver sin problemas la identidad del navegador. El ataque mostrado en la Fig. 6 explota la utilización de código Java ejecutado desde JavaScript para abrir un socket a través de *LiveConnect*. El código en cuestión realiza un petición HTTP al servidor que aloja la página `http://ha.ckers.org/weird/tor.cgi`. Puesto que la petición no pasa a través de la red de nodos de Tor, tras un simple análisis de las peticiones recibidas, y de manera automatizada, el atacante del sitio Web visitado averigua y muestra por pantalla información asociada al usuario, tal como la dirección IP. La Fig. 6(b) muestra como la activación de XAPO y, por consiguiente, el bloqueo de recursos asociado con la política gestionada a través de XAPO, previene la apertura del canal entre atacante y navegador.

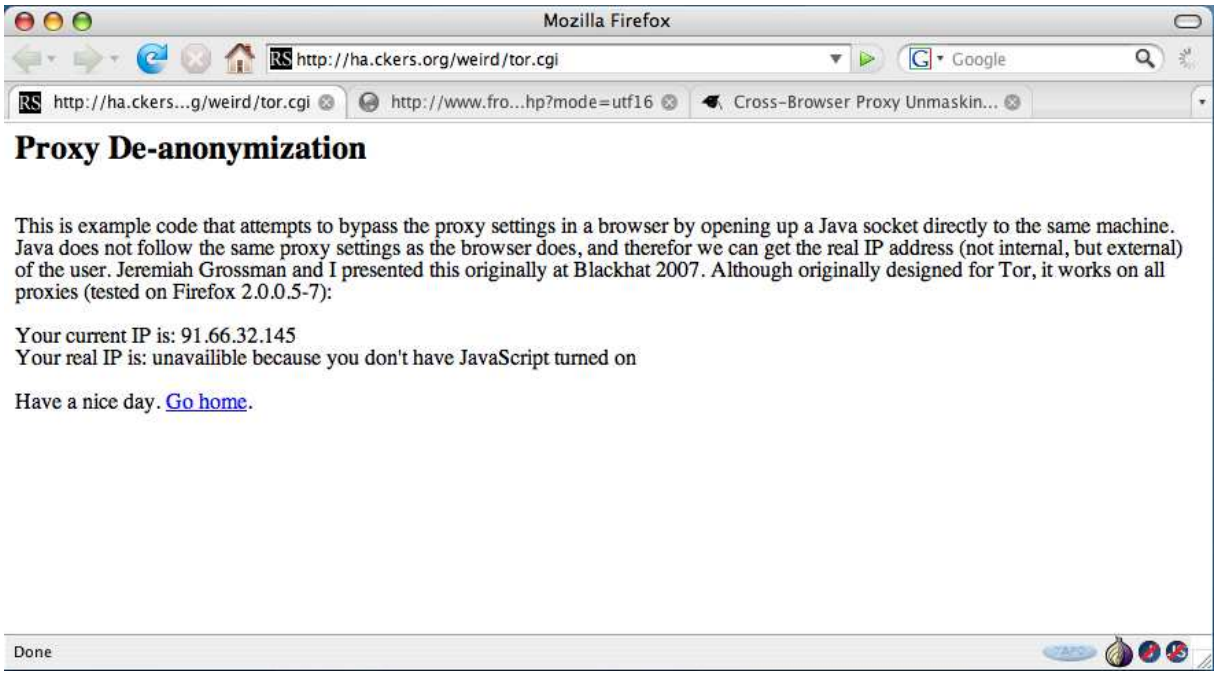
IV. Conclusiones

En este articulo hemos presentado una propuesta para la aplicación de políticas de seguridad en un navegador Web. Concretamente, hemos presentado una extensión del navegador Mozilla/Firefox que permite utilizar políticas expresadas en el lenguaje estándar XACML para proteger los recursos del navegador. Así mismo se ha visto como dicha extensión, que recibe el nombre de XAPO, se puede utilizar para garantizar el anonimato y la privacidad de los usuarios como complemento a la infraestructura de comunicaciones del proyecto Tor.

Tor proporciona una red de encaminadores de tráfico con el objetivo de garantizar la privacidad de sus usuarios.



(a) Torbutton activado y XAPO desactivado.



(b) Torbutton y XAPO activados.

Fig. 6. (a): Ejemplo de ataque para traspasar la configuración de Privoxy en un navegador Mozilla/Firefox con las extensión Torbutton activada y la extensión XAPO desactivada. El ataque consiste en abrir un canal adicional entre el entorno de ejecución del navegador y el atacante, y a través de este canal, la extracción de información de información asociada con el navegador; (b): Prevención del ataque al activar la extensión XAPO.

Para ello, la infraestructura redirige el tráfico de dichos usuarios a través de sus encaminadores mediante circuitos protegidos criptográficamente. Sin embargo, existen ataques conocidos que permiten vulnerar el anonimato y privacidad de usuarios redirigiendo tráfico a través de Tor. La inyección de código malicioso en un navegador Web, por ejemplo, permite a posibles atacantes obtener una conexión directa entre servicios Web bajo su control y el navegador. De esta manera, comprometer el anonimato e información asociada al navegador es relativamente sencillo.

Nuestra propuesta permite evitar dichos ataques definiendo una política de privacidad orientada a navegación anónima a través de Tor que garantice al usuario una mayor protección de su identidad e información sensible. Para ello, la política permite definir los recursos del navegador que se tienen que proteger como medida adicional a la protección proporcionada por Tor. Dicha política es suficientemente flexible como para adaptarse a los hábitos de navegación del usuario. Por ejemplo, a la vez que proporciona una protección completa, permite también definir con detalle *whitelists* de dominios de confianza. Mediante el uso de estas listas, el usuario puede indicar que dominios podrán disponer de recursos adicionales.

Actualmente, el prototipo de nuestra propuesta está desarrollado únicamente como extensión del navegador Mozilla/Firefox. Estamos trabajando en desarrollar extensiones equivalentes para otros navegadores como Safari, Konqueror o Internet Explorer. La utilización de un lenguaje estándar como XACML permite que las políticas se puedan intercambiar entre diferentes navegadores fácilmente. Así mismo, estamos estudiando la aplicación de XAPO en otros contextos no directamente relacionados con la infraestructura del proyecto Tor.

REFERENCIAS

[1] Godik, S., Moses, T., et al. eXtensible Access Control Markup Language (XACML) Version 2. Standard, OASIS. February 2005.

[2] Aboba, B., Mitton, D., Loughney, J., et al. Authentication, Authorization and Accounting IETF working group. [Online]. Available: http://tools.ietf.org/wg/aaa/

[3] Laat, C., Gross, G., Gommans, L., Vollbrecht, J. and Spence, D. Generic AAA Architecture RFC 2903, The Internet Society, August, 2000.

[4] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., Bruijn, B. de, Laat, C. de, Holdrege, M. and Spence, C. AAA Authorization Framework RFC 2904, The Internet Society, August, 2000.

[5] Yavatkar, R., Pendarakis, D. and Guerin, R. A Framework for Policy-based Admission Control RFC 2753. The Internet Society. January, 2000.

[6] Sloman, M. Policy Driven Management for Distributed Systems *Journal of Network and Systems Management*, vol. 2, part 4. Plenum Press. 1994.

[7] Ginda, R. Writing a Mozilla Application with XUL and Javascript. *O'Reilly Open Source Software Convention*, USA, 2000.

[8] Mcfarlane, N. *Rapid Application Development with Mozilla*. Prentice Hall PTR., 2004.

[9] Sun Microsystems SunXACML Implementation. [Online]. Available: http://sunxacml.sourceforge.net

[10] Dingledine, R., Mathewson, N., and Syverson, P. F. Tor: The second-generation Onion Router. *In: 13th conference on USENIX Security Symposium*, 2004.

[11] Reed, M. G., Syverson, P. F., and Goldschlag, D. M. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, 1998.

[12] Murdoch, S. J. and Danezis, G. Low-cost traffic analysis of Tor. *IEEE Symposium on Security and Privacy)*, pp. 183–195, 2005.

[13] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., and Sicker, D. Low-resource routing attacks against Tor. *2007 ACM workshop on Privacy in electronic society*, pp. 11–20, 2007.

[14] Wright, M. K., Adler, M., Levine, B. N., Shields, C. Passive-Logging Attacks Against Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC)*, Vol. 11, No. 2, Article 7, 1–33, Pub. date: May 2008.

[15] Privoxy - Home Page [Online]. Available: http://www.privoxy.org/

[16] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and Jones, L. SOCKS Protocol Version 5. *RFC1928*, March 1996.

[17] Abbott, T., Lai, K., Lieberman, M., and Price, E. Browser-Based Attacks on Tor. *7th Workshop on Privacy Enhancing Technologies (PET 2007)*, Lecture Notes in Computer Science, 4776(1):184–199, 2007.

[18] Christensen, A. et al. Practical Onion Hacking: Find the real address of Tor clients.. *FortConsult*, October 2006.

[19] Squires S. Torbutton. [Online]. Available: http://www.freehaven.net/ squires/torbutton/