A. Nieto, and G. Fernandez, "Sistema Colaborativo de Detección y Reacción ante Intrusiones basado en Intel vPro", *XII Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2012)*, pp. 45-50, 2012.
NICS Lab. Publications: https://www.nics.uma.es/publications

Sistema Colaborativo de Detección y Reacción ante Intrusiones basado en Intel vPro

Ana Nieto y Gerardo Fernandez
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga, España
Email: {nieto,gerardo}@lcc.uma.es

***Resumen*—En este trabajo proponemos una plataforma para el desarrollo de un sistema colaborativo para la detección y reacción ante intrusiones, empleando como base las tecnologías presentes en Intel vPro. La solución presentada está dirigida a solventar la necesidad de implantación de nuevas tecnologías que posibiliten la reacción ante ataques, independientemente del sistema operativo usado. Con este fin, en este trabajo abordamos tres puntos fundamentales: la detección de intrusiones colaborativa, la respuesta automática de los nodos ante la detección de una intrusión y el uso de herramientas que posibiliten asegurar la confianza en un nodo. En un sistema colaborativo como el que se propone aquí, un aspecto clave para la seguridad es la protección de las comunicaciones entre los mecanismos de detección y reacción frente a intrusiones. La modificación o el simple acceso a los datos intercambiados por tales sistemas supone un grave riesgo para la seguridad del entorno. Como resultado hemos desarrollado un prototipo preliminar para probar la solución propuesta en un escenario de ataque real.**

I. Introducción

Los entornos colaborativos están formados por diversas entidades que cooperan para lograr un objetivo común. Desde el punto de vista de los sistemas colaborativos para la detección de intrusiones, existen dos problemas fundamentales a la hora de garantizar su seguridad. Por una parte, el asegurar que un componente es confiable no es siempre factible, y se requieren complejos esquemas de confianza a tal fin. Por otra parte, el intercambio de información es un punto crítico en el que un atacante podría adquirir información sensible muy útil para sus propios fines, alterar dichos datos o incomunicar el canal de comunicación. En este trabajo presentamos una solución basada en la tecnología Intel vPro para el intercambio de información sensible entre varios sistemas de detección de intrusiones. Entre las características de Intel vPro a destacar encontramos el canal fuera de banda (OOB), el establecimiento de políticas de filtrado a nivel hardware y las mejoras de virtualización, que incluyen además la verificación de código ejecutable y el establecimiento de un camino de confianza desde el arranque de la plataforma. El modelo propuesto emplea estas características en un escenario de detección de intrusiones colaborativo, capaz además de reaccionar lo antes posible frente a las amenazas detectadas. De hecho, esta es una cuestión importante, no solo en entornos corporativos, donde un ataque efectivo puede acarrear consecuencias tan negativas como la fuga de información o la inhabilitación de la red. Este tipo de incidentes no sólo perjudica económicamente a la empresa, sino que, además, pueden dañar considerablemente su imagen ante futuros clientes. La solución propuesta no sólo pretende detener la propagación de los ataques lo antes posible, sino que, además es particularmente útil para mantener la confianza en el sistema de detección. Esto es fundamental para la seguridad, no sólo para evitar los falsos positivos/negativos intencionados, dirigidos a provocar el caos en el sistema, sino también para evitar aquellos equipos que no sean confiables. Mantener un esquema de confianza suele ser costoso, aquí empleamos elementos hardware incluidos en los equipos Intel vPro a tal fin, como describiremos a lo largo del artículo.

El artículo está estructurado como sigue. En la Sección II describimos el trabajo relacionado con sistemas de detección de intrusiones colaborativos. En la Seccion III describimos las características de Intel vPro empleadas en el desarrollo de la solución. La Sección IV presenta la solución propuesta y la Sección V describe el prototipo desarrollado para la demostración de la solución en un escenario de ataque real. Finalmente, la Sección VI muestra las conclusiones y el trabajo futuro.

II. Trabajo relacionado

Los Sistemas de Detección de Intrusiones (IDS) colaborativos se presentan hoy día como alternativa a los sistemas centralizados para la detección de intrusiones. La principal desventaja de estos últimos radica en que suponen un punto crítico de fallo y un cuello de botella, y estos dos factores hacen muy difícil su implantación en sistemas distribuidos. Si, además tenemos en cuenta que los nuevos paradigmas como el Internet del Futuro promueven insistentemente la futura interconexión de redes, no es de extrañar que los IDS colaborativos se alcen como una opción más adecuada que los sistemas centralizados para las redes del futuro. Otro factor a considerar es la heterogeneidad de redes, y, no solo en cuanto a dispositivos, sino también el software en ejecución en estos dispositivos. El considerable incremento de aplicaciones también aumenta el número de vulnerabilidades susceptibles de convertirse en una amenaza propagable por todo el entorno.

Diversos IDS distribuidos son analizados y comparados en [17], donde se concluye que la tecnología multi-agente incrementa el rendimiento y la precisión de tales sistemas. Este y otros estudios han motivado una parte importante de nuestra solución, ya que el rendimiento es un factor clave

en la respuesta temprana ante incidencias. Por otra parte, es fundamental para el desarrollo de la solución el disponer de una tecnología que permita la construcción del código a intercambiar por los agentes de forma ágil y simple, y que, además sea lo más dinámica posible. Algunos de ejemplos de soluciones que emplean código móvil autónomo son Micael[18], IDA[9], Sparta[16] y MAIDS[13]. Estos trabajos demuestran la adecuación del código móvil autónomo a la resolución de nuestro trabajo.

No obstante, un punto crítico en la detección de ataques, y por tanto foco de atención de numerosos trabajos de investigación, es el proceso de toma de decisiones, que puede ser local o colaborativo. La solución a adoptar depende de si un agente tiene suficientes pruebas para enviar una alerta (decisión local) o si necesita consultar a otros agentes en la red para contrastar la información, en cuyo caso el proceso de toma de decisiones debiera ser colaborativo. A modo de ejemplo, establecer un sistema de votaciones es una práctica común para decidir cuándo un incidente necesita ser tratado como alerta o no [12][20].

Finalmente, no todos los IDS responden a ataques. La respuesta a ataques en muchos casos consiste en enviar la evidencia del ataque a una base de datos que, posteriormente es analizada por un operario para la toma de decisiones final. En efecto, en varios sistemas, como los entornos críticos, esto debe ser así, ya que la respuesta automática puede causar más daño que beneficio (ej. aislar un servidor). Pero, en otros entornos este tipo de respuesta es insuficiente y permite la propagación de ataques que podrían ser evitados con el aislamiento de un equipo no crítico. Aquí nosotros entendemos que un mecanismo de respuesta no sólo es capaz de elevar alertas, sino que también proporciona respuesta automática ante situaciones de ataque. De forma resumida, un mecanismo de reacción colaborativo tradicional normalmente sigue uno de los siguientes enfoques: construcción de listas blancas/negras [22], seguimiento del ataque [21], obtención de evidencias [14] y aislamiento del tráfico de ataque [11] [10]. En concreto, el aislamiento es una medida comúnmente usada en escenarios de detección y control de ataques distribuidos [15][19][14].

El modelo de detección y respuesta ante incidencias colaborativo propuesto en este trabajo no se centra en un tipo particular de ataque. En lugar de ello la idea es que el código móvil intercambiado por nuestros agentes pueda ser modificado en tiempo real en base a las características del ataque y las evidencias recogidas. Más aún, la solución propuesta, al emplear componentes hardware, posibilita su adecuación a entornos independientes del sistema operativo. Una novedad bastante interesante de la solución propuesta es que involucra la utilización de mecanismos de confianza anti-modificación.

III. Intel vPro

Intel vPro agrupa un conjunto de tecnologías relacionadas con el mantenimiento remoto, la virtualización [1] y la ejecución de código seguro [2]. La diferencia respecto de otras soluciones similares es la presencia de componentes hardware que dotan a los equipos de seguridad adicional.

La gestión remota de equipos es llevada a cabo mediante la tecnología Intel AMT (*Intel Active Management Technology*), que cuenta con un dispositivo hardware a tal fin con servicios básicos incorporados [4]. Esta tecnología permite que un equipo en uso pueda ser gestionado remotamente sin que el usuario se percate de ello. Para ello emplea un canal fuera de banda (OOB), que usa una IP distinta a la IP usada por el sistema operativo. De hecho, el usuario no puede acceder a esta IP para el envío de tráfico, siendo un canal reservado accesible sólo por medio de servicios de gestión definidos por Intel AMT y que usan una clave de administrador. Como ejemplo de uso en nuestro caso, podríamos analizar el estado de un equipo para comprobar si está comprometido sin que el usuario se percate de que está siendo evaluado. Además de la característica de canal OOB, hemos empleado el espacio de almacenamiento destinado a los datos para aplicaciones de terceros (3PDS) para almacenar los comandos intercambiados por los sistemas de detección de intrusos. El espacio 3PDS es una memoria flash protegida que puede ser leída y escrita con independencia del sistema operativo.

Debemos destacar que contamos con dos tipos de interfaces que pueden proporcionar acceso al dispositivo Intel AMT: local y remota. La interfaz de acceso remoto está diseñada para proveer acceso remoto a los administradores, luego cuenta con más servicios disponibles que aquellos accesibles desde el interfaz local, desde el que no es posible por ejemplo visualizar el canal OOB. De hecho, el equipo emplea filtrado a nivel hardware [5] para evitar el acceso de usuarios locales a las características remotas definidas. También hay servicios que son accesibles desde ambas interfaces, por ejemplo el servicio de almacenamiento en el 3PDS (*Storage Service*). Otro servicio empleado en la solución propuesta es *System Defense*, que permite establecer las reglas de filtrado hardware por medio de la definición de políticas, de acuerdo a los parámetros de tráfico (ej. número de paquetes recibidos, dirección IP fuente/destino, etc.).

Por otra parte, la tecnología Intel VT (Intel Virtualization Technology) aporta ventajas para la virtualización. Al considerar que el sistema puede ser virtualizado esta tecnología incorpora características para incrementar el rendimiento, pero también la seguridad. En general, las principales características aportadas por Intel VT son: un mayor anillo de privilegios para el monitor de máquina virtual (VMM) o hipervisor, soporte hardware para el cambio entre VMM y los sistemas operativos virtualizados (o invitados) y que la información de estado del procesador es almacenada en espacios dedicados de memoria para el VMM y cada sistema operativo invitado [3].

Una tecnología de la que depende Intel VT es Intel TxT *Intel Trusted eXecution Technology*). La ejecución de código se valida usando esta tecnología, y permite además el arranque controlado del equipo dentro de un entorno de confianza. Intel TxT fue diseñada para proteger contra ataques basados en software, y es usada para validar tanto el entorno virtual como las aplicaciones. Para lograr tales objetivos, Intel TxT usa un dispositivo hardware conocido como TPM (*Trusted Platform Module*). Es a partir de la versión 1.2 de TPM cuando este

dispositivo incorpora características que lo hacen más útil para los propósitos de arranque seguro.

IV. Solución propuesta

La Figura 1 muestra un diagrama de la solución propuesta y los componentes harware y software empleados a tal fin. Como requerimos el uso de la tecnología Intel vPro, asumimos que todos los equipos de la red cuentan con esta tecnología. Esta es una restricción que puede relajarse si sólo nos interesa que los sistemas de detección y reacción ante intrusiones (IRS) permanezcan activos. En este caso, si lo que queremos proteger son sólo las comunicaciones entre los IRS sólo estos tienen que contar con la tecnología habilitada. Luego si, además de los IRS el resto de equipos también cuenta con esta tecnología o similar, entonces no sólo protegemos los IRS y su comunicación, sino también el resto de equipos de la red. Asumimos este último enfoque por ser el más general.

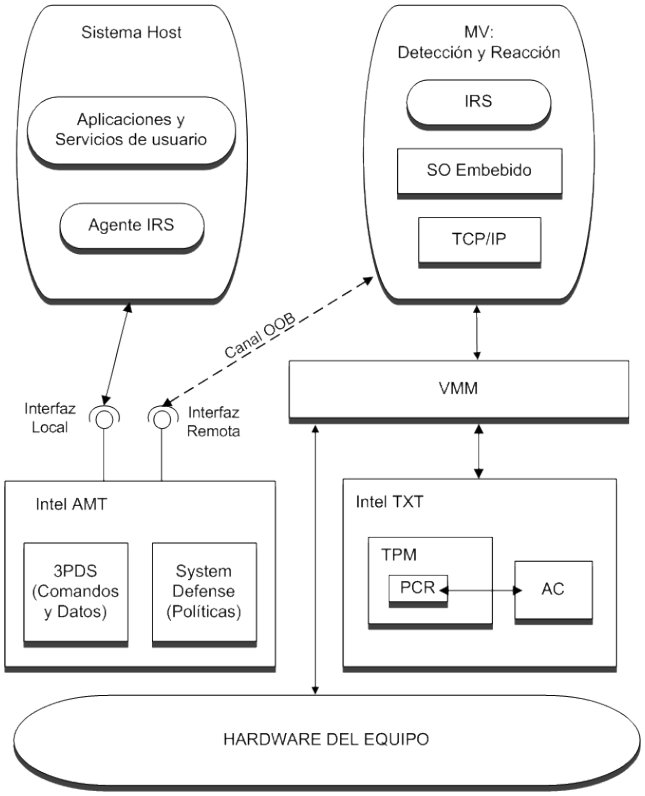

Figura 1. Arquitectura

El sistema de reacción frente a intrusiones propuesto consta de dos tipos de elementos: IRS y agente IRS. El primer componente actúa como detector de intrusiones y, además envía comandos a los equipos infectados para recuperarlos. El segundo componenete, el agente IRS, es un mecanismo local que lee los comandos enviados por los IRS y los ejecuta. Dos requisitos fundamentales perseguidos con esta estructura son proteger el sistema de comunicaciones y la validación de código para evitar la ejecución de código malicioso dentro del sistema virtualizado. Pero, además, es deseable que el sistema sea robusto. Intel vPro puede ayudar a cumplir tales objetivos usando:

- Canal OOB y Filtros Hardware. Permite que el IRS construya comandos para aplicar reglas de filtrado sobre el sistema comprometido, y que, remotamente envíe estas instrucciones al equipo comprometido. El canal OOB se representa en la Figura 1 mediante las líneas punteadas.

- 3PDS. Permite que los comandos enviados de forma remota puedan escribirse en un espacio de memoria que sigue siendo utilizable aunque el equipo se encuentre apagado o en suspensión, al igual que el canal OOB. Luego, el IRS puede enviar los datos a través del canal OOB, escribirlos en el 3PDS, y el agente IRS acceder a éstos cuando proceda.

- Entorno virtualizado. El IRS se encuentra virtualizado para permitir que, ante un posible fallo o caída del servicio éste pueda ser despertado nuevamente lo más rápido posible o migrado a otro equipo.

- Validación de código y medición del entorno. Permite mantener un sistema confiable, con características ya integradas en los equipos. El sistema permite verificar si el código IRS fue modificado.

Un inconveniente con el que nos hemos encontrado es que actualmente las implementaciones de VMM no cuentan con soporte de Intel AMT. Luego el agente IRS, que debe poder acceder a algunas características locales del dispositivo Intel AMT no puede ser virtualizado. No ocurre lo mismo con el IRS, que directamente usa la IP del canal OOB para mandar los comandos. Al descomponer el sistema en componentes para proteger al sistema de forma local (agente IRS) y componentes para la protección remota (IRS), tenemos que la solución puede aplicarse por partes, donde unos equipos actúen sólo de IRS y otros reciban órdenes, o bien la solución completa, donde un equipo que actúa como IRS también es capaz de ser restaurado en caso de amenaza por otro IRS o conjunto de IRSs. Otra ventaja de disponer del IRS virtualizado es que la máquina virtual puede ser optimizada para el IRS, instalando un mínimo conjunto de servicios requeridos por el IRS. De hecho, podemos contar con diferentes IRS en el sistema, cada uno especializado en detectar un tipo de ataque específico. Esto se haría para evitar la sobrecarga de un IRS y agilizar el proceso de toma de decisiones.

En la Figura 1, las líneas punteadas representan el acceso remoto al dispositivo Intel AMT, mientras que las líneas continuas representan accesos a características locales. Estas relaciones son explicadas mejor en la Figura 2, que ilustra el funcionamiento del sistema. Por una parte, el agente IRS puede efectuar acciones sobre el equipo comprometido tales como obtener una lista de procesos en ejecución[1] y reflejar

[1]Funcionalidad no incluida en Intel AMT.

estos datos en el espacio 3PDS del dispositivo Intel AMT. Por otra parte, el IRS puede habilitar políticas de tráfico en el dispositivo Intel AMT, incluidas el aislamiento del equipo. Destacamos aquí que esto no afecta al canal OOB, porque el filtrado de tráfico definido en este caso sería sobre la IP que conoce el sistema operativo, no la IP usada para el OOB. Como consecuencia, el atacante no podría acceder a la red, mientras que el IRS podría seguir enviando instrucciones al equipo comprometido usando el canal OOB.

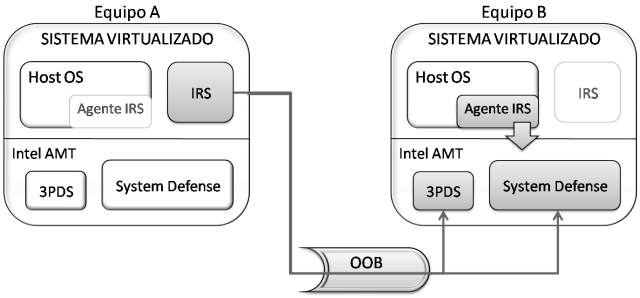

Figura 2. Elementos del Sistema de Reacción frente a Intrusiones

Un IRS puede efectuar acciones sobre el equipo comprometido usando las características remotas de Intel AMT. El principal problema es que estas características están limitadas a efectuar acciones generales sobre el equipo, y no es posible obtener información más específica del sistema. Por ejemplo, si queremos monitorizar el sistema operativo para comprobar si realmente se encuentra comprometido antes de tomar acciones correctivas no podemos usar el IRS. Para estas tareas más específicas usamos el agente IRS, que permite un control de grano fino sobre el sistema monitorizado y ejecutar acciones previamente enviadas por el IRS. Cabe destacar que, de acuerdo con la descripción de la solución propuesta, no es obligatorio que todos los equipos de la red tengan un proceso IRS, que podría estar instalado sólo en algunos equipos que actúen como IRS. Sin embargo, sí es deseable que los equipos críticos cuenten con un agente IRS para permitir su monitorización.

Un inconveniente de ejecutar el agente IRS como parte del sistema operativo es que un usuario malicioso podría interrumpir su ejecución. Para evitar esta situación es posible hacer uso de otra funcionalidad ofrecida por *System Defense* conocida como *Watchdog* [7]. Consiste en un agente confiable del sistema que puede ser configurado para aplicar una política de actuación (integrada por un conjunto de filtros) dependiente del estado de la aplicación o aplicaciones monitorizadas. Por ejemplo, la política podría ser el aislamiento del equipo mientras se dispara una alerta para avisar de que el proceso monitorizado detuvo su ejecución.

V. PROTOTIPO

En esta sección describimos la implementación de una versión preliminar del diseño anterior para demostrar el funcionamiento de la solución propuesta en un escenario de ataque real. El prototipo desarrollado implementa algunas de las principales características descritas en las secciones anteriores. Las pruebas de funcionamiento se realizaron sobre las versiones 2.6, 3.0 y 6.0 de Intel AMT. Para el desarrollo de código que use los servicios de Intel AMT mencionados hemos usado el conjunto de herramientas Intel AMT SDK [6]. Para verificar algunas de las operaciones recomendamos el uso de la herramienta Intel AMT MDTK [8], que ofrece un interfaz gráfico muy intuitivo en el que se muestran los parámetros del equipo monitorizado y los datos que estamos escribiendo en el espacio 3PDS (siempre usando una clave de administrador). Además, hemos usado Mobile C para la creación de las acciones reactivas que serán enviadas a los equipos infectados. El agente IRS está preparado para recibir este tipo de acciones y ejecutarlas. Los equipos se encuentran conectados a una red cableada. A modo de ejemplo hemos usado dos equipos: uno actúa como equipo infectado que intenta propagar un virus y el otro equipo detectará la amenaza y emprenderá las operaciones pertinentes para frenar el contagio.

Disponemos de tres componentes principales:

- **Mecanismo de detección.** Captura y analiza el tráfico de red en busca de evidencias de ataques. Con este fin hemos utilizado Snort para la detección de ataques, desacoplando el módulo de detección de intrusiones del módulo de reacción frente a éstas.

- **Mecanismo de reacción.** Analiza las alertas recogidas durante la fase de detección y decide cuáles son los siguientes pasos a ser tomados. Hemos desarrollado una aplicación IRS que recibe alertas del sistema Snort y selecciona una serie de pasos de respuesta a la incidencia, que serán enviados a los equipos comprometidos de la red.

- **Agente Local**. Es la aplicación que se ejecuta en el equipo comprometido. Es usada cuando un IRS necesita ejecutar comandos en el sistema host para verificar que fue comprometido. Además, hemos desarrollado el agente IRS de forma que no sólo obtiene evidencias de ataques, sino que es capaz de actuar sobre el sistema para, por ejemplo, eliminar el código malicioso. Por lo que, el agente IRS permite aplicar un control más específico sobre el equipo comprometido, evitando en algunos casos la ejecución de políticas más radicales como el reinicio del equipo[2]. Hemos usado Mobile C para crear una librería de código móvil que será enviado por el IRS y ejecutado localmente por el agente IRS, y el espacio 3PDS para la escritura de este código.

Cuando un IRS detecta un indicio de propagación de virus proveniente de un equipo de la red, una serie de acciones son almacenadas en el espacio 3PDS del equipo comprometido para obtener información, solucionar el problema, extraer

[2]Es posible aplicar políticas de respuesta más generales desde el IRS, pero aquí queremos evitar que el equipo quede inutilizable o que el usuario se percate de nuestra presencia.

evidencias o bien modificar algunas políticas del mecanismo de reacción. Cabe destacar, que, como el espacio 3PDS sigue accesible aunque el equipo esté apagado o suspendido, el usuario puede apagar el equipo y el IRS obtener tales evidencias registradas del mismo modo que si estuviese encendido. Tras esto, el IRS puede escribir los comandos de reacción en el 3PDS para que, al iniciarse el equipo el agente IRS los ejecute, pero, además también puede utilizar las políticas de *System Defense* para efectuar acciones sobre el equipo como aislarlo o restringir cierto tráfico de entrada (filtrado hardware). Para validar la solución propuesta hemos implementado tres acciones de reacción que son efectuadas por el agente IRS sobre el sistema host: la generación de una lista de procesos en ejecución, consultas al registro del sistema en busca de ciertos valores de clave almacenados en éste, y ejecutar los comandos almacenados en la memoria 3PDS. Todas estas acciones pueden ser ejecutadas automáticamente en nuestro sistema, porque asumimos que puede ser así. En otros sistemas, puede ser más aconsejable derivar algunas alertas hacia algún administrador que tome la decisión final. En este caso, hemos considerado que este tipo de infección de un virus que intenta propagarse puede ser resuelto automáticamente en nuestro entorno.

En el escenario propuesto para la validación de la solución, hemos infectado una máquina con el virus Sasser que intentará infectar a otros equipos en la red. El mecanismo de detección generará una alerta y el IRS reaccionará a este incidente aislando el equipo comprometido, enviando comandos para la obtención de evidencias y desinfectando el sistema, tras lo que podrá habilitar nuevamente el equipo afectado como parte de la red. La Figura 3 muestra los pasos seguidos a tal fin.

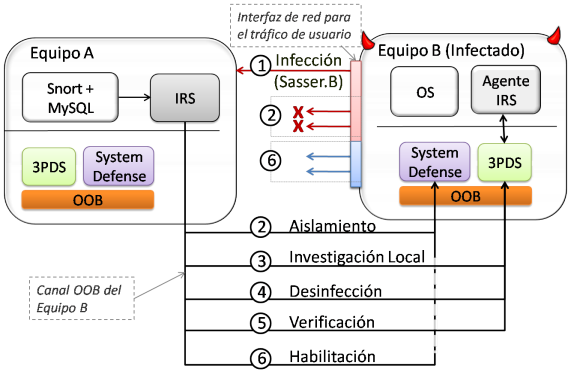

Figura 3. Infeccion/Desinfección virus Sasser

El sistema Snort identifica un intento de propagación de virus (1) y eleva una alerta al IRS. El IRS procesa la alerta enviada por Snort, que informa de un posible virus Sasser intentando propagarse. Entonces, el IRS consulta su base de datos local para extraer la IP del dispositivo Intel AMT que pertenece al sistema comprometido. Tras esto, el IRS construye una política que aísla el equipo comprometido remotamente y la envía al servicio *System Defense* (2). Como consecuencia, sólo el tráfico por el canal OOB es permitido. Para adquirir información de la naturaleza del virus, le IRS escribe comandos en la memoria 3PDS del equipo infectado para extraer evidencias de actividades maliciosas (3). Como resultado, el IRS tendrá un conocimiento más detallado sobre el virus y estará preparado para seleccionar el mecanismo de respuesta más adecuado para resolver el incidente. Tras analizar los resultados de los comandos ejecutados por el agente IRS, el IRS almacenará en el espacio 3PDS un programa de desinfección y comandos para su ejecución (4). El agente IRS extraerá este programa y lo ejecutará. Tras esto, el IRS envía comandos para comprobar si la desinfección fue ejecutada con éxito analizando tanto los procesos en ejecución como el registro del sistema (5). Tras validar que no existen trazas de la actividad del virus, la política de aislamiento puede ser desactivada y el equipo desinfectado puede ser parte de la red nuevamente (6).

Por ejemplo, en el paso 3, el código siguiente sería almacenado en el espacio 3PDS:

$$
\begin{aligned}
&\#include < stdio.h > \\
&\_irs\_getProcesses(); \\
&\_irs\_queryRegistry(\text{“}SOFTWARE/MicrosoftWindows \\
&/CurrentVersion/Run\text{''}) \\
\end{aligned}
$$

Como puede comprobarse, el código anterior es bastante simple, y esto supone una clara ventaja de la solución. El objetivo de buscar esta simplicidad en los mensajes es doble: no sobrecargar el canal y agilizar la construcción automática del código. Respecto a esto último, la idea es que el IRS construya este código automáticamente en función del ataque detectado. Para que el código anterior funcione, las funciones empleadas (ej. *getProcesses* y *queryRegistry*) deben ser compiladas como parte del intérprete CH de Mobile C. En nuestro ejemplo, tras ejecutar el código anterior, usando el intérprete CH en el equipo comprometido, una lista de procesos es obtenida y esta información puede ayudar al IRS en la selección de un proceso de limpieza automático para un virus particular, en este caso para el virus Sasser.

Para probar la seguridad el canal OOB, hemos usado la herramienta *Wireshark* para capturar tráfico desde equipos con la tecnología Intel AMT. Los resultados prueban que los equipos con Intel AMT no visualizan el tráfico del canal OOB y, además, gracias a las reglas de filtrado hardware, podrían configurarse para no tener acceso a ninguno de los canales OOB del resto de dispositivos Intel AMT de la red. Sin embargo, por defecto, cualquier equipo puede acceder al resto de canales OOB. Luego, por defecto, un equipo Intel AMT sólo filtra el tráfico dirigido a la IP del dispositivo local Intel AMT, luego el tráfico intercambiado por el canal OOB puede ser accedido por terceras partes. El problema podría ser parcialmente resuelto usando TLS en la comunicación con los dispositivos Intel AMT. Sin embargo, para asegurar que el tráfico OOB no es visible por ningún equipo no autorizado todos los equipos de la red deberían contar con Intel vPro, o bien tecnologías similares que permitan filtrado hardware del tráfico saliente/entrante y gestión remota.

Algunos errores difíciles de solventar, son que, eventualmente durante las pruebas se dieron casos de colisión por el acceso múltiple de varias aplicaciones al espacio 3PDS, y, en ocasiones la escritura en el espacio 3PDS ocurrió con segundos de retardo. Sin embargo, debido a que no contamos con emuladores que permitan simular una red de más nodos, con diferentes características y funcionalidades, extraer conclusiones más amplias sobre esta cuestión no es factible.

VI. Conclusiones y trabajo futuro

En este artículo hemos planteado un sistema colaborativo para la detección y reacción ante intrusiones, empleando como base las tecnologías presentes en Intel vPro. Un factor básico para lograr este objetivo es proteger las comunicaciones entre los diferentes sistemas reactivos que componen el sistema cuando las acciones de respuesta son enviadas a los equipos afectados. La solución propuesta aquí permite dicho intercambio de datos incluso cuando el equipo afectado está apagado, suspendido o en otro estado donde el sistema operativo está inoperativo, usando para esto las características presentes en la tecnología Intel vPro. Además, el esquema propuesto permite la virtualización del IRS para proteger los mecanismos de detección y respuesta , así como obtener mejor rendimiento con el uso de instrucciones específicas de procesador. Los resultados preliminares muestran que la solución presentada resuelve eficazmente el caso de prueba planteado. Estos resultados son aplicables a casos de prueba similares, donde otros tipos de virus de propagación intentan infectar la red y las respuestas automáticas son factibles. Intel vPro ha demostrado ser una herramienta muy útil para establecer el canal de comunicaciones seguro entre los módulos del sistema de reacción. Sin embargo, a pesar del conjunto de herramientas tan amplio para el desarrollo de aplicaciones que usen Intel vPro, la carencia de emuladores que permitan validar virtualmente la solución impide obtener resultados estadísticos sobre el rendimiento general de la solución y sólo podemos basarnos en las pruebas de laboratorio efectuadas con equipos reales. Consideramos que este es un punto clave para fomentar el desarrollo de aplicaciones que utilicen este tipo de herramientas, en especial aquellas dirigidas a su implantación en entornos empresariales. Otras cuestiones abiertas que debieran solventarse son la evaluación del riesgo de colisión en el acceso al espacio 3PDS y el tiempo de retardo por escritura en éste.

VII. Agradecimientos

Este trabajo ha sido parcialmente subvencionado por el Ministerio de Ciencia e Innovación a través de los proyectos SPRINT (TIN2009-09237) y ARES (CSD2007-00004). Adicionalmente, ha sido financiado por la Junta de Andalucía a través del proyecto PISCIS (TIC-6334). El primer autor ha sido subvencionado por el Programa FPI.

Referencias

[1] Intel virtualization technology: Hardware support for efficient processor virtualization. Technical report, Intel Technology Journal, 2006.

[2] Intel trusted execution technology. Technical report, Technology Overview, 2007.

[3] Understanding full virtualization, paravirtualization, and hardware assist. Technical report, VMware, 2007.

[4] Intel active management technology overview. Technical report, Intel, 2008.

[5] Architecture guide: Intel active management technology. Technical report, Intel, 2009.

[6] Intel active management technology (intel amt) software development kit (sdk) start here guide. Technical report, Intel, 2009.

[7] Intel active management technology network interface guide. Technical report, Intel, 2010.

[8] Intel manageability developer tool kit. Technical report, Intel, 2010.

[9] M. Asaka, A. Taguchi, and S. Goto. The implementation of ida: An intrusion detection agent system. In *In 11th Annual FIRST Conference on Computer Security Incident Handling and Response*, 1999.

[10] Min Cai, Kai Hwang, Yu-Kwong Kwok, Shanshan Song, and Yu Chen. Collaborative internet worm containment. *IEEE Security and Privacy*, 3:25–33, May 2005.

[11] S.P. Chung and A.K. Mok. Collaborative intrusion prevention. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2007. WETICE 2007. 16th IEEE International Workshops on*, pages 395 –400, 2007.

[12] Arjita Ghosh and Sandip Sen. Agent-based distributed intrusion alert system. In Arunabha Sen, Nabanita Das, Sajal Das, and Bhabani Sinha, editors, *Distributed Computing - IWDC 2004*, volume 3326 of *Lecture Notes in Computer Science*, pages 7–47. Springer Berlin / Heidelberg, 2005.

[13] Guy Helmer, Johnny Wong, Mark Slagell, Vasant Honavar, Les Miller, Yanxin Wang, Xia Wang, and Natalia Stakhanova. Software fault tree and coloured petri net based specification, design and implementation of agent-based intrusion detection systems. *Int. J. Inf. Comput. Secur.*, 1:109–142, January 2007.

[14] R. Janakiraman, M. Waldvogel, and Qi Zhang. Indra: a peer-to-peer approach to network intrusion detection and prevention. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, pages 226 – 231, 2003.

[15] Hai Jin, Zhiling Yang, Jianhua Sun, Xuping Tu, and Zongfen Han. Cips: Coordinated intrusion prevention system. In Cheeha Kim, editor, *Information Networking*, volume 3391 of *Lecture Notes in Computer Science*, pages 89–98. Springer Berlin / Heidelberg, 2005.

[16] C. Kruegel and T. Toth. Distributed pattern detection for intrusion detection. In *Network and Distributed System Security (NDSS) Symposium*, Reston, Virginia 20190, USA, 2002. Internet Society.

[17] Nita Patil, Chhaya Das, Shreya Patankar, and Kshitija Pol. Analysis of distributed intrusion detection systems using mobile agents. In *Proceedings of the 2008 First International Conference on Emerging Trends in Engineering and Technology*, pages 1255–1260, Washington, DC, USA, 2008. IEEE Computer Society.

[18] José Duarte De Queiroz, Luiz Fern, Rust Da Costa Carmo, and Luci Pirmez. Micael: An autonomous mobile agent system to protect new generation networked applications. In *In 2nd Annual Workshop on Recent Advances in Intrusion Detection*, Purdue, USA, 1999.

[19] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. Internet besieged. chapter DIDS (distributed intrusion detection system)-motivation, architecture, and an early prototype, pages 211–227. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998.

[20] Kun Xiao, Ji Zheng, Xin Wang, and Xiangyang Xue. A novel peer-to-peer intrusion detection system. In *Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, PDCAT '05, pages 441–445, Washington, DC, USA, 2005. IEEE Computer Society.

[21] Vinod Yegneswaran, Paul Barford, and Somesh Jha. Global intrusion detection in the domino overlay system. In *In Proc. of the Network and Distributed System Security Symposium*, February 2004.

[22] Chenfeng Vincent Zhou, Leckie C., Karunasekara S., and Tao Peng. A self-healing, self-protecting collaborative intrusion detection architecture to trace-back fast-flux phishing domains. In *Networks Operations and Management Symposium Workshops*, Salvador, Brazil, 2008. IEEE.