# Metodología para automatizar agentes atacantes en plataformas de entrenamiento Cyber Range

Pablo Martínez Sánchez[1], Pantaleone Nespoli[1,2], Joaquín García Alfaro[2], Félix Gómez Mármol[1]
[1]Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia, 30100, Murcia, España
{pablo.martinezs2,pantaleone.nespoli,felixgm}@um.es
[2]SAMOVAR, Télécom SudParis, Instituto Politécnico de París, 19 place Marguerite Perey, 91120 Palaiseau, Francia
joaquin.garcia_alfaro@telecom-sudparis.eu

***Resumen*— El mundo se enfrenta día tras día a ciberataques, donde más de la mitad de los profesionales en ciberseguridad no poseen los conocimientos necesarios para desplegar las contramedidas pertinentes. En este sentido, no cabe duda que la formación y capacitación en ciberseguridad es esencial para defender los activos tecnológicos. Es por ello que, en este contexto, es fácil entender que los Cyber Ranges juegan un papel crucial, puesto que, estas herramientas proveen al usuario de una experiencia hiperrealista para un entrenamiento de calidad. Es gracias a los simuladores de ataques, comúnmente serán generadores de APTs, que se pueden realizar ciberejercicios defensivos realistas. Para implementar este componente deberemos contar con una matriz de comportamiento, las cuales marcan las diferentes etapas que usa un experto de ciberseguridad durante un ataque, p.ej., reconocimiento, penetración, infiltración de datos, etc. Dado que llevar las metodologías actuales a un entorno de producción hiperrealista es un reto desmesurado, a partir de estas se diseñará una nueva matriz. Esta nueva metodología compactará las fases dependientes y simplificará las etapas similares, en vistas de realizar automáticamente el flujo mediante una implementación. Será entonces cuando se diseñe una prueba de concepto que, usando la nueva metodología y diferentes patrones de diseño, estará desligada de herramientas y funcionará en entornos donde la información se conocerá paulatinamente.**

***Index Terms*—Metodología de ataque, Advanced Persistent Threat, Cyber Range, Ciberseguridad, Agentes ofensivos**

**Tipo de contribución:** *Formación e innovación educativa*

## I. Introducción

La inversión en ciberseguridad aumenta cada año pese a no ver un resultado robusto frente a los ciberataques, ya que, como afirman estudios publicados en el CSIS (Center for Strategic and International Studies) esta inversión en seguridad e investigación sigue sin ser suficiente [1]. En el anterior artículo, se puede observar como Estados Unidos o China han evolucionado en este ámbito gracias a apostar en innovación y tener independencia respecto a otras potencias.

Para estudiar el impacto de los ataques sufridos por empresas, se han empleado técnicas que cuantifican las distintas posturas de seguridad de las redes mediante la simulación de ciberataques. Algunas de estas pruebas son: i) Pruebas de penetración tradicionales, las cuales se centran en comprobar la robustez, seguridad e integridad de la red y ii) Ejercicios Red Team, donde un grupo de expertos simulan un ataque realista, donde intentan comprometer la red imitando los ataques llevados a cabo por los ciberdelincuentes. Como se puede observar, ambas son ejecutadas por equipos de personas, lo que supone que estas pruebas: i) no sean de total fiabilidad, dada la tasa de error de un humano producida por la fatiga y ii) sean costosas, por el gran desembolso que supone contratar un profesional experimentado. Por este motivo nace el gran reto de automatizar las acciones desempeñadas por los expertos y cibercriminales, con la finalidad de conseguir: robustez, para poder asegurar el determinismo; la escalabilidad, para poder replicar las pruebas las veces oportunas; y bajo coste de los resultados, frente a una simulación realista de un ataque.

Además, con la devastadora llegada de inteligencia artificial, se puede asegurar la detección de patrones de comportamiento que un humano podría obviar. El aprendizaje automático ha permitido crear asistentes inteligentes, generadores de imágenes o código y un sin fin de nuevas posibilidades, por lo que no sería descabellado pensar en una inteligencia artificial capaz de orquestar un ataque a una organización. Una posible implementación podría ser aprendizaje por refuerzo, ya que, por definición, se busca planear estrategias efectivas en base a la experimentación interaccionando con el entorno.

Una posible herramienta para formar a profesionales en ciberseguridad son los Cyber Ranges, donde se pueden implementar: inteligencia artificial, gamificación, enfrentamientos por equipos, etc. Estos son entornos de virtualización pensados para poder simular y/o emular entornos hiperrealistas donde llevar a cabo las actividades, como son: acciones defensivas, evasivas, penetración, exfiltración, etc. Gracias a este tipo de herramientas, a día de hoy, los usuarios de un Cyber Range pueden adquirir habilidades altamente demandadas en ciberseguridad. Según el informe del Foro Económico Mundial [2], existe un 62 % de profesionales en ciberseguridad que aún no poseen las habilidades necesarias para ser capaces de responder a un ciberataque, entre las cuales tenemos la defensa frente a ataques novedosos y disruptivos, como son los ataques de día cero.

Simular un ataque realista del que los usuarios de un Cyber Range deban defenderse supone afrontar un reto que tiene una gran complejidad. Se deberá definir y plasmar la metodología usada para atacar de una forma tan completa e inclusiva, que sea capaz de adaptarse a un escenario nunca antes visto por la herramienta. Pese a existir metodologías de ataque que definen esta casuística, estas no tienen porqué cumplir estos requisitos, dado el paso del tiempo y la complejidad en aumento de los nuevos vectores de ataque. Por ello, en este documento, se pretende responder a las siguientes preguntas de investigación:

1. ¿Pueden los simuladores de APTs seguir matrices de ataque realistas?
2. ¿Son los simuladores de APTs suficientemente autónomos para realizar un ataque?
3. ¿Pueden servir los simuladores de APTs a los usuarios de un Cyber Range para mejorar sus habilidades en el ámbito de la ciberseguridad?

Para solventar esto se ha planteado una nueva matriz que compacta, automatiza e intercomunica las diferentes fases presentadas en las metodologías existentes (en respuesta a la pregunta 1). Esto ha sido probado con una implementación software basado en un entorno Cyber Range donde, gracias a diferentes patrones de diseño e interfaces de programación de aplicaciones. De esta manera se consigue que el simulador de ataque utilice cualquier herramienta y tome decisiones con un entorno que va conociendo paulatinamente (contestando a la pregunta 2). La mejora en la autenticidad y adaptabilidad del atacante tendrá un impacto directo en la mejora de las maniobras defensivas de los usuarios, lo que a su vez mejorará las habilidades necesarias para responder eficazmente a un ciberataque (como respuesta a la pregunta 3).

La estructura del artículo es la siguiente. La Sección II analizará la literatura existente sobre metodologías de ataque y herramientas ofensivas. A continuación, la Sección III propondrá la metodología desarrollada que resuelva los problemas planteados. La Sección IV expone una arquitectura desligada de herramientas, haciendo uso de patrones de diseño. Seguidamente, la Sección V contiene la implementación de la herramienta basada en la arquitectura planteada. En la Sección VI muestra, como prueba de concepto, el uso de la herramienta sobre un entorno de simulación. Finalmente, la Sección VII resume las conclusiones sobre la investigación presentada y presenta posible vías futuras.

## II. Estado del arte

El problema de la búsqueda de un simulador de ataques realista sigue siendo un problema no resuelto. Actualmente las grandes empresas nacionales tienen grupos de expertos en ciberseguridad realizando ejercicios de Red Team, donde cada integrante se suele especializar en determinadas competencias del ataque (servicios web, servicios de correo, firewalls, etc.). Dada la complejidad de los ataques actuales estos expertos siguen una metodología propia, basada en su experiencia. A su vez, estos utilizan herramientas que automatizan acciones mecánicas y repetitivas, ahorrando tiempo e incrementando la fiabilidad de los ataques. Algunas de estas son: Nmap, para la enumeración de dispositivos; Hashcat, para comprobar contraseñas poco seguras; Burpsuite, para automatizar acciones sobre interfaces web; entre otras.

### *II-A. Matrices*

A día de hoy los sistemas cuentan con más medidas de protección, lo que supone un incremento en la complejidad de los ataques y herramientas que automatizan la explotación de vulnerabilidades. Es por ello que definir una metodología que estandarice todo ello requiere mucho conocimiento y es de gran complejidad. A lo largo del tiempo han aparecido varias matrices, aunque en la actualidad se encuentran clasificaciones reconocidas mundialmente, como son: la matriz MITRE ATT&CK [3] o la Cyber Kill Chain (CKC) [4].

MITRE ATT&CK es una base de conocimientos de acceso global sobre tácticas (es decir, conjuntos temáticos de técnicas) y técnicas (es decir, conjuntos de ataques basados en las observaciones del mundo real). La base de conocimientos MITRE ATT&CK se utiliza como fundamento para el desarrollo de modelos y metodologías de amenazas específicas en el sector privado, en la administración pública y en la comunidad de productos y servicios de ciberseguridad. La matriz se crea desde un punto de vista teórico y reúne los ataques más utilizados en entornos reales. El repositorio por excelencia, que contiene la implementación de los ataques organizados bajo la jerarquía de MITRE ATT&CK, es Atomic Red Team[1]. En particular, la matriz está compuesta por las fases descritas en la Tabla I.

Tabla I
FASES DE LA MATRIZ MITRE ATT&CK

| Etapa | Identificador | Descripción |
|---|---|---|
| Reconnaissance | TA0043 | Recopilación de información libre |
| Resource Development | TA0042 | Creación de recursos de soporte |
| Initial Access | TA0001 | Recopilación de información de red |
| Execution | TA0002 | Explotación de vulnerabilidades |
| Persistence | TA0003 | Establecimiento de persistencia |
| Privilege Escalation | TA0004 | Obtención de derechos de administrador |
| Defense Evasion | TA0005 | Maniobras que eviten la detección |
| Credential Access | TA0006 | Hurto de usuarios y contraseñas |
| Discovery | TA0007 | Descubrimiento del entorno interno |
| Lateral Movement | TA0008 | Desplazamiento sobre el entorno interno |
| Collection | TA0009 | Recopilación de información de interés |
| Command and Control | TA0011 | Comunicación con los equipos vulnerados |
| Exfiltration | TA0010 | Robo de información |
| Impact | TA0040 | Manipulación, interrupción y destrucción |

Por otro lado, el modelo de la Cyber Kill Chain de la ciberseguridad explica el procedimiento típico que siguen los ciberdelincuentes para completar un ataque cibernético con éxito. Se trata de un marco desarrollado por Lockheed Martin, derivado de los modelos de ataque militares y trasladado al mundo digital para ayudar a los equipos a comprender, detectar y prevenir las ciberamenazas persistentes. Se diferencia a esta metodología entre las citadas como la más teórica, ya que, carece de referencia a ataques reales. Está compuesta por las fases reportadas en la Tabla II.

Tabla II
FASES DE LA MATRIZ CIBER KILL CHAIN

| Etapa | Descripción |
|---|---|
| Reconocimiento | Recopilación de información de fuentes libres |
| Preparación | Selección y explotación de los vectores de ataque |
| Distribución | Distribución de la carga maliciosa por los sistemas |
| Explotación | Explotación del malware distribuido |
| Instalación | Establecimiento de persistencia |
| Comando y control | Comunicación con los equipos vulnerados |
| Acciones sobre los objetivos | Monitorización y postexplotación |

También se pueden encontrar otras metodologías con enfoques prácticos, como es la que nos proporciona el experto Carlos Polop, es decir, HackTricks. Esta metodología se crea partiendo de la experiencia, ya que, ligada a esta podemos encontrar una detallada descripción de la implementación de cada uno de los vectores de ataque. En ella se explica al usuario el razonamiento de cada decisión tomada en la metodología, así como se describen los conceptos, pasos a seguir o configuraciones necesarias para las herramientas implicadas en ciberseguridad [5]. Está compuesta por las fases descritas en la Tabla III.

Lamentablemente, las metodologías de MITRE y CKC, como se ha comentado anteriormente, se basan puramente en la teoría y fragmentan en exceso las etapas de un ciberataque actual. Los ataques y herramientas actuales unifican y combinan diferentes etapas de estas metodologías con el fin de poder automatizar la ejecución, sin perder adaptabilidad

[1]https://atomicredteam.io/

Tabla III
FASES DE LA MATRIZ HACKTRICKS

| Etapa | Descripción |
|---|---|
| Physical Attacks | Ataques de carácter físico al equipo |
| Discovering | Descubrimiento de vías de acceso y equipos |
| Discovering internal | Captación de información sensible una vez dentro de la red |
| Port scan | Búsqueda de servicios vulnerables en los equipos |
| Searching service version exploits | Búsqueda de vulnerabilidades conocidas de los servicios |
| Pentesting Services - Automatic Tools | Explotación de vulnerabilidades con herramientas automáticas |
| Pentesting Services - Brute-Forcing services | Explotación mediante ataques de fuerza bruta |
| Phising | Si las etapas anteriores no han funcionado, realizar phishing |
| Getting Shell | Obtención de medios para la ejecución de comandos remotos |
| Inside | Ejecución de acciones remotas en las máquinas víctima |
| Exfiltration | Extracción e introducción de archivos en la víctima |
| Privilege Escalation - Local Privesc | Obtención de privilegios en la máquina víctima |
| Privilege Escalation - Domain Privesc | Obtención de privilegios en la organización |
| POST - Looting | Obtención de datos críticos y/o confidenciales |
| POST - Persistence | Establecimiento de persistencia |
| Pivoting | Infección de otros equipos conectados con la víctima |

o robustez. Si bien MITRE ATT&CK, gracias a Atomic Red Team, cuenta con la implementación de los ataques, estos son muy dependientes del entorno y sus condiciones. El motivo es que han sido recopilados de forma aislada a, por ejemplo, la topología de red. Por otra parte, al ser la matriz HackTricks tan práctica, es la más adecuada para poder implementar una automatización, dada la flexibilidad que aporta al panorama actual. El punto desfavorable, por el cual no haremos uso de ella, es que supone un gran reto concatenar las diferentes fases, puesto que la matriz se plantea para que sea seguida por un usuario. Es por ello que pese a encontrar con las soluciones descritas, estas por si solas, no se acercan a lo requerido.

### *II-B. Herramientas ofensivas y Cyber Ranges*

Aunque no existe una metodología específica para el desarrollo de herramientas que automatizan los ataques, la creciente demanda en ciberseguridad ha impulsado el enfoque en ellas. Es decir, se han desarrollado aplicaciones que mecanizan las acciones realizadas por los grupos mencionados anteriormente.

A nivel nacional el panorama es aún joven, se pueden apreciar algunas soluciones basadas en el marco de MITRE ATT&CK, como lo es la herramienta desarrollada por la Universidad Politécnica de Madrid (UPM) [6]. Esta se basa en la plataforma Caldera[2], desarrollada por MITRE. Su principal cometido es automatizar la ejecución de las diferentes fases de ataque predefinidos y de forma serial en la herramienta. Esto supone una serie de graves problemas: i) de fallar en una etapa no crítica podría detener el ataque, ii) la solución está totalmente ligada a la herramienta y hereda los defectos que esta posea, iii) la interfaz de programación de la aplicación no es robusta, etc.

Mirando, esta vez, al marco internacional encontramos soluciones un tanto superiores, entre ellas las principales referentes son: Caldera, realizada por MITRE; HARMer, realizada por el Ministerio de Defensa surcoreano [7]; CyberBattleSim, realizada por Microsoft [8]; entre otras.

El proyecto subvencionado por Corea del Sur tiene una gran trayectoria, involucrando a 6 profesionales. En este desarrollan una solución completa y con buenos resultados, donde la herramienta debe conocer toda la topología para poder operar con todo su potencial. Esto provoca que no se pueda ejecutar en entornos hiperrealistas donde el atacante vaya aprendiendo gradualmente el entorno donde se encuentra. Además, para poder explotar las vulnerabilidades utiliza software como Metasploit[3], lo que supone una degradación equitativa a las carencias que puedan tener la herramienta en cuestión.

Otra solución a considerar es CyberBattleSim. Se aprecia como la inteligencia artificial está empezando a adentrarse también en ámbito de la ciberseguridad, concretamente a la hora de tomar decisiones cuando hay varios vectores de ataques disponibles. Esta herramienta es muy potente, pese a estar hecha en un entorno idealizado donde poder entrenar inteligencia artificial, es decir, muy lejos poder implementar una versión sobre una topología realista.

Todas las herramientas ofensivas comentadas están orientadas al ámbito educativo, por lo que se puede asumir que el entorno donde se utilizarán será también educativo. Como se ha comentado anteriormente, este entorno podría ser y será en la mayoría de los casos, un Cyber Range. Algunos ejemplos de este tipo de entornos virtualizados hiperrealistas, a nivel nacional, son: el Cyber Range desarrollado por Indra, bajo licencia privada [9]; o COBRA [10], siendo este último desarrollado por la Universidad de Murcia en colaboración con el Ministerio de Defensa de España.

Pese a ser COBRA una herramienta muy completa, los requisitos pedidos para su desarrollo fueron validaciones de componentes y/o disposición de los mismos en entorno de laboratorio. Por lo tanto, aún dista de poder utilizarla en un entorno de producción. En ella se contaba con gamificación, generación pseudoaleatoria de escenarios, ciberejercicios red (simulación de ataque) y blue (simulación defensiva), etc. Todas ellas pruebas de concepto.

En el ámbito internacional se encuentra KYPO [11], una herramienta desarrollada por la universidad de Masaryk, la cual cuenta con casi una década de desarrollo. Esta sí se encuentra en el entorno de producción y es un ejemplo reconocido a nivel mundial. No obstante, no cuenta con Learning Tools Interoperability (Interoperabilidad entre Herramientas de Aprendizaje o LTI) o automatización de algunos servicios, como la generación de escenarios. Dada su trayectoria sus opciones son limitadas pero muy robustas.

Al adentrase mejor en los Cyber Ranges anteriores se puede ver que la mayoría hacen uso de un generador y/o controlador de APTs para realizar ciberejercicios blue. Estos proporcionan a los instructores "atacantes virtualizados", pudiendo así replicarlos, configurarlos y desplegarlos de manera sencilla y eficaz. De esta manera los usuarios pueden cursar estas maniobras en un entorno hiperrealista donde "al otro lado de la pantalla" no hay ningún humano.

Como se ha comentado anteriormente, las matrices existentes tienen un enfoque divulgativo y distan bastante de poder llevar a cabo implementaciones hiperrealistas. Esto se debe a que al fragmentar las etapas con tanto detalle incrementa la complejidad que tiene la automatización de las mismas. Este es el principal motivo de no poder encontrar ninguna herramienta que simule fielmente a un atacante, puesto que su adaptabilidad se ve mermada. No obstante, se puede ver como Cyber Ranges de caché internacional implementan sus propios generadores de APTs. Pese a no contar con una herramienta flexible y completa, se puede concluir que su uso es efectivo para poder entrenar usuarios en la actualidad, contestando así a la pregunta 3 planteada en la introducción.

[2]https://github.com/mitre/caldera

[3]https://www.metasploit.com/

## III. METODOLOGÍA

El principal reto de esta investigación es desarrollar una metodología que sea lo suficientemente genérica y flexible como para poder automatizar todo tipo de ataques realistas dentro de una plataforma de entrenamiento (p.ej., plataformas Cyber Ranges) donde poder desarrollar capacidades de ciberseguridad. Debido a su conocimiento maduro y su gran importancia en el ámbito, pese a no ser completas, se tendrán en cuenta las metodologías comentadas anteriormente: MITRE ATT&CK I, CKC II y HackTricks III. Las fases resultantes del exahustivo estudio realizado son: *Discovery*, *Penetration*, *Inside*, *Pivoting* y *Logic*.

La fase *Discovery* comprende todo lo referente a descubrimiento de información de la organización: finger/footprint (recopilación de datos en el ámbito público), descubrimiento de equipos, servicios y vulnerabilidades en la organización, relación entre servicios y Common Vulnerabilities and Exposures (CVE), etc.

Seguidamente, *Penetration* engloba todo tipo de explotación de vulnerabilidades: explotación de vulnerabilidades y exposiciones comunes (CVE), ataques de fuerza bruta y denegación de servicio, desarrollo de software, ejecución de código remoto, entre otras.

Una vez que se ha conseguido tomar el control del equipo, se pasa a la fase *Inside* donde toman lugar todas las acciones que ocurrirán dentro de la máquina víctima: exfiltración de datos, persistencia, escalada de privilegios, etc.

Para finalizar, es necesario hacer una distinción a la hora de establecer los canales de comunicación para realizar un movimiento lateral entre diferentes equipos. A esta fase se la denomina *Pivoting*.

Todo ello es englobado por una lógica que relaciona y gestiona las dependencias entre las diferentes fases anteriormente descritas. Es gracias a *Logic* que un profesional, en el caso de un Cyber Range un instructor, puede configurar y adaptar las diferentes etapas para la simulación de un ataque.

La creación de estas se basa en que las diferentes etapas son llevadas a cabo sobre una misma fase del ataque y la fuerte dependencia que ciertas partes tienen entre sí. Son varias las acciones realizadas tras conseguir la toma de un objetivo, haciendo que estas se puedan realizar sobre una misma etapa. Así mismo, hay ciertas fases que están fuertemente ligadas a una etapa previa, haciendo posible su unificación. Como resultado obtenemos que al compactar las fases, si bien es cierto que no se pierde información a cambio de aumentar levemente la complejidad, se logra dotar a las implementaciones basadas en esta metodología de automaticidad.

Una vez definidas las diferentes etapas, se planteará sustentar la fiabilidad y completitud de esta nueva metodología. Para ello será necesario la comparación con las citadas anteriormente. Es decir, la Tabla IV compara nuestra propuesta con MITRE ATT&CK, mientras que la Tabla V la relaciona con CKC, y finalmente la Tabla VI la contrapone a HackTricks.

Como se puede observar ver en dichas tablas, para cada fase de la matriz con la que se está haciendo la comparación siempre hay una de las columnas marcadas, es decir, todas las fases están comprendidas en la metodología diseñada, indicando así que es completa. Esto conlleva que apartados de metodologías anteriores queden compactados y potencialmente automatizados, convergiendo en una metodología de ataque realista. De esta manera resolvemos la pregunta 1.

Tabla IV
COMPARACIÓN MITRE ATT&CK

| | Discovery | Exploitation | Inside | Pivoting | Logic |
|---|---|---|---|---|---|
| TA0043 | X | | | | |
| TA0042 | | X | | | |
| TA0001 | X | | | | |
| TA0002 | | X | | | |
| TA0003 | | | X | | |
| TA0004 | | X | | | |
| TA0005 | | | | | X |
| TA0006 | | | X | | |
| TA0007 | X | | | X | |
| TA0008 | | | | X | X |
| TA0009 | | | X | | |
| TA0011 | | | | | X |
| TA0010 | | | X | | |
| TA0040 | | X | X | | |

Tabla V
COMPARACIÓN CKC

| | Discovery | Exploitation | Inside | Pivoting | Logic |
|---|---|---|---|---|---|
| Reconnaissance | X | | | | |
| Weaponization | | | | | X |
| Delivery | | X | X | X | X |
| Exploitation | | X | | | X |
| Installation | | | X | | |
| Command and Control | | | | | X |
| Actions on Objective | | | X | | |

Con el fin de no heredar ninguna dependencia de las herramientas que se usen para automatizar el ataque, se propone que se usen interfaces que hagan posible la homogeneización entre herramientas. Es común ver como el software simulador de ataques se liga a una herramienta, lo que supone la herencia de las carencias de dicha herramienta. En nuestra implementación se hace uso de varias herramientas simultáneamente, dado que la arquitectura así lo permite.

Si bien esta metodología se ha creado con la intención de poder automatizar los ataques, también se ha tenido en cuenta que una inteligencia artificial, como podría ser la anteriormente mencionada CyberBattleSim, sustituya a la etapa *Logic*. Esto dotaría a la implementación de la matriz de un gran avance, aportando realismo y complejidad.

## IV. ARQUITECTURA

En esta sección se plantea una arquitectura donde poder llevar a cabo la metodología. Para la propuesta será necesaria una capa de abstracción adicional que haga posible

Tabla VI
COMPARACIÓN HACKTRICKS

| | Discovery | Exploitation | Inside | Pivoting | Logic |
|---|---|---|---|---|---|
| Discovering | X | | | | |
| Internal Test | X | | | | |
| Port scan | X | | | | |
| Searching service | X | | | | |
| Pentesting service | | X | | | |
| Phishing | | X | | | |
| Getting shell | | X | | | |
| Inside | | | X | | |
| Exfiltration | | | X | | |
| Privilege escalation | | | X | | |
| Post | | | X | | |
| Pivoting | | | | X | |

la implementación de la matriz. En este caso se plantea el uso de generadores de APTs dada su gran potencia e importancia en los Cyber Ranges. Es por ello que se ha recurrido a la creación de una interfaz de programación de aplicaciones (Application Programming Interface, API), con la cual, se consigue homogeneidad frente al uso de diferentes generadores de APTs.

Adicionalmente, se ha creado una API para cada una de las etapas, dado que, para cada una podemos encontrar diferentes herramientas de automatización que nos permiten explotar vulnerabilidades o fallos de seguridad. Cabe añadir que dentro de cada API se cuenta con la opción de poder lanzar comandos de manera manual. A continuación, se puede apreciar la arquitectura en la Fig. 1.

Diferenciamos entre distintos componentes en función de los colores dentro de la Fig. anterior: i) purpura, las cuales albergan la lógica de cada fase, intercomunicando las diferentes herramientas y funcionalidades disponibles; ii) celeste, las interfaces correspondientes a cada una de las etapas de la metodología; iii) naranja, la implementación de la interfaz para cada herramienta disponible; iv) gris, cada una de las herramientas disponibles; v) rojo, donde está la base de conocimiento de los ataques; vi) y verde, para definir la API con la que poder comunicarse con el controlador de APTs, concretamente con la clase *Logic*. Además de la clasificación visual, cabe destacar el módulo *Agents*, que contiene los APTs, en este caso de la herramienta Caldera.

La base de la solución recaerá sobre la interfaz *APT_controller*, que comunicará los APTs de una herramienta ya existente con el resto de la arquitectura. Además, *Logic* será la encargada de intercomunicar al resto de fases, siendo este el motivo por el cual el resto de las etapas están contenidas en ella.

Como se puede observar, contamos con herramientas muy conocidas y utilizadas por otros simuladores de ataque, como pueden ser: Metasploit, Nmap, Hashcat, Atomic Red Team, entre otras. Esta arquitectura nos permite seccionar la funcionalidad de las herramientas, utilizarlas y conseguir evadir la problemática causada por el uso de una única herramienta. Para cada herramienta se deberá implementar la funcionalidad de su interfaz, la cual que deberá ser redefinida.

Para poder intercomunicar las diferentes etapas, estas devolverán mensajes en formato JSON. Todos ellos serán recibidos, procesados y gestionados bajo la lógica de la etapa *Logic*. Esta seguirá un flujo donde, en cada iteración se tendrá en cuenta la información descubierta hasta el momento y se decidirá cuál es la mejor acción a tomar. Podemos ver el flujo en la Fig. 2.

Vemos como, tras obtener una configuración inicial, intentamos descubrir algún equipo y servicio (*Discovery*). Si esto es posible pasaremos a atacar al servicio, bien sea por una vulnerabilidad detectada o fuerza bruta (*Exploitation*). Al conseguir esto podemos dividir el flujo en tres caminos: se consigue el objetivo o no se consigue nada, donde en ambos casos termina la ejecución; y se consigue la ejecución remota de código. Será en ese momento donde se despliegue un APT en la máquina vulnerada y se proceda a la recolección de información (*Inside*). Tras escalar privilegios se intentarán establecer medios para poder lanzar las herramientas desde la máquina vulnerada (*Pivoting*) para así volver a empezar de nuevo el flujo.

El punto crítico en la arquitectura propuesta viene cuando se debe decidir que ataque, de los disponibles dados por las herramientas implementadas, se ejecutará. Para solventar esto se propone una base de conocimiento donde se almacene y actualice el "potencial" del ataque para cada herramienta implementada. Se ha optado por esta opción por los siguientes motivos: i) las vulnerabilidades explotadas no siempre son explotables, ii) existen implementaciones diferentes en función de las herramientas usadas y iii) la criticidad no es suficiente, deberemos priorizar aquellos ataques que nos permitan la toma del equipo. En la Fig. 1 denotamos la base de conocimiento por su color rojo, llamada *Attack score*. Con el uso de la herramienta se irá evaluando y actualizando la precisión con la que ese ataque puede penetrar en la vulnerabilidad, de las siguientes maneras:

$$puntuacion = CVSS * nivel\_exito$$

$$nueva\_puntuacion = (media * n + puntuacion)/(n + 1)$$

Donde las variables toman valores en función de:

- *puntuacion*: la puntuación actual calculada sobre la criticidad
- *CVSS*: nivel de criticidad de la vulnerabilidad detectada
- *nivel_exito*: una variable que puede tomar los valores:
  - 0, al no poder explotar la vulnerabilidad
  - 0.5, al poder explotarla y conseguir Remote Code Execution (ejecución remota de código o RCE) con un usuario que no tenga permisos root
  - 1, al poder explotarla y conseguir RCE con un usuario que tenga permisos de root
- *media*: puntuación media guardada en la base de conocimiento
- *n*: la cantidad de veces que se ha calculado la media
- *nueva_puntuacion*: la nueva puntuación media

Tras intentar vulnerar un sistema se almacenará el resultado y se actualizará en la base de conocimiento los siguientes datos: *n* y *nueva_puntuacion*. En el caso de que ese ataque no haya obtenido ningún resultado para cierta máquina se registrará en *Logic* y se procederá a probar el siguiente más prometedor. Todos los ataques parten de base con la valoración establecida, a nivel mundial, en Common Vulnerability Score System (sistema de puntuación de vulnerabilidades comunes o CVSS).

Además, dada la finalidad que un atacante persigue diferenciaremos entre dos grandes tipos de ataques: tomar el control de la máquina o denegar un servicio (Denial of Service o DoS en inglés) de la organización víctima. Elegir entre uno u otro vendrá determinado por los datos de inicialización.

En algunas ocasiones se podrá denegar el servicio sin necesidad de tomar el control de la máquina que lo contiene. Sin embargo, si nuestro objetivo es, por ejemplo, conseguir acceder a una base de datos de copias de seguridad, deberemos tomar el control de la máquina. Es por ello que, en una primera instancia, si la meta predefinida en la inicialización es DoS, se probará a cortar el acceso al servicio. Será entonces, si no se alcanza el objetivo, cuando se proceda a la toma del equipo.

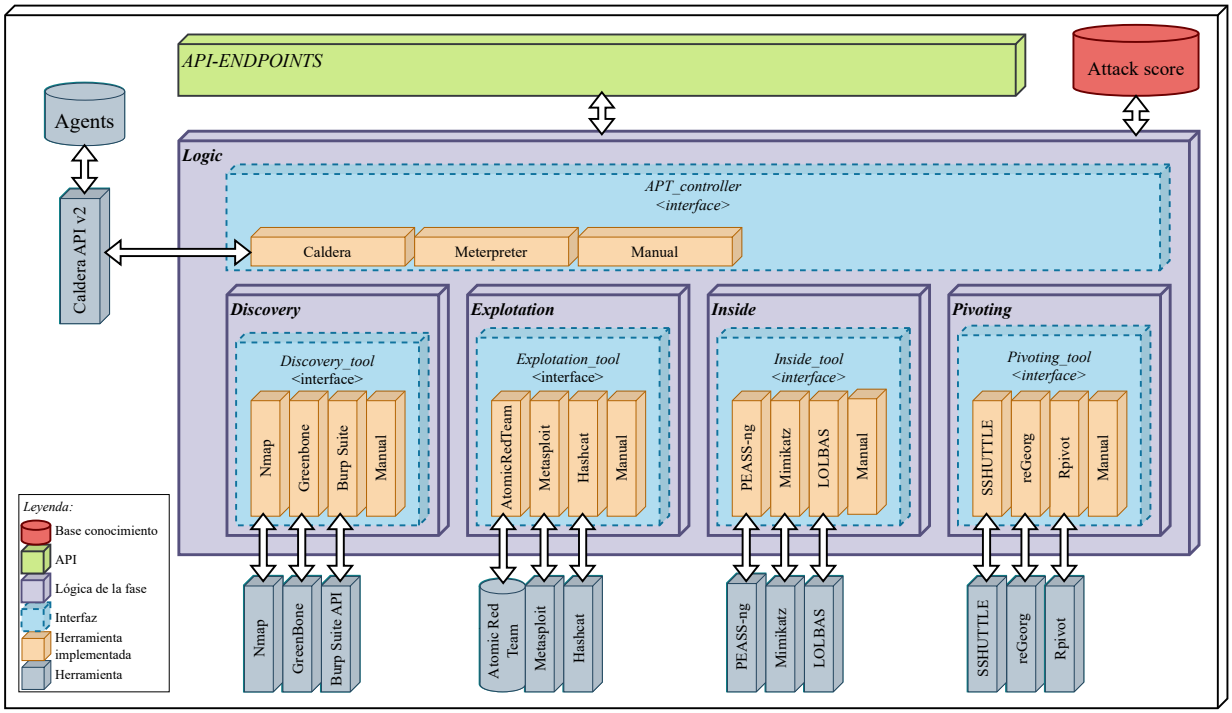


Figura 1. Arquitectura propuesta: diagrama de bloques

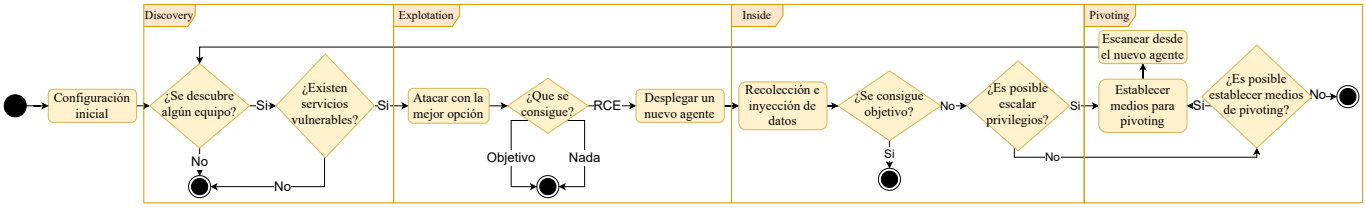


Figura 2. Arquitectura propuesta: Flujograma de ataque en *Logic*

Esto nos hace plantearnos las diferentes condiciones de victoria. Comúnmente en los Cyber Range se ha planteado el tipo de ejercicio Capture The Flag (captura la bandera o CTF). En ellos se pretendía obtener una cadena de texto a modo de bandera, sirviendo esta como muestra de la victoria del usuario. A día de hoy se plantean retos innovadores que no pueden ser cuantificados de esta forma, como el anteriormente mencionado DoS. Es por ello que se deberá comprobar la condición de victoria en diferentes etapas del ataque, como podemos observar en al Fig. 1 con decisiones como: *"¿Qué se consigue?"* o *"¿Se consigue el objetivo?"*.

Adicionalmente, *Logic* tendrá en cuenta la profundidad del equipo descubierto, ya que, se ha optado por una aproximación a la búsqueda primero en profundidad. Creemos que este enfoque es mejor que la búsqueda primero en anchura, dadas las buenas prácticas recomendadas por INCIBE[4]. Estas proponen políticas muy laxas a los equipos que se encuentran en las partes más profundas de la topología, lo que se traduce en más vectores de ataque. No obstante, el objetivo final del algoritmo será, en la medida de lo posible, inundar la red de APTs.

La conjunción de todos los elementos planteados permite tener un entorno educativo más complejo y completo, donde el alumno podrá desarrollar mejor sus habilidades. De esta manera damos por resuelta la pregunta 3.

[4]https://www.incibe-cert.es/blog/buenas-practicas-configuracion-red-inteligente

## V. Implementación

La implementación ha sido desarrollada en Python y, como bien se ha definido en la arquitectura, contará con diferentes APIs acompañadas del patrón adaptador para cada una de las fases. Este patrón jugará un rol importante, puesto que, permitirá la colaboración entre objetos con interfaces incompatibles.

Antes de continuar comentaremos la terminología que se considera indispensable empleada durante la implementación de la solución:

- Agent: APT depositado en la víctima.
- Abilities: Acciones que el *Agent* puede ejecutar.
- Facts: Implementaciones de *Abilities* dependientes del sistema operativo.
- Adversaries: Conjunto de *Abilities*.
- Operation: Ejecución de *Facts* (*Abilities*) sobre *Agent*.

Para desarrollar esta solución se han definido todas las APIs comentadas en la arquitectura. A continuación, se explican el patrón adaptador y la API creada para controlar los generadores de APTs y/o simuladores de ataque. Esta es la parte que más complejidad e importancia tiene, ya que, es la base para implementar la matriz creada. Cualquier limitación que presente esta parte la heredará el resto de la solución.

En el código del patrón adaptador creado se han implementado las funciones de creación, lectura, actualización y borrado (Create, Read, Update, Delete, CRUD en inglés) para las operaciones y las habilidades, como se puede apreciar en el siguiente listado:

Listing 1. Implementación de la interfaz de APT_controller

```
clase APT_controller(Singleton):

    #AGENTS
    interfaz recuperar_agents()
    interfaz recuperar_agents_detail(id)
    interfaz borrar_agent(id)

    #ABILITIES
    interfaz recuperar_abilities()
    interfaz recuperar_abilities_by_tactic(id)
    interfaz recuperar_abilities_by_tactic_string(id)
    interfaz recuperar_ability_details(id)
    interfaz anadir_ability(ability)
    interfaz borrar_abilities(ids)

    #ADVERSARIES
    interfaz recuperar_adversaries()
    interfaz recuperar_adversarie_details(id)

    #OPERATIONS
    interfaz recuperar_operations()
    interfaz recuperar_operation_details(id)
    interfaz anadir_operation(operation)
    interfaz borrar_operations(id)
    interfaz anadir_fact_a_operation(fact)
    interfaz recuperar_facts_de_operation(id)
```

Las interfaces definen métodos que posteriormente serán redefinidos por las clases hijas, donde se implementará la funcionalidad especifica de cada herramienta. Además, la clase utiliza el patrón Singleton, consiguiendo así evitar un posible solapamiento entre diferentes instancias.

Como se vio en el estado del arte, existen simuladores de ataque de gran calidad. Puesto que la herramienta persigue extender y concatenar herramientas y funcionalidades, a continuación, se verá parte del código de la API desarrollado para poder controlar Caldera, concretamente la sección de los *Agents* y *Abilities* como listado:

Listing 2. Implementación de la API de Caldera

```
clase Caldera(APT_controller):
    Caldera(direccion_red):
        api = Caldera_API(direccion_red)

    #AGENTS
    funcion recuperar_agents():
        return api.get_agents()
    funcion recuperar_agents_detail(id):
        return api.get_agents(id)
    funcion borrar_agent(ids):
        return api.delete_agents(borrar, ids)

    #ABILITIES
    funcion recuperar_abilities():
        return api.get_abilities()
    funcion recuperar_abilities_by_tactic(id):
        return formateo(api.get_abilities(id))
    funcion recuperar_abilities_by_tactic_string(id):
        return formateo(api.get_abilities(id))
    funcion recuperar_ability_details(id):
        return api.get_abilities(id)
    funcion anadir_ability(ability):
        return api.add_ability(ability)
    funcion borrar_abilities(ids):
        return api.delete_abilities(ids)
```

Puesto que Caldera cuenta con una API propia, la cual funciona mediante peticiones *HTTP*, se ha optado por la creación de una clase adicional que contenga las llamadas "en crudo" a dicha API. Esta decisión se ha basado en las reiteradas incoherencias que contiene la API nativa, con vistas a poder solventar cambios futuros fácilmente. En el listado mostrado a continuación se hará el control de las diferentes respuestas y errores:

Listing 3. Implementación de la API de Caldera

```
clase Caldera_API():
    TOKEN = 'token_api_caldera'

    Caldera_API(direccion_red):
        API_HTTP = establecer_conexion(direccion_red, token)

    #AGENTS
    funcion get_agents():
        return API_HTTP.get('api/v2/agents')
    funcion get_agents(agent):
        return API_HTTP.get('api/v2/agents', agent)
    funcion remove_agents(ids):
        return API_HTTP.post('api/v2/agents', ids)

    #ABILITIES
    funcion get_abilities():
        return API_HTTP.get('api/v2/abilities')
    funcion get_abilities(id):
        return API_HTTP.get('api/v2/abilities', id)
    funcion add_ability(ability):
        return API_HTTP.put('api/v2/abilities', ability)
    funcion delete_abilitiy(id):
        return API_HTTP.delete('api/v2/abilities', id)
```

Vemos en este último caso el uso de métodos CRUD: i) creación, donde se crean los objetos (set); ii) lectura, donde se recuperan objetos (get), iii) actualización, donde se sobreescriben los objetos; iv) y borrado, donde se eliminan los objetos (delete). Para poder acceder a la API de Caldera hacemos uso de un token, para seguidamente usar los métodos descritos por la documentación.

Es gracias a la unión de ambas decisiones de diseño que la solución propuesta puede desligarse de herramientas, evadir los inconvenientes que esta contiene y automatizar la metodología de ataque. De esta manera afirmamos que un simulador de APTs puede ser lo suficientemente autónomo, respondiendo a la pregunta 2.

## VI. Prueba de concepto

A la hora de realizar una primera implementación completa se han incluido el uso de los siguientes componentes: *nmap*, para la fase *Discovery*; *metasploit*, para la fase *Exploitation*; *PEASS-ng*, para la fase *Inside*; y *SSH*, para la fase *Pivoting*. Dado que hay una herramienta por fase, se puede completar el esquema expuesto en la Fig. 2, realizando así una prueba de concepto que verifique la implementación propuesta.

Para mostrar el correcto funcionamiento de la herramienta se realizará una prueba sobre el siguiente escenario:

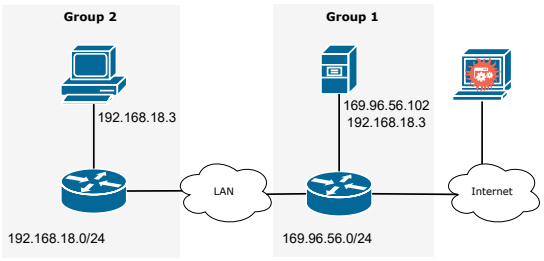


Figura 3. Arquitectura propuesta: Topología de la prueba de concepto

El grupo 1 está conectado a Internet y contiene un servidor. Este servidor es una plataforma de almacenamiento personal en la nube y cuenta con el protocolo File Transport Protocol (Protocolo de transporte de archivos o FTP) vulnerable. Por otra parte, el grupo 2 alberga un equipo personal con un servicio activo, para conexión remota: Secure Shell (Shell seguro o SSH) vulnerable. Ambos grupos están interconectados y todos los dispositivos tienen conexión a Internet, aunque solo el servidor es accesible desde fuera de la red.

Para realizar la prueba ejecutaremos la herramienta con todos los parámetros por defecto, es decir, esta intentará infectar todo equipo a su alcance y no hará tiempos de espera entre fases. Pese a que esto último no es óptimo para el entrenamiento de un usuario, se ha optado por esta política para realizar pruebas tempranas en el desarrollo de la herramienta. Dado que el servidor es accesible desde la red,


Figura 4. Resultados de la ejecución del ataque PoC

se empezará el ataque indicando únicamente la dirección del servidor.

Como se aprecia en la Fig. 4, la ejecución va iterando sobre la metodología planteada. En primer lugar se ejecuta la fase *Discovery*, donde se consigue localizar un equipo con servicios vulnerables en la dirección IP dada. Seguidamente se procede con la fase *Exploitation*, la cual explota la vulnerabilidad encontrada con mejor puntuación en la fase anterior, escogiendo en este caso la única encontrada y obteniendo como resultado un nuevo agente en el sistema vulnerado. Es entonces cuando la aplicación procede a la recolección de información, esta vez en la fase *Inside*, obteniendo en el primer equipo el sistema operativo, las interfaces de red e incluso los posibles CVE. Dado que el usuario obtenido es "root", se omite la etapa en la que se escala privilegios. Por ello se procede a establecer los medios para pivotar, en la fase *Pivoting*. Finalmente se vuelve a ejecutar todo el procedimiento descrito anteriormente desde el nuevo agente para encontrar el último equipo, con la salvedad de que este último deja sin posibles caminos al algoritmo, haciendo que este termine.

Vemos con esta prueba de concepto que, basándonos en la metodología, obtenemos un resultado exitoso sobre la implementación aportada. Esto plantea un gran desarrollo en las herramientas ofensivas y en los Cyber Ranges, respondiendo en esta ocasión de manera afirmativa la pregunta 2.

## VII. Conclusiones

La ciberseguridad juega un papel fundamental en la sociedad actual. Para ello, es fundamental dotar de competencias ofensivas y defensivas a cualquier usuario de sistemas informáticos. Para esto último, los Cyber Ranges han demostrado ser una herramienta crucial, pudiendo albergar agentes de ataque realistas, para entrenar los usuarios de forma automática.

En este sentido, la principal motivación detrás de nuestra propuesta es que las matrices actuales no se pueden orientar hacia una implementación que automatice los ataques de manera adaptativa. Es por ello que se ha desarrollado una matriz que generaliza lo suficiente como para poder englobar todos los conceptos necesarios y evitar perder información o utilidad por el camino. A partir de ella hemos desarrollado una solución donde se implementa una herramienta por etapa que, de manera modular, puede albergar la funcionalidad requerida por la metodología. Para probarlo se ha demostrado el correcto funcionamiento mediante una prueba de concepto, avalando así que la metodología y arquitecturas propuestas son prometedoras. En ella se presenta una topología donde se atacan los diferentes recursos disponibles y se consiguen vulnerar los equipos de manera progresiva.

En este artículo hemos planteado tres preguntas, las cuales hemos demostrado y respondido afirmativamente a lo largo del texto. Esto nos conduce a seguir desarrollando, dado que la implementación no cuenta con la suficiente madurez como para ser usada en un entorno de producción, un microservicio que pueda ser utilizado por un Cyber Range. Debemos fomentar la ciberseguridad y formar con rigor a los profesionales que así lo deseen, no podemos despreocuparnos de este aspecto en la Unión Europea.

Como trabajo futuro planteamos el estudio de: técnicas que planteen tiempos de espera realistas, modelados por dificultades variables y que mantengan la motivación del estudiante; y la implementación de inteligencia artificial, sustituyendo la etapa *Logic* y dotando de un nuevo nivel de realismo.

## Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Universidades español vinculado a la Unión Europea a través del programa NextGenerationEU, con cargo a la beca postdoctoral Margarita Salas (172/MSJD/22).

## Referencias

[1] "Viewpoint: For stronger tech, Europe must spend more on defence and research," https://sciencebusiness.net/viewpoint/Sovereignty/stronger-tech-europe-must-spend-more-defence-and-research, accessed: April 20, 2023.

[2] World Economic Forum, "Wef global cybersecurity outlook 2022," World Economic Forum, Tech. Rep., 2022. [Online]. Available: https://www3.weforum.org/docs/WEF_Global_Cybersecurity_Outlook_2022.pdf

[3] W. Xiong, E. Legrand, O. Åberg, and R. Lagerström, "Cyber security threat modeling based on the mitre enterprise att&ck matrix," *Software and Systems Modeling*, vol. 21, no. 1, pp. 157–177, Feb 2022.

[4] T. Yadav and A. M. Rao, "Technical aspects of cyber kill chain," in *Security in Computing and Communications*, J. H. Abawajy, S. Mukherjea, S. M. Thampi, and A. Ruiz-Martínez, Eds. Cham: Springer International Publishing, 2015, pp. 438–452.

[5] C. P. Castaño, "Hacktricks," https://book.hacktricks.xyz/welcome/readme, Accessed on April 26, 2023.

[6] X. Larriva-Novo, V. A. Villagra, O. Jover, M. Sanz Rodrigo, C. Sánchez-Zas, and M. Álvarez Campana, "Simulador de APTs realistas basado en el marco de MITRE ATT&CK," in *VII Jornadas Nacionales de Investigación en Ciberseguridad (JNIC '22)*, 2022, pp. 138–141.

[7] S. Y. Enoch, Z. Huang, C. Y. Moon, D. Lee, M. K. Ahn, and D. S. Kim, "Harmer: Cyber-attacks automation and evaluation," *IEEE Access*, vol. 8, pp. 129 397–129 414, 2020.

[8] E. Walter, K. Ferguson-Walter, and A. Ridley, "Incorporating deception into cyberbattlesim for autonomous defense," 2021.

[9] R. D. Medenou Choumanof, S. Llopis Sanchez, V. M. Calzado Mayo, M. Garcia Balufo, M. Páramo Castrillo, F. J. González Garrido, A. Luis Martinez, D. Nevado Catalán, A. Hu, D. S. Rodríguez-Bermejo, G. R. Pasqual de Riquelme, M. A. Sotelo Monge, A. Berardi, P. De Santis, F. Torelli, and J. Maestre Vidal, "Introducing the cysas-s3 dataset for operationalizing a mission-oriented cyber situational awareness," *Sensors*, vol. 22, no. 14, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/14/5104

[10] P. Nespoli, M. Albaladejo-González, J. A. Pastor Valera, J. A. Ruipérez-Valiente, and F. Gómez Mármol, "Capacidades avanzadas de simulación y evaluación con elementos de gamificación," in *VII Jornadas Nacionales de Investigación en Ciberseguridad (JNIC '22)*, 2022, pp. 55–62.

[11] J. Vykopal., R. Oslejsek., P. Celeda., M. Vizvary., and D. Tovarnak., "Kypo cyber range: Design and use cases," in *Proceedings of the 12th International Conference on Software Technologies - ICSOFT,*, INSTICC. SciTePress, 2017, pp. 310–321.