CyberSPL: Plataforma para la verificación del cumplimiento de políticas de ciberseguridad en configuraciones de sistemas usando modelos de caracterısticas

A. J. Varela-Vaca, Rafael M. Gasca, Rafael Ceballos, y Pedro Bernáldez Torres
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla, Spain
{ajvarela, gasca, ceball, pedbertor}@.us.es

***Resumen*—Los ataques de ciberseguridad se han convertido en un factor muy relevante que pueden contravenir el cumplimiento de las políticas de ciberseguridad de las empresas y organizaciones. Dichos ataques pueden estar provocados en gran medida por una ausencia de configuraciones de seguridad o de valores por defecto en la configuración de productos y sistemas. La complejidad en la configuración de productos y sistemas es un reto en la industria del software. En este artículo proponemos una plataforma, Cybersecurity Software Product Line (CyberSPL), basado en la metodología de diseño de líneas de productos de tal manera que a través de la definición de modelos de características podamos agrupar patrones de configuraciones de aplicaciones y sistemas relacionados con la ciberseguridad. Mediante el análisis automatizado de estos modelos permitiríamos la diagnosis de los posibles problemas en las configuraciones de seguridad y por tanto evitarlos. Como soporte para dicha plataforma se ha implementado una solución multiusario y multiplataforma que permite definir un catálogo de modelos de características público o privado. Además se han integrado mecanismos para determinar todas las configuraciones de un modelo, detectar si una configuración es correcta o no, además de diagnosticar las causas de fallos dada una configuración determinada. Para validar la propuesta se usará un escenario real donde se plantea la configuración de un canal seguro de transmisión mediante el protocolo SSL/TLS, aplicado a un servidor de aplicaciones. En dicho escenario se analizaran dos modelos de características, se validarán diferentes configuraciones, y se diagnosticarán varias configuraciones con problemas.**

***Index Terms*—configuraciones de seguridad, configurabilidad, cumplimiento, políticas de seguridad, modelos de características, automatización**

Tipo de contribución: *Investigación original*

I. Introducción

El notable aumento de ataques en ciberseguridad ha conducido a los investigadores a utilizar técnicas más eficaces para mitigar el impacto de dichos ataques. En muchos casos, estos ataques pueden estar provocados por una ausencia de configuraciones de seguridad, o también por unos valores por defecto en la configuración de productos y sistemas que no están de acuerdo con las políticas de ciberseguridad establecidas. La complejidad en la configuración de productos y sistemas es un reto en la industria del software. Un ejemplo de herramienta de configuración es el KConfig [10] donde los desarrolladores pueden seleccionar entre más de 12.000 opciones de configuración del Kernel de Linux. Una configuración errónea o inadecuada puede desencadenar en problemas de seguridad como por ejemplo ataques por debilidades de la propia configuración.

Entre las técnicas que se utilizan para detectar ataques de ciberseguridad podemos destacar las relacionadas con líneas de productos (Software Product Lines) [6], más concretamente, el uso de técnicas de análisis de variabilidad o análisis de características (Feature-oriented Domain Analysis) [7]. Otros trabajos relacionados con la ciberseguridad son la extracción y selección de características de ficheros de logs [1] para detectar amenazas y ataques de ciberseguridad mediante machine learning, o la verificación de seguridad de las configuraciones de aplicaciones móviles mediante la técnica de model checking parcial [2], y la aproximación orientada a características para la construcción modular de árboles de fallos, que posibilita la reutilización de las estructuras de caminos de propagación de fallos [3].

En nuestro caso, dado los beneficios probados que han tenido las técnicas orientadas a modelos de características en el desarrollo de líneas de productos software, en este artículo proponemos su uso para favorecer el cumplimiento de políticas de ciberseguridad, comprobando las adecuadas configuraciones de los sistemas y productos que la gestionan. También la podríamos considerar muy convenientes como una aproximación de diseño de los productos y sistemas de ciberseguridad basada en modelos (Model-based design) de acuerdo a los requisitos de ciberseguridad exigidos en las políticas de diseño seguro de los mismos.

Los modelos de características (feature models) y las características (features) [7] son el principal concepto de la descomposición funcional de la aproximación de líneas de producto, por tanto, podemos considerar estos modelos para representar los parámetros de configuración de acuerdo a las políticas establecidas. Además, estos modelos permiten expresar restricciones y atributos entre las características de las configuraciones, tal como se ha realizado en un trabajo previo [4]. Permitiendo una mayor expresividad de las dependencias y relaciones entre las diferentes características de las configuraciones de sistemas y productos relacionados con la ciberseguridad.

Una vez las configuraciones correctas se han hecho explíci-

tas en un modelo de características, podemos aplicar técnicas automáticas de verificación formal [8]. Estos mecanismos de verificación formal nos ayudarán a determinar el cumplimiento o no de las políticas o diseños de productos (*Detección de fallos en configuración*) y también identificar los posibles incumplimientos o configuraciones de los sistemas y productos (*Diagnosis de fallos de configuración*) que no cumplen las políticas.

También debemos tener en cuenta que la naturaleza de los modelos de ciberseguridad son altamente dependientes del contexto [11]. Por tanto, los modelos de características deberán ser adaptados en función de los objetivos y los contextos a aplicar para las políticas de cumplimiento [12].

Derivado de todo esto, en este artículo se presenta una propuesta de plataforma que cubre los siguientes objetivos:

- **OBJ1.** Facilitar la definición por parte de los usuarios de un catálogo de políticas de ciberseguridad a través de modelos de características asociados a las diferentes contextos de productos y sistemas relacionados con la ciberseguridad.
- **OBJ2.** Automatizar la derivación de propiedades de los modelos de características. Por ejemplo una propiedad puede ser comprobar la corrección del modelo, es decir, indicar si a partir del modelo se puede extraer algún producto. Otro ejemplo podría ser la extracción de todas las configuraciones de productos o sistemas admitidos por la política de ciberseguridad.
- **OBJ3.** Automatizar la validación del cumplimiento de políticas de ciberseguridad a través de la descripción de características de los diferentes contextos de ciberseguridad que se disponen.
- **OBJ4.** Automatizar el diagnóstico de configuraciones para determinar la causa o causas del incumplimiento de políticas de ciberseguridad a través de la identificación de los fallos en las configuraciones establecidas.
- **OBJ5.** Validación de la propuesta mediante casos de uso complejos. Como por ejemplo la configuración de mecanismos de ciberseguridad de un servidor de aplicaciones web.

El artículo se ha dividido en varias secciones de acuerdo con todo lo anteriormente expuesto. En la sección II se hace una pequeña introducción a los modelos de características y del análisis de estos. En la sección III se presenta nuestra propuesta, incluyendo la arquitectura seguida, el flujo de trabajo establecido, y una descripción del funcionamiento en un escenario real. En la sección IV se expondrá un caso de uso, y se detallarán los resultados obtenidos de la experimentación con dicho caso. En la sección V daremos una perspectiva general de trabajos relacionados. Se finalizará con las conclusiones y los trabajos futuros.

II. Modelos de características

Los modelos de características son el método más común para el análisis de líneas de productos software. Un modelo de características generalmente se representa gráficamente mediante diagramas que definen características y sus relaciones, tal como se puede ver en el ejemplo de la Fig. 1. Este modelo de características puede representar todas las configuraciones de un producto o sistema que cumplan una determinada

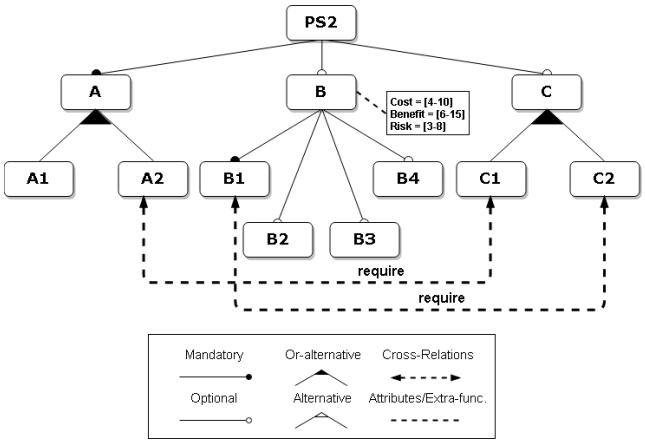

Figura 1: Modelo de características de ejemplo.

política de ciberseguridad. En este ejemplo podemos ver como la características $PS2$ y A siempre deberán aparecer en nuestros productos o sistemas, ya que el elemento raíz siempre debe aparecer y la característica A es obligatoria siempre que aparece $PS2$. Por otro lado, B y C son opcionales, es decir, podrán o no aparecer en las configuraciones. También podemos expresar opcionalidad, como por ejemplo, si está la característica A en nuestras configuraciones podrán aparecer $A1$ ó $A2$ ó ambas. Existen otras dependencias que expresan restricciones cruzadas entre características, por ejemplo, si aparece la característica $A2$ implica que debe aparecer también la característica $C1$.

Además estos modelos pueden ser extendidos con propiedades y funciones extra que aportan información adicional al modelo. Por ejemplo, como se puede observar en Fig. 1, el modelo tiene asociado ciertos atributos a la característica B, que son el $Cost$, $Benefit$, y $Risk$. Estos atributos se han definido de tipo entero con un dominio como el que se indica en la imagen. Esto quiere decir que si aparece la característica B en alguna configuración, dicha característica puede tener asociado estos atributos con algún valor de entre los que se indican en el modelo.

Estos modelos son la base del análisis de una línea de productos e intentan representar el espacio de posibles soluciones. Con estos modelos podemos inferir cierta información, como por ejemplo, el número total de posibles productos/sistemas válidos, si un producto/sistema concreto es válido o no, o si cierta configuración de productos puede ser válida o no. A modo de ejemplo, a continuación mostramos todas las configuraciones obtenidas del modelo de la Fig. 1:

```
Configuracion1 = "PS2;A;A1;",
Configuracion2 = "PS2;C;C2;A;A1;",
Configuracion3 = "PS2;C;C1;A;A1;",
Configuracion4 = "PS2;C;C1;C2;A;A1;",
Configuracion5 = "PS2;C;C1;A;A2;",
Configuracion6 = "PS2;C;C1;A;A1;A2;",
Configuracion7 = "PS2;C;C1;C2;A;A2;",
Configuracion8 = "PS2;C;C1;C2;A;A1;A2;"
```

Por cada configuración se indican las características seleccionadas, por ejemplo, la $Configuracion1$ se han seleccionado las características $PS1$, A, y $A1$. Esto quiere decir que el producto resultante está compuesto de estas características.

Para el análisis automatizado de estos modelos se suelen

utilizar métodos formales [8] basados en lógica proposicional, lógica descriptiva o programación con restricciones. En la literatura se han definido herramientas como FAMA tool [13] o [4] que hacen transformaciones directas de los modelos de característica a modelos de satisfacción de restricciones (Constraint Satisfaction Problem, CSP), y modelos de optimización con restricciones (Constraint Optimization Problem, COP). También, para estos problemas, si los dominios de las variables son de tipo boolean (verdadero o falso), se pueden aplicar resolutores del tipo SAT Solver que facilitan el modelado y la eficiencia en la resolución de los mismos.

El principal objetivo de los CSP para modelos de características es encontrar la satisfacción del modelo, o calcular el número de productos válidos. Mientras que los COP persiguen la optimización de ciertas funciones. En el código 1 podemos ver un ejemplo de modelo CSP válido para la herramienta ChocoSolver [25] y obtenido para el ejemplo de la Fig. 1.

```
==== VARIABLES ====
B1 [0, 1], A [0, 1], B2 [0, 1], A2 [0, 1], C1 [0, 1]
PS2 [0, 1], B [0, 1], B.cost [0, 10], B.risk [0, 8],
B.benefit [0, 15], B3 [0, 1], A1 [0, 1], B4 [0, 1]
C [0, 1], C2 [0, 1], rel-4_card [1, 2], rel-9_card [1, 2]
S-B1 [0, 1], S-B2 [0, 1], D-A [0, 1], S-A2 [0, 1],
S-C1 [0, 1], S-B3 [0, 1], S-B [0, 1], S-B.cost [0, 1],
S-B.risk [0, 1], S-B.benefit [0, 1], S-PS2 [0, 1],
S-A1 [0, 1], S-B4 [0, 1],S-C [0, 1],S-C2 [0, 1]
==== CONSTRAINTS ====
ifonlyif({PS2[0,1],cst[1],A[0,1],cst[1]})
implies({B[0,1],cst[0],B2[0,1],cst[0]})
ifonlyif({S-C1[0,1],cst[1],C1[0,1],cst[1]})
ifthenelse({C[0,1],cst[0],C1[0,1]C2[0,1],rel-9_card[1,2],
   C1[0,1]C2[0,1],cst[0]})
ifthenelse({B[0,1],cst[1],B.risk[0,8],cst[1090519040],
   B.risk[0,8],cst[1077936128],B.risk[0,8],cst[0]})
ifonlyif({S-B.benefit[0,1],cst[1],B.benefit[0,15],cst[0]})
ifonlyif({S-B1[0,1],cst[1],B1[0,1],cst[1]})
ifonlyif({S-C2[0,1],cst[1],C2[0,1],cst[1]})
...
```

Código 1: Trozo de código del modelo de CSP de ChocoSolver.

En Tab. I presentamos los datos más relevantes del modelo de ejemplo de la Fig. 1. Se indica el número de características y la cantidad de relaciones de cada tipo. También aparece el resultado de aplicar dos operaciones sobre el modelo: la comprobación de si el modelo es válido (símbolo ✓), y el cálculo de todas las configuraciones posibles del modelo.

Tabla I: Datos extraído del modelo de ejemplo.

Numero de características	12
Mandatory	2
Optional	5
OR	2
XOR	0
Attributes	1
Cross-Relations	2
Válido	✓
Número de configuraciones	8

III. CyberSPL: Cybersecurity Software Product Line

CyberSPL está enfocado a la verificación del cumplimiento de las políticas de ciberseguridad. El flujo de trabajo de esta plataforma se basa en la ejecución del proceso de negocio que se describe en la Fig. 2.

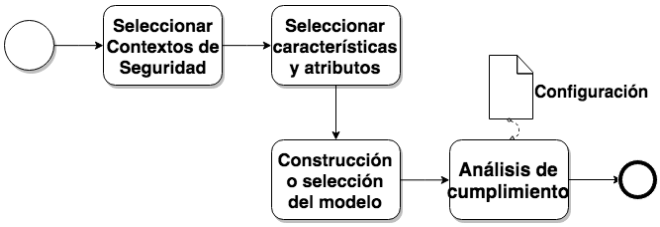

Figura 2: Proceso de verificación del cumplimiento de las políticas.

Este proceso director de CyberSPL nos permitirá evaluar los diferentes contextos de ciberseguridad de una organización. En primer lugar, se *seleccionará el contexto de ciberseguridad* a analizar y se *seleccionarán las características y atributos* de las mismas que son permitidos según la política de ciberseguridad. A continuación, a partir de ellos se construirán o se seleccionarán un catálogo de modelos de características y se procederá a realizar los correspondientes análisis de los productos o servicios que se pretenden verificar de acuerdo a la política establecida, esto se representa mediante la tarea *análisis de cumplimiento*. Es muy importante en la ejecución de este proceso de análisis que los modelos de características que se utilicen estén totalmente actualizados con los últimos detalles de la política de ciberseguridad establecida por la organización.

La *selección de los contextos de seguridad* y la *selección de características y atributos* es algo que tendrá que establecerse en la política de ciberseguridad y será necesario un trabajo de análisis [12] previo a la definición de los modelos de características. Por ejemplo, podemos seleccionar un contexto de despliegue de servicios de una organización que estará basado en un servidor de aplicaciones. El servidor podría ser *Apache Tomcat*, y nuestra política de seguridad indicaría que las comunicaciones hacia el servidor deben ser totalmente seguras a través de la configuración de canales basados en Secure Socket Layer (SSL) [14] y/o Transport Layer Security (TLS) [15][16].

Como se ha indicado en el proceso tendremos que, o bien construir un modelo si no existe, o seleccionar un modelo de características de los existentes en un catálogo. CyberSPL se ha provisto de mecanismos para tanto la construcción de modelos de características nuevos en formato FAMA [13], como para la selección de modelos de un catálogo de modelos de características, de tal manera que el esfuerzo de desarrollo y análisis se facilite a los gestores. En un trabajo previo [4] fue definido un catálogo de modelos de características para la configuración de controles de seguridad para un motor de procesos. Actualmente, CyberSPL no sólo nos proporciona mecanismos de modelado para la construcción de modelos de características, sino que también facilita la persistencia y actualización de los mismos mediante un catálogo de modelos. Dicho catalogo se puede compartir, permitiendo así reutilizar el conocimiento recogido en los modelos.

En la Fig. 3 se muestra el flujo de trabajo de CyberSPL usando el catálogo de modelos. En este caso, seleccionaríamos un modelo dependiendo del contexto y de la política indicada, y en un paso posterior se podría reajustar el modelo con las características y atributos adecuados. Este modelo reajustado

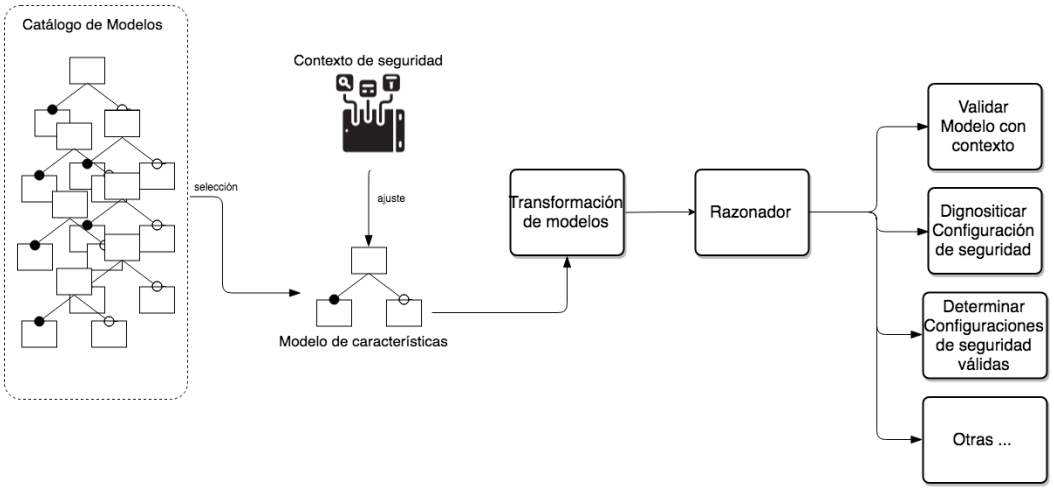

Figura 3: Flujo de trabajo de CyberSPL con el catálogo de modelos.

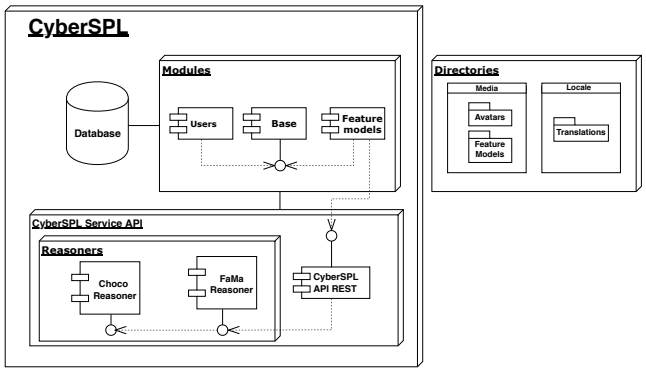

Figura 4: Arquitectura de CyberSPL.

pasaría por una transformación en la que se generaría un modelo formal basado en restricciones, tal como se indicó en la sección II. Usando un razonador ajustado al modelo obtenido podemos aplicar el *análisis de cumplimiento* que necesitemos. Como por ejemplo validar el modelo según el contexto, diagnosticar una configuración de seguridad determinada, determinar todas las configuraciones válidas, u otras operaciones.

CyberSPL ha sido implementado siguiendo la arquitectura de la Fig. 4. Dicha implementación es modular donde se integran un módulo de gestión de usuarios, un módulo base de gestión de la plataforma, y un módulo para gestión de modelos de características (feature models). CyberSPL se ha integrado con un API REST que le permiten consumir FAMA tools junto con el razonador de ChocoSolver, para realizar el análisis y razonamiento de los modelos que se vayan creando.

CyberSPL está pensado para ser una solución web multiplataforma y multiusuario. Es decir, cualquier usuario puede registrarse desde un navegador web y utilizarla. Cualquier usuario puede ver los modelos públicos disponibles en CyberSPL y de manera interna a través de su perfil puede definir su propio catálogo de modelos tal y como se muestra en la Fig. 5. En la misma figura podemos ver como es la edición de un modelo de CyberSPL. Los modelos que cada usuario genere se pueden indicar como privados, para uso exclusivo del usuario, o como públicos, para ser consultados y usados por cualquier otro usuario.

Sobre cada modelo se puede actuar editándolo, borrándolo, ó realizando un análisis. La sección de análisis de cada modelo provee al usuario de tres opciones principales como se puede observar en la Fig. 6 (aparece difuminado en el fondo): (i) validar modelo, (ii) diagnosticar una configuración que responde a una política de ciberseguridad establecida, y (iii) obtener todas las configuraciones válidas para una política de ciberseguridad establecida. En la Fig. 6 se ha seleccionado la opción de diagnosticar una configuración (resaltado con un modal), donde CyberSPL da al gestor de ciberseguridad la oportunidad de especificar cierta configuración para ser cruzada con el modelo, que en este caso se corresponde con el modelo que ya fue presentado previamente en la Fig. 1. En la figura también se puede observar que en dicha sección tendremos disponible una consola donde se irán mostrando al usuario los resultados de las diferentes operaciones realizadas sobre el modelo. En la Fig. 6 se observa el listado de las configuraciones obtenidas para el modelo. Además CyberSPL da la opción de guardar dichas configuraciones de manera externa en formato JSON, usando la opción *Exportar configuraciones* que se encuentra debajo de la opción de obtener todas las configuraciones.

IV. Caso de uso y datos de experimentación

Para poder analizar las ventajas de aplicar y usar CyberSPL vamos a utilizar el caso de uso planteado en la Fig. 7. Dicho caso de uso representa un contexto de ciberseguridad real, donde una organización tiene un conjunto de usuarios o empleados que usan los servicios de la red corporativa de la organización, a través de un servidor de aplicaciones *Apache* que actúa de proxy. En la política de ciberseguridad se establece que las conexiones a la red corporativa deben hacerse de manera segura. Por tanto, en dicho servidor se pueden establecer ciertas configuraciones de seguridad para asegurar el canal a través del uso del protocolo *SSL/TLS*. En la Fig. 7 podemos observar una parte de la configuración de un servidor *Apache*, donde se han establecido ciertas características de la comunicación segura tales como versiones del protocolo o incluso tamaños de claves permitidos.

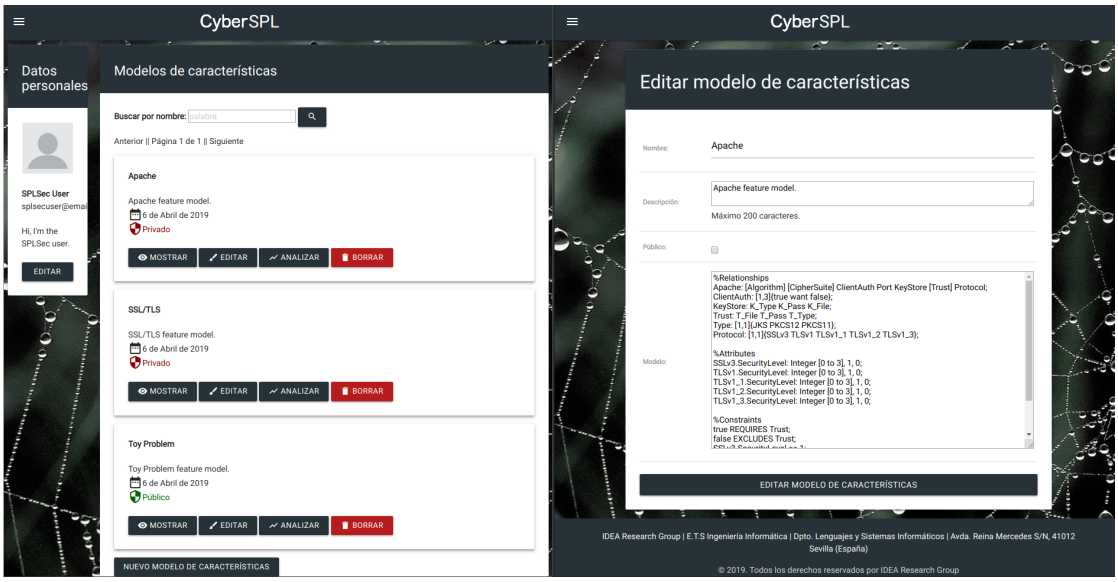

Figura 5: Catálogo de modelos de un usuario.

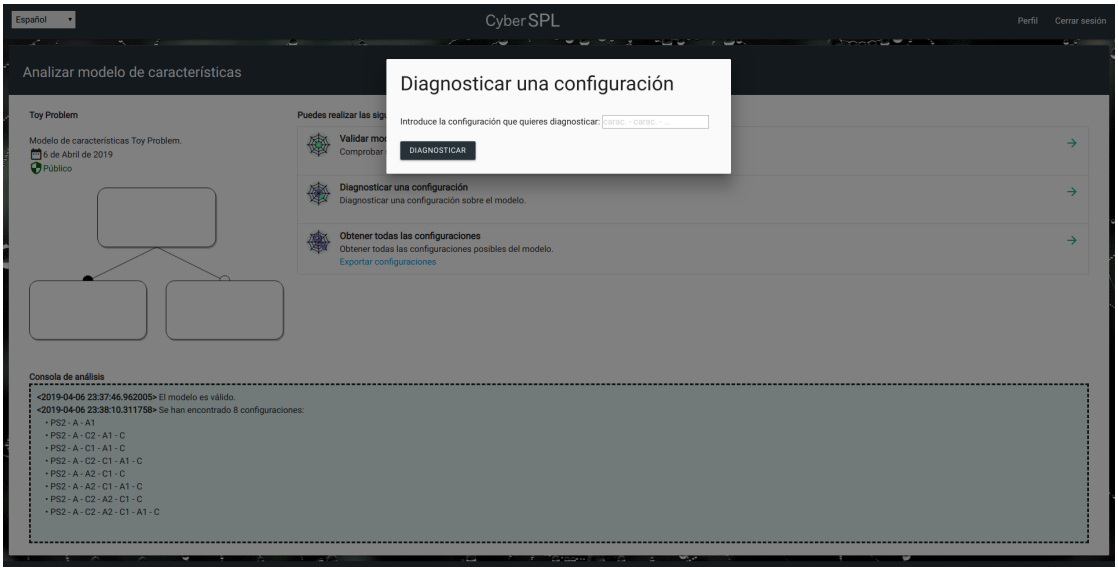

Figura 6: Sección de análisis de modelos.

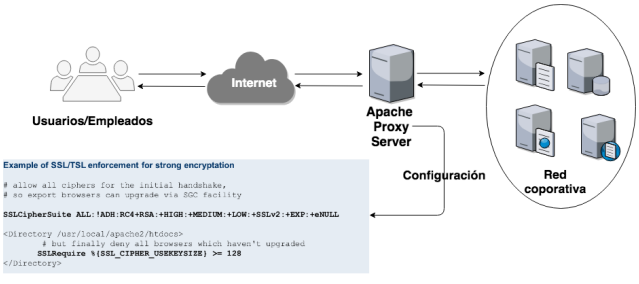

Figura 7: Caso de uso.

IV-A. Descripción de contexto y modelos de característica

Para analizar el contexto de ciberseguridad vamos a usar dos modelos características: (1) modelo de *Apache*, y (2) modelo de *SSL/TLS*. Ambos modelos son una evolución actualizada de los presentados en [4], donde se han eliminado las características consideradas inseguras y se han añadido otros parámetros como los tamaños de claves en el caso del modelo de SSL/TLS.

El primer modelo que usaremos es el presentado en la Fig. 8, que muestra características configurables de seguridad de un servidor *Apache* con respecto a *SSL/TLS*. El protocolo *SSL/TLS* se basa en el handshake cuya secuencia de pasos son: (1) Negociación del *Cipher Suite* a usar durante la transferencia, y generación e intercambio de un número aleatorio (master key); (2) Establecer e intercambiar un identificador de sesión entre cliente y servidor; (3) Autenticar al servidor frente al cliente; (4) Autenticar al cliente frente al servidor. Este handshake se ha simplificado en la nueva versión *TLSv1.3*. SSL/TL permite autenticar tanto cliente como servidor, y la comunicación anónima. La autenticación se hace a través de firmas digitales como certificados o claves. En el caso

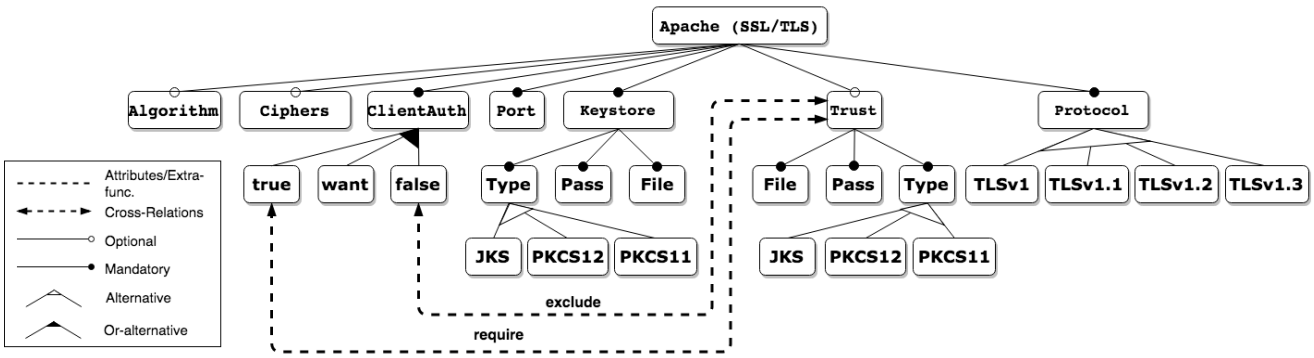

Figura 8: Modelo de características de Apache.

de certificados estos siempre se cruzan con autoridades de certificación (CA). Por otra parte, estos protocolos también permiten la autenticación anónima usando intercambio de claves *Diffie-Hellman* en los protocolos *SSLv3* y *TLSv1.0*.

En el modelo podemos observar como hay ciertos parámetros como qué versiones del protocolo (*Protocol*) se pueden configurar. En este caso hemos descartado aquellas versiones de protocolo que ya están desaconsejadas como es el caso de SSLv3. Hemos querido mantener versiones débiles como *TLSv1* ó *TLSv1.1*, porque aún se siguen usando y porque su cese está programado para el año 2020, mientras tanto hay proveedores y clientes que soporten dichos protocolos. Por otro lado, si usamos autenticación en cliente (*ClientAuth*) esto va a requerir que usemos cierta infraestructura como almacenes (*Keystore* y *Trust*) y certificados. Además existirá la posibilidad de establecer para ciertos protocolos las *Ciphers* aceptadas. La configuración de los *Ciphers* nos llevo a realizar un análisis en profundidad de todas las características de los cipher suites seguras.

El segundo modelo que usaremos es el mostrado en la Fig. 9, que corresponde con un modelo completo de *SSL/TLS* donde se presentan todas las características para la configuración de los cipher suites para cualquier producto o sistema basado en las versiones de *TLS1.2,* y *TLS1.3*. En principio todas las versiones anteriores se han considerado inseguras y por tanto se han descartado. Este modelo se puede enlazar con el de *Apache* de la Fig. 8 por la característica de *Ciphers*.

Queremos destacar que el modelo de *SSL/TLS* es simplificado, ya que no se han representado las restricciones cruzadas. Representar todas estas restricciones implicaría tener un modelo que sería visualmente muy complejo de interpretar. Sin embargo, se han dejado anotados en la caja de texto *Cross Relations* de la Fig. 9. Por ejemplo, *AES_1238_CBC* no se recomienda para la versión del protocolo *TSLv1.3*, por lo que se ha añadido una relación cruzada de exclusión. Por otro lado, el uso de *RSA* implica que el uso de tamaño de claves (*KeySize*) debe ser de *2048*.

IV-B. Análisis de los modelos de característica y diagnosis de configuraciones

A continuación, vamos a realizar un análisis de los modelos anteriores que describen nuestro contexto y se adecuan a una política de ciberseguridad. El análisis aplicado se centra en describir los modelos, validarlos, y ver cuántas son las configuraciones que pueden estar disponibles. Los resultados se pueden ver en Tab. II.

Tabla II: Datos extraído del modelo de ejemplo.

Modelo de característica	*Apache*	*SSL/TLS*
Número de características	27	48
Mandatory	10	8
Optional	3	0
OR	1	1
XOR	3	9
Cross-Relations	2	12
Válido	✓	✓
Número de configuraciones	96	1482

Podemos destacar que ambos modelos son válidos, por lo que de ambos modelos obtendríamos al menos una configuración válida. Con respecto a las configuraciones, podemos decir que el número de configuraciones posibles para *Apache* son 96, mientras que las de *SSL/TLS* son 1482. Estos datos nos dan una perspectiva de la complejidad que plantea la configuración de un sistema como *Apache* y *SSL/TLS*. Aún acotando las configuraciones a pocos parámetros, podemos ver como el dominio del problema sería seleccionar o analizar una configuración de entre miles, lo que clasifica a este problema como inabarcable si se realizara de manera manual por un humano. La complejidad aumentaría si nos planteáramos combinar ambos sistemas, ya que el número de configuraciones se multiplicaría significativamente, dando lugar a una explosión combinatoria en el número de posibles configuraciones, que haría el problema aún más intratable.

Uno de los puntos fuertes de nuestra propuesta, CyberSPL, es la posibilidad de hacer diagnosis de configuraciones. La diagnosis va más allá del mero hecho de determinar si una configuración es válida o no. La diagnosis pretende dar respuesta a el porqué una configuración es no válida de acuerdo a la política de ciberseguridad establecida. A continuación, vamos a presentar varios ejemplos de configuraciones de sistemas que necesitan ser verificadas, es decir, comprobar si están de acuerdo con la política representada en los modelos de características. En el caso de que las configuraciones no sean válidas, aplicaremos CyberSPL para establecer el diagnóstico, es decir, determinar cuáles son los posibles fallos en las mismas.

Como podemos observar en ambos casos en Tab. III y Tab. IV, se han probado cuatro configuraciones de las cuáles dos fueron no válidas, en cuyo caso se ha proporcionado

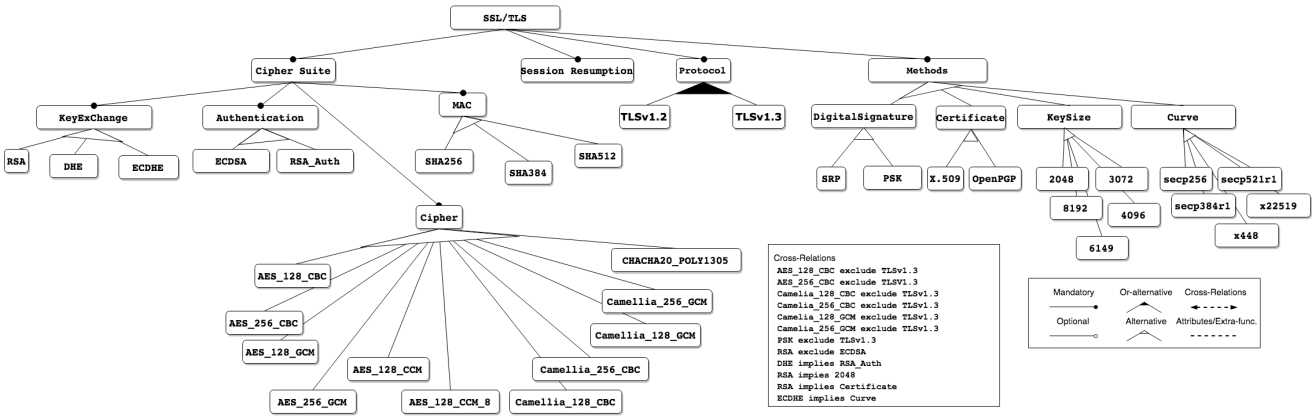

Figura 9: Modelo de características de SSL/TLS.

Tabla III: Diagnóstico de configuraciones en *Apache*

Configuración	Resultado	Diagnosis
Apache, Protocol, TLSv1.1, KeyStore, File, Pass, Type ClientAuth, false, Port	**Válida**	-
Apache, Protocol, TLSv1.2, KeyStore, File, Pass, Type Ciphers, Algorithm, ClientAuth, want, Port	**Válida**	-
Apache, Protocol, KeyStore, File, Pass, Type ClientAuth, false, Port, Algorithm, Ciphers	**No Válida**	**Seleccionar**: TLSv1, TLSv1.2, TLSv1.1, TLSv1.3
Apache, Protocol, TLSv1.2, KeyStore, File, Pass, Type, ClientAuth, Port, Algorithm, Ciphers	**No Válida**	**Seleccionar**: want, false

Tabla IV: Diagnóstico de configuraciones en *SSL/TLS*

Configuración	Resultado	Diagnosis
SSL/TLS, Protocol, TLSv1.2, KeyExchange, DHE, CipherSuite, Cipher, AES_128_GCM, MAC, SHA256, Authentication, ECDSA, Methods, KeySize, 3072, SessionResumption	**Válida**	-
SSL/TLS, Protocol, TLSv1.2, CipherSuite, Cipher, AES_128_CCM, Authentication, ECDSA, KeyExchange, DHE, MAC, SHA512, Methods, KeySize, 3072, SessionResumption	**Válida**	,
SSL/TLSS, Protocol, TLSv1.2, CipherSuite, Cipher, AES_128_CCM, Authentication, KeyExchange, ECDHE, MAC, SHA512, Methods, KeySize, 3072, SessionResumption	**No Válida**	**Seleccionar**: RSA_Auth, DHE **Deseleccionar**: ECDHE
SSL/TLS, Protocol, CipherSuite, MAC, SHA384, Authentication, ECDSA, Cipher, AES_256_GCM, KeyExchange, DHE, Methods, KeySize , 4096, SessionResumption	**No Válida**	**Seleccionar**: TLSv1.2, TLSv1.3

el diagnóstico. Para el caso de las configuraciones inválidas de *Apache* (Tab. III) podemos ver que la primera configuración fallida es porque no se ha especificado la versión del protocolo, aquí el diagnóstico sugiere la selección de una versión del protocolo concreta. Mientras que en el segundo caso se ha configurado todo pero no se ha indicado qué tipo de *ClientAuth*. En este caso no se puede seleccionar *true* porque implicaría la selección de la característica de *Trust*, por lo que el diagnóstico, nos sugiere que debemos seleccionar entre las opciones de *want* ó *false*. Para el caso de *SSL/TLS* (Tab. IV) tenemos también dos configuraciones inválidas. Para el diagnóstico de la primera configuración inválida, se observa que para la característica de *Authentication* no se ha seleccionado ningún mecanismo, que podría ser *ECDSA* o *RSA_Auth*. Por otro lado se ha seleccionado *ECDHE* como característica de *KeyExchange* que requeriría una curva elíptica (*Curve*) pero en nuestra configuración nos dice que estamos usando claves con *KeySize* de 3072. Por todo esto, la diagnosis más factible nos indica que debemos no seleccionar *ECDHE* y que podríamos seleccionar las características *RSA_Auth* y *DHE*, ya que seleccionando estas dos características se cumpliría con el requisito de una clave (*KeySize*) y un tamaño determinado (3072).

V. Estado del arte

Tras hacer una revisión bibliográfica del estado del arte al respecto, no se han encontrado referencia alguna sobre el uso de modelos de características aplicados en el ámbito del cumplimiento de políticas de ciberseguridad. Por tanto, a la vista de ello podemos decir que este trabajo es pionero en tratar esta problemática aplicando técnicas basadas en modelos de características. Dado que se integran dos áreas de investigación hasta ahora no integrados, los trabajos relacionados los hemos dividido en estas dos principales perspectivas que aborda el artículo:

Análisis y diagnosis sobre modelos de características

El análisis automatizado de modelos de características es algo conocido y aplicado desde hace décadas en el área de la líneas de productos software [8][22][24]. El análisis automatizado persigue extraer o inferir ciertas propiedades de los modelos. En ciertos trabajos el análisis se centraba en determinar, analizar, o diagnosticar errores en modelos de características ya sea en diseño [21] o en etapas de reconfiguración [23].

Existen otras aproximaciones donde el análisis de modelos de características se aplica a otros ámbitos. En [17], se propone una extensión de un método basado en objetivos (KAOS) para generar modelos de requisitos adaptables a partir

de modelos de variabilidad. En [19], se utilizan modelos de características para analizar los requisitos de variabilidad y, en consecuencia, transformar estos modelo de característica para generar un modelo arquitectónico. En [20], se utiliza el análisis de modelos de características para proporcionar sistemas auto-adaptables mediante la determinación dinámica de las mejores variantes adaptadas a los requisitos específicos de QoS.

Análisis o aplicación de mecanismos de verificación de configuraciones en el ámbito de la ciberseguridad

En este apartado, podemos considerar el trabajo [18], donde se propone un enfoque que facilita el desarrollo de líneas de productos de software seguro (SPLs) y sus productos derivados.

También debemos reseñar en el contexto de verificación de configuraciones de seguridad, que dado que los datos de configuración de los sistemas Internet-of-Things (IoT) son no estructurados, las técnicas tradicionales no pueden tratar de forma automática las configuraciones específicas del IoT tales como la seguridad. Es por ello que se ha propuesto la plataforma *IoTChecker* [9] para el análisis de seguridad en productos IoT. Finalmente, también debemos reseñar para el cumplimiento de políticas Bring your own device (BYOD) se ha propuesto en [2] una técnica para la verificación automática de seguridad en aplicaciones móviles.

VI. Conclusiones y trabajos futuros

En este artículo se ha presentando la plataforma CyberSPL y una herramienta para su implementación que nos permite la verificación de cumplimiento de políticas de ciberseguridad a través del análisis de configuraciones de productos, aplicaciones y servicios que participan en ella. Los resultados obtenidos con el uso de dicha propuesta nos llevan a pensar que puede representar un avance importante para facilitar el trabajo en la gestión automatizada del cumplimiento de políticas por parte de los gestores de la ciberseguridad o en el desarrollo operacional (DevOps), donde asegurar y comprobar que una configuración es acorde con la política supondría un notable incremento del alineamiento entre la capa de desarrollo y la operacional.

Como trabajos futuros se han identificado dos líneas principales: (1) la necesidad de realizar la actualización automática de los modelos de características de acuerdo con los avances tecnológicos o vulnerabilidades que se produzcan a lo largo del tiempo; y (2) el mantenimiento de un histórico de la evolución de los modelos de características para cada uno de los contextos de tal manera que nos permita alertar de forma temprana a los usuarios cuando se produzcan cambios en las políticas de ciberseguridad establecidas.

Agradecimientos

Este artículo ha sido parcialmente financiado por el Ministerio de Ciencia y Tecnología de España a través del proyecto ECLIPSE (RTI2018-094283-B-C33), de la Junta de Andalucía vía los proyectos PIRAMIDE y METAMORFOSIS, los fondos FEDER. Los autores quieren agradecer a la Cátedra de Telefónica "Inteligencia en la Red" de la Universidad de Sevilla por su apoyo en el desarrollo de este trabajo, y a José A. Galindo y David Benavides por el soporte de la herramienta FAMA.

Referencias

[1] D. Sisiaridis and O. Markowitch, "Automating Feature Extraction and Feature Selection in Big Data Security Analytics," in Artificial Intelligence and Soft Computing, Springer International Publishing, 2018, pp. 423–432.
[2] G. Costa, A. Merlo, L. Verderame, and A. Armando, "Automatic security verification of mobile app configurations," Future Generation Computer Systems, vol. 80, pp. 519–536, Mar. 2018.
[3] B. Behringer, M. Lehser, and S. Rothkugel, "Towards Feature-Oriented Fault Tree Analysis," in 2014 IEEE 38th International Computer Software and Applications Conference Workshops, 2014.
[4] A. J. Varela-Vaca and R. M. Gasca, "Towards the automatic and optimal selection of risk treatments for business processes using a constraint programming approach," Information and Software Technology, vol. 55, no. 11, pp. 1948–1973, Nov. 2013.
[5] M. Schumacher, Security Engineering with Patterns. Springer Berlin Heidelberg, 2003.
[6] Kang. Kyo, Cohen. Sholom, Hess. James, Novak. William, and Peterson. A., "Feature-Oriented Domain Analysis (FODA) Feasibility Study,"Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-90-TR-021 , 1990. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11231
[7] D. Batory, "Feature Models, Grammars, and Propositional Formulas," in Software Product Lines, Springer Berlin Heidelberg, 2005, pp. 7–20.
[8] D. Benavides, S. Segura, and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: A literature review," Information Systems, vol. 35, no. 6, pp. 615–636, Sep. 2010.
[9] M. Mohsin, Z. Anwar, F. Zaman, and E. Al-Shaer, "IoTChecker: A data-driven framework for security analytics of Internet of Things configurations," Computers & Security, vol. 70, pp. 199–223, Sep. 2017.
[10] R. Lotufo, S. She, T. Berger, K. Czarnecki, and A. Wąsowski, .Evolution of the Linux Kernel Variability Model,ïn Software Product Lines: Going Beyond, Springer Berlin Heidelberg, 2010, pp. 136-150.
[11] M. Schumacher, Security Engineering with Patterns. Springer Berlin Heidelberg, 2003.
[12] A. J. Varela-Vaca and R. M. Gasca, "Formalization of security patterns as a means to infer security controls in business processes," Logic Journal of IGPL, vol. 23, no. 1, pp. 57–72, Dec. 2014.
[13] D. Benavides, P. Trinidad, A. Ruiz Cortés, and S Segura. "FaMa". Springer Berlin Heidelberg, 2013, Chapter FaMa, 163–171. DOI:http://dx.doi.org/10.1007/978-3-642-36583-6-11.
[14] K. Hickman, The SSL Protocol. "Netscape Communications Corp.", 1995.
[15] T. Dierks, E. Rescorla, "The TLS Protocol Version 1.2". RFC 5246, 2008.
[16] E. Rescorla, "The TLS Protocol Version 1.3". RFC 8446, 2018.
[17] F. Semmak, C. Gnaho, and R. Laleau, "Extended KAOS Method to Model Variability in Requirements," in Communications in Computer and Information Science, Springer Berlin Heidelberg, 2010, pp. 193–205.
[18] D. Mellado, E. Fernández-Medina, and M. Piattini, "Security requirements engineering framework for software product lines," Information and Software Technology, vol. 52, no. 10, pp. 1094–1117, Oct. 2010.
[19] J. Pérez, M. A. Laguna, Y. C. González-Carvajal, and B. González-Baixauli, "Requirements Variability Support Through MDA™ and Graph Transformation," Electronic Notes in Theoretical Computer Science, vol. 152, pp. 161–173, Mar. 2006.
[20] P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, and D. Hughes, "Using Constraint Programming to Manage Configurations in Self-Adaptive Systems," Computer, vol. 45, no. 10, pp. 56–63, Oct. 2012.
[21] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro, "Automated error analysis for the agilization of feature modeling," Journal of Systems and Software, vol. 81, no. 6, pp. 883–896, Jun. 2008.
[22] D. Benavides and J. A. Galindo, "Automated analysis of feature models," in Proceeedings of the 22nd International Conference on Systems and Software Product Line - SPLC '18, 2018.
[23] A. Felfernig et al., "Anytime diagnosis for reconfiguration," Journal of Intelligent Information Systems, vol. 51, no. 1, pp. 161–182, Jan. 2018.
[24] J. A. Galindo, D. Benavides, P. Trinidad, A.-M. Gutiérrez-Fernández, and A. Ruiz-Cortés, "Automated analysis of feature models: Quo vadis?," Computing, Aug. 2018.
[25] C. Prud'homme, J.-G. Fages, X. Lorca, Choco Documentation, http://www.choco-solver.org, 2017.