Series de Potencia como un Medio Básico para Autenticar Multiagentes de Software: Una Simulación Simple

Miguel Torrealba
mtorrealba@usb.ve

Departamento de Computación y Tecnología de la Información, Universidad Simón Bolívar, Baruta, Venezuela

Resumen: Mientras los computadores más se han diseminado en los ambientes cotidianos, la inseguridad que sobre ellos incide también se ha hecho notar. Los sistemas con soporte de inteligencia artificial, como aquellos con agentes autónomos o de máquinas que aprenden por si mismos no escapan de esa situación y hoy, peligrosamente se emplean para apoyar la automatización de plantas industriales y otras áreas críticas. Aún así, una gran cantidad de esos sistemas tecnológicos continúan ofreciendo como principal mecanismo de identificación a sus usuarios, el viejo esquema de emplear una contraseña para su acceso. Sin embargo, desde hace tiempo los mecanismos de protección que siguen esa orientación, en sistemas comerciales y de uso libre resultan de baja seguridad y por ello se impone ofrecer mejores alternativas en esa materia. Este trabajo describe una propuesta para autenticar a agentes inteligentes de software con base a un protocolo reto – respuesta, que se sustenta en los cálculos de series de potencias numéricas y funciona en un marco de ZKP. Se presenta también una simulación de eventos discretos básica, que permite apreciar algunas posibilidades simples de control e inseguridad sobre la misma.

Palabras Clave: Autenticación; Agentes Autónomos de Software; Reto - Respuesta; ZKP; Series de Potencia.

1. Introducción

Durante la pasada década de los 90's la creciente complejidad en el desarrollo de software fue una de las razones que condujo a impulsar el interés en el empleo del paradigma de *agentes inteligentes*. Era frecuente que se deseara además que tales agentes *aprendieran* de sus experiencias y en *tiempo real*, respondieran con acciones adecuadas a los cambios de las condiciones en que operaban. Una década después, con algo de más solidez en las investigaciones, entre las primeras definiciones ampliamente conocidas para esos agentes estuvo la de la Universidad de Liverpool:

"*Un agente es un sistema de computación que está situado en algún ambiente y es capaz de ejecutar acciones autónomas sobre ese ambiente en orden para satisfacer sus objetivos de diseño.*" [1]

En el caso de los *Sistemas de Multiagentes* (MAS) era común que estos operaran sobre sistemas interconectados para constituir grandes sistemas distribuidos y que se les delegara la ejecución de tareas específicas, en aras de mejorar el control de los mismos; una característica significativa en sistemas críticos o extensamente grandes. De modo que el oportuno y preciso intercambio de mensajes entre los agentes físicamente separados, resulta básico en este paradigma. Pero para alcanzar esa meta, es necesario que los agentes previamente se reconozcan entre si y que ello resulte además confiable, algo propio del área de la *ingeniería en seguridad* [2]. *Autenticar* es el nombre que recibe este procedimiento y desde 1991, la Recomendación CCITT X.800 de la Unión de Telecomunicaciones Internacional lo describió como un *servicio de seguridad* para *sistemas abiertos* [3].

En computación y redes, la operación de reconocer a otra entidad ha sido estudiada desde hace décadas atrás e impone el uso previo de algún *esquema de identificación*, algo tampoco trivial. La *criptografía* se convirtió en la ciencia de soporte para instrumentar soluciones de estos requerimientos [4] y se popularizaron orientaciones de este tipo que están sustentadas en simplificaciones del *protocolos reto respuesta*, como es el uso de una *contraseña* para autenticación. Aún así, tal aproximación en el área es conocida como un mecanismo de autenticación débil [5] y únicamente es por lo reducido de sus costos [6], al igual que por la disponibilidad y variabilidad de *bibliotecas de funciones y métodos de software* para desarrollar modernos sistemas informáticos, que se ha extendido su empleo. Sin embargo, notables y públicos *incidentes* sobre *sistemas en línea* de los reconocidos proveedores y prestadores de servicio como Yahoo®, Gmail®, LinkedIn®, Adobe®, Twitter® y el sitio web para citas eHarmony®, vuelven a poner a la vista las serias debilidades de esa aproximación [7].

Así resulta entendible admitir la necesidad de buscar otros medios y técnicas que permitan autenticar MAS que funcionan sobre sistemas críticos. Este trabajo describe el uso básico de una *serie de potencias numérica*, como un elemento adicional al soporte de protocolos criptográficos tradicionales de las redes de computadoras comerciales, que conforman la moderna Internet, para instrumentar otros procedimientos de autenticación en MAS que no se sustenten en la clásica contraseña. La explicación se acompaña también con una simulación por computadoras simple, que ofrece primeras señales estadísticas de la viabilidad de la propuesta, en materia de fiabilidad informática. El escrito ha sido

estructurado para exponer el fundamento conceptual de las bases del diseño del sistema, luego sus limitaciones y alcance, posteriormente se presentan detalles técnicos de la simulación, la comprobación, los resultados y las conclusiones obtenidas.

2. Método de Trabajo Aplicado

Esta propuesta se realizó siguiendo una aproximación científica con bases filosóficas orientadas en el *empirismo* y *positivismo* [8], por lo cual la formulación lógica se apoyó en el *racionalismo*, mientras que lo *experimental* se planteó como el eje central de la comprobación. La investigación está enmarcada en el *paradigma cuantitativo* [9] y se inscribió como una ejecución alineada con los principios de la teoría de *proyecto factible*, así como un enfoque *deductivo*, que progresivamente permitió emitir las conclusiones finales.

3. Descripción y Base de Confianza en el Sistema Propuesto

El sistema alterno que presentamos se fundamenta en el intercambio de información confidencial entre partes que estarán separadas, pero operarán con un propósito común. Estas entidades serían los agentes inteligentes y de software, que bajo esta concepción tendrán una articulación de fase inicial en un mismo e único entorno, su génesis. Luego, se separarán y operarán en distintos ambientes. La comunicación en esa segunda etapa ocurrirá con base a protocolos criptográficos comunes que hoy ofrecen los protocolos de comunicación y redes, siendo *Secure Sockets Layer*/*Transport Layer Security (SSL/TLS)* [10] el más frecuente en el mundo TCP/IP Protocol Stack Suite. Este cambio de localización resulta clave para la ejecución de una base de instrucciones, que permitirán establecer las *relaciones de confianza* que sostendrán el modelo propuesto. Es necesario recordar que desde inicio de los noventa se dejó en claro que los profesionales de seguridad conciben a esta, más en términos de confianza y confiabilidad que con la absoluta idea de que nada será perturbado, por lo que a comienzos del tercer milenio Spafford, Schwartz y Garfinkel hicieron énfasis sobre esta característica, expresando además su vínculo con el diseño de sistemas al afirmar, en la tercera edición de una obra reconocida en el área, que:

"*Desarrollar confianza adecuada en su sistema de computación requiere cuidadoso pensamiento y planificación. Decisiones operacionales deberán estar basadas sobre la política percibida y el análisis de riesgo.*" [11]

Ello quiere decir que la base de confianza que soportará la interacción comunicativa entre agentes, se forja en su fase inicial y es que, la información secreta que sustentará el reconocimiento de identidades se genera y comparte solamente en ese momento. El hecho de que las diferentes partes estén reunidas en un mismo ambiente, resuelve la necesidad de recurrir a un *tercero de confianza* o algún tipo de *autoridad certificadora* que avale la correctitud de la información confidencial. Cada agente inteligente la recibirá directamente durante el proceso de origen de la misma y luego podrá desplazarse como es una de sus características de movilidad comunes, dejando en sus manos resguardar apropiadamente su contenido. Posteriores ajustes o ampliaciones de los elementos de confianza, pueden seguir el patrón descrito anteriormente.

Por otra parte, una diferencia clave con el esquema común de contraseña será que el contenido oculto y confidencial nunca deberá ser transmitido. Esto enmarca la propuesta dentro de la teoría de *Pruebas de Conocimiento Cero – Zero Knowledge Proof (ZKP)* que se alínean con esquemas donde una parte comprueba a otra, remota, que una declaración es cierta sin revelar nada de la exactitud de lo que clama [12]. En nuestro trabajo esto se logró dejando que el procedimiento de autenticación se sustente en una instrumentación del tipo *reto respuesta* [13] más sofisticada que utilizando una contraseña; esto significa que principalmente se sustenta con matemáticas. De esa manera es que a partir de preguntas y respuestas diferentes, no fácilmente predecibles, de una *única sesión* interactiva y tan variante en el tiempo -se apoya en la incorporación de "nonces"- es que los *agentes verificadores* pueden decidir si están en presencia de quien sostiene el *agente demandante* que están. Estas consultas y respuestas deberán formularse de un modo que tengan como raíz la información confidencial que cada uno de los agentes maneja y preserva secretamente, estableciendo con solidez que si esta no es correcta la respuesta que se produzca sea equivocada. Resulta necesario indicar que para esta propuesta, este tipo de protocolo de autenticación, requiere que el canal que comunica a ambas partes tenga una probabilidad baja de sufrir un ataque, por lo que es indispensable que todos los mensajes viajen encriptados. Un formato básico que el diálogo puede seguir es:

Agente Verificador → *Reto*: {<operación matemática> <parámetros numéricos de la serie>}
Agente Demandante → *Respuesta*: valor "hash" del {valor resultante}

La pregunta puede variar en complejidad según sea el nivel de incertidumbre que predomine en el ambiente en que opera el agente verificador. Adicionalmente, la interacción puede incluir numerosas preguntas y respuestas, así como deliberadas consultas erróneas para dificultar que cualquier *escucha furtiva* pueda analizar patrones en las respuestas. Una condición de alerta ambiental, derivada de una escala preliminarmente acordada, podría elevar o bajar las exigencias de comprobación con el protocolo. Es de notar que las respuestas deberán ser emitidas como resultado de alguna función unidireccional tipo "hash", que contribuya a ocultar aún más el valor real de la respuesta, pero que haga fácil la comparación con el agente que validará el mensaje.

4. Series de Potencia como Centro Numérico del Proceso de Autenticación

El esquema matemático de la propuesta se fundamenta en trabajar con algún tipo de serie de potencia que podría estar acotada en un intervalo específico. Las selecciones y precisiones que se sigan serán establecidas durante la fase inicial de generación de la información secreta.

Así por ejemplo, el juego de funciones a usar, los intervalos de cada una, el número de agentes que operarán en conjunto, la cantidad de términos que incluirá la serie y el número de cifras decimales que deberá contener las respuestas, son algunos de los posibles parámetros a referir en consultas y/o respuestas del protocolo. Para un atacante, en comparación con la aproximación de contraseña de acceso este esquema es más complejo de superar, pero para los desarrolladores ofrece la ventaja de que no resulta costoso o arduo de implementar, ya que se sustenta en cálculos que pueden ser adecuados a la sensibilidad de los sistemas críticos y no requieren de "hardware" especial.

De forma que cada agente puede conocer las potenciales operaciones y funciones que pueden ser usadas, según alguna correspondencia numérica simple que se elaboró durante la etapa del génesis de la información del protocolo y que sería confidencial fuera del grupo de agentes. Esto hace que no sea necesario transmitir toda la información durante las consultas, reduciendo la cantidad de bits en cada mensaje y escondiendo los elementos básicos de cada diálogo. La Tabla I expone una posible correspondencia a usar.

Tabla I: Posible Correspondencia y Parámetros de Series

Número	Función	Límite Inferior	Cantidad de Cifras Decimales
0	$1 - x + (2!)^{-1}x - (3!)^{-1}x^3 + (4!)^{-1}x^4 – (5!)^{-1}x^5 + ...$	*0*	*5*
1	$1x^1 + 2x^2 + 3x^3 + 4x^4 + ...$	*1*	*4*
2	$½x^1 + ⅓x^2 + ¼x^3 + 1/5x^4 + ...$	*1*	*6*
3	$1x^1 - 2x^2 + 3x^3 - 4x^4 + ...$	*3*	*3*
4	$x + ¼x^2 + 1/9x^3 + 1/16x^4 + 1/25x^5 + 1/36x^6 + ...$	*0*	*5*

La exposición superior deja por fuera otros parámetros como son la cantidad de agentes del grupo, el número de términos que debe contener la serie, la identificación de cada uno de ellos y las veces previas que han interactuado los agentes. Esos valores, también restringidos para los agentes, deben ser incorporados en las consultas y por eso la Tabla II presenta algunos de esos posibles retos a responder.

Tabla II: Posibles Retos a Utilizar en el Protocolo de Autenticación

Número del Reto	Reto	Argumento
1	Calcular la serie escogida previamente con la cantidad de agentes del grupo, para posteriormente restarle el número 5	x=10
2	Calcular el valor de la serie escogida previamente. Luego multiplicar lo obtenido por 100	x=2
3	Indicar cantidad de veces que se ha interactuado a partir de la fecha señalada	DD/MM/AAAA
4	Indicar la fecha y hora de inicio del génesis, desplazando hacia atrás 3 días	No aplica
5	Responda con "1" si es cierto que la cantidad de agentes del grupo multiplicado por 123 es el valor que se indica seguidamente. En caso contrario responda con "0"	10947
6	Responda con "1" si es cierto que el número de identidad mía multiplicado por 11 es el valor que se indica seguidamente. En caso contrario responda con "0"	7.33333
7	Responda con "1" si es cierto que la cantidad de agentes en el grupo sumado con -2 es el valor que se indica seguidamente. En caso contrario responda con "0"	11
8	Responda con "1" si es cierto que la cantidad de contactos previos sumada con 3 es el valor que se indica seguidamente. En caso contrario responda con "0"	15

Se puede concebir la selección de la pregunta a efectuar para que se produzca aleatoriamente, de forma que el agente demandante no sepa de antemano la respuesta que deberá emitirse. Esto puede hacerse aún menos predecible, si los argumentos no se incluyen originalmente y se señala además que estos serán comunicados al momento de preguntar. Ello podría imponer el empleo de algunos formatos adicionales para formular los retos, al igual que las respuestas, cosa que haría más compleja la autenticación pero también más sólida su escalabilidad en condiciones variables de incertidumbre. Y es que dichos parámetros serán de uso único por sesión y como son numéricos, pueden ser vistos como los tradicionales "números de un único uso" del mundo criptográfico; lo que en inglés se cono como "numbers once" que se acostumbra a contraer lexicográficamente como "*nonce*".

Para hacer más comprensible el funcionamiento del protocolo seguidamente ilustraremos con un ejemplo de potenciales diálogos, la idea de trasfondo del mecanismo. Por razones de mayor comunicación con el lector, deliberadamente se ignorará que el protocolo impone que las respuestas se transmitirían únicamente como valores de una función "hash", no tal como se mostrará a continuación. La Tabla III presenta la información que previamente manejan los agentes.

Tabla III: Información Local para el Proceso de Autenticación

Datos ocultos que maneja cada agente:	Datos propios y ocultos que maneja el agente verificador:	Datos propios y ocultos que maneja el agente demandante:
Fecha y hora de inicio del génesis: *10-08-2020/14:23:00*	Identificación propia: *-1.33333 que corresponde a "- 2^3/3!"*	Identificación propia: *0.66666 que corresponde a "2^4/4!"*
Número de agentes en el grupo: *3*	Identificación remota de su interlocutor: *0.66666 que corresponde a "2^4/4!"*	Identificación remota de su interlocutor: *-1.33333 que corresponde a "- 2^3/3!"*
Función preestablecida durante el génesis: Número *0*		
Valor del límite inferior de los términos que conforman la serie: *0*	Fecha del intercambio: *11-08-2020*	Fecha del intercambio: *11-08-2020*
Número de términos de la serie: *7*	Número de contactos previos: *12*	Número de contactos previos: *12*

Se supone además que el agente demandante acaba de enviar un mensaje de "Hola" (Hello), que señala su presentación e incluye su identificación "0.66666". La Tabla IV presenta un posible diálogo para autentificar a una parte:

Tabla IV: Posible Diálogo Interactivo entre Partes para Autenticar

Intercambio	Verificador	Demandante
1	*Reto número 2*	
2		*0.33333 x 100 = 33.33333*
3	*Reto número 6*	
4		*0 (ya que el demandante obtuvo como respuesta 14.66667)*

5	*Reto número 4*	
6		*07-08-2020*
7	*Reto número 8*	
8		*0 (ya que el demandante obtuvo como respuesta 1)*
9	*Reto número 7*	
10		*1 (ya que el demandante obtuvo como respuesta 15)*

Después de estas cinco preguntas, el agente verificador puede decidir si realmente está interactuando con la identidad "0.66666" que el agente demandante sostiene tener. También es factible que en seguida se intercambien los roles y que el anteriormente agente demandante proceda a autenticar a su contraparte. De esa forma los agentes pueden comunicarse con mayor confiabilidad que cuando usan únicamente la transmisión del valor de la contraseña de acceso, que comúnmente en algún instante deben traducir al texto plano. Para dificultar aún más, potenciales ataques del tipo "hombre en el medio – Man in the Middle (MITM)", también puede incorporarse que después de recibir el mensaje inicial del agente demandante, el agente verificador no responda de inmediato para ser quien espera un breve lapso y posteriormente procede a ser quien inicia el nuevo diálogo con un mensaje de "hola", seguido de su autenticación; esto a modo de los viejos "modems" que hacían la operación de "te llamo de vuelta (call-back)" para dificultar haber sido objeto de una invocación falsa. Otras verificaciones podrían incluir la interrupción de la comunicación para activar una nueva comprobación, que bien podría incluir preguntas sobre la comprobación anterior, por ejemplo "¿En la autenticación anterior se aplicó algún tipo de esquema de cuelgo y te llamo?". Esto quiere decir, que existe flexibilidad para hacer más ardua el proceso de establecer la autenticación, algo que la clásica consulta de una contraseña ignora. La decisión de hacer una o más consultas puede variar también y ello es importante, ya que complica la automatización de ataques a ciegas, hoy de uso común.

5. Limitaciones y Extensiones de la Propuesta

En el texto Ingeniería Criptográfica sus autores Fergunson, Schneier y Kohno afirman tempranamente lo siguiente:

"*Así que aquí está su primera lección en criptografía: mantenga una mente crítica. No confíe ciegamente en nada, incluso si eso está impreso. Usted aprenderá pronto que tener esa mente crítica es un ingrediente esencial para lo que denominamos 'paranoia profesional'.*" [14]

Este pensamiento apunta a una visión moderna en seguridad digital que rompe con aquella de pasadas décadas, que empleaba términos como garantizar e inviolabilidad, donde se tenía una creencia de que la tecnología por sí sola bastaba para asegurar a ella misma. Un enorme número de incidentes, sobre notables organizaciones y reconocidas instituciones gubernamentales, permitieron re-evaluar premisas ideológicas y suposiciones para decantarse por la *gestión de riesgos* y un punto de partida, que considera la vigilancia y evaluación contínua de los eslabones débiles en cualquier cadena de protección.

De modo que la propuesta no es concebida como capaz de soportar cualquier ataque, por el contrario se parte de la presunción de que es ideal considerar los desarrollos tecnológicos bajo una perspectiva de ser *sistemas sociotécnicos* que encaran amenazas y pueden ser sobrepasados. En consecuencia, deben ser elaborados para ofrecer capacidades de restauración así como pedir apoyo al exterior de ellos.

Desde el punto de vista lógico un primer análisis de los esquemas tradicionales de contraseña de acceso debe considerar que estos fueron concebidos décadas atrás, bajo una realidad distinta a la de este tiempo, por lo que se desconocían peligros que hoy están activos y es que surgieron años más tarde de la planeación de la contraseña. Además muchas pautas de diseño originales fueron ignoradas al usar instrumentaciones diferentes a las que los ingenieros suponían en su momento, tal como aconteció con la misma Internet, que durante los años noventa fue escogida como plataforma para el *Comercio Electrónico* y la *Economía Digital*, algo que dista mucho de la historia en DARPA, que es el inicio de como ésta fue creada [15]. "*En 1973 la única tecnología criptográfica que podíamos tener en nuestras manos era clasificada*" ha expresado Vinton Cerf, uno de los creadores de la red de redes. Esta frase revela lo distinto del mundo de la seguridad en esa época e induce a pensar que mucho de lo que ahora se dispone fue agregado posteriormente, no durante su diseño; este tipo de acontecimientos tiende a incidir significativamente en los resultados de baja efectividad en cualquier sistema de seguridad.

En esta sección es indispensable señalar que en ciertos aspectos el área de la *ciberseguridad* puede contrastar con la del desarrollo de software o con la ciencia de la computación, debido a que con frecuencia los medios para comprobar que se ha logrado satisfacer un requerimiento de software se basan en la evaluación práctica de que satisfactoriamente se tiene capacidad para realizar alguna funcionalidad operativa. Hace casi cinco décadas atrás que Saltzer y Schroeder sentaron las bases en la construcción de software confiable y adicionalmente, expresamente refirieron el problema que describimos:

"*El objetivo de un sistema seguro es evitar todo uso no autorizado de información, un tipo de requisito negativo. Es difícil demostrar que se ha cumplido este requisito negativo, porque hay que demostrar que se han anticipado todas las amenazas posibles. Por tanto, una visión amplia del problema es la más apropiada para ayudar a asegurar que no aparezcan lagunas en la estrategia. Por el contrario, una estrecha concentración en los mecanismos de protección, especialmente aquellos que son lógicamente imposibles de vencer, puede conducir a una falsa confianza en el sistema en su conjunto.*" [16]

En consecuencia resulta necesario diseñar sistemas con un *modelo de amenazas* en mente, con una *visión holística* y presumiendo que la seguridad absoluta no es factible de

obtener. La propuesta ahora expuesta pretende dejar en claro que tiene fortalezas, así como debilidades con respecto a la opción del uso de la contraseña de acceso, en especial sobre el área de la fiabilidad de propósito, donde puede resultar igual de efectiva que esta, pero con la ventaja de ser más compleja para subvertir.

Se elaboró considerando repeler los peligros que el modelo de amenaza con las contraseñas contempla. Por lo tanto, las principales amenazas son la *escucha furtiva del canal* y la *suplantación de un actor*. Para el primer problema, la propuesta puede enfrentar un *ataque tipo diccionario* o de *fuerza bruta* [4] sin ser comprometida, ya que opera con un protocolo enmarcado en ZKP. El segundo inconveniente demanda incorporar un método complementario de prevención, que notifique al exterior del agente, *el riesgo* de perder el control sobre la información confidencial; más adelante discutiremos una posible instrumentación de eso.

Otra bondad de la propuesta como herramienta de autenticación, radica en que no deja el peso de su fortaleza, en la buena escogencia de contenido que realice un usuario final y es que la experiencia muestra que, con frecuencia, esta aproximación resulta pobre y ello se convierte en el eslabón débil de la cadena de protección, sin importancia de cuán bueno sea el mecanismo que se emplea. Es por ello que los ataques de *ingeniería social* clásicos, como el "Phishing", con décadas de tiempo atrás que se originaron y que continuamente son objeto de campañas de concientización, aún siguen teniendo resultando desvastadores para sistemas bancarios y comerciales. A esto hay que agregarle la preocupante realidad de ataques cada vez más sofisticados y creativos. Posiblemente sea ideal citar una opinión reciente que puede ayudar a comprender el contexto tecnológico de la inseguridad digital del momento:

"*Los ciberatacantes de hoy comprenden completamente las deficiencias de las tecnologías de seguridad que despliegan las organizaciones, así como la forma en que estas se implementan. Por ejemplo, los atacantes saben que casi todas las redes de hoy están protegidas por un firewall. Sin embargo, los atacantes aún saben cómo obtener acceso a las redes internas con bastante eficacia, pasando a través de los firewalls.*" [17]

Por otra parte, es necesario advertir que este trabajo no afirma que la propuesta sea invulnerable, por el contrario se diseñó bajo la creencia de que para proveer los niveles de protección que se espera dar, requiere acompañarse con complementos procedimentales de corte de estrecha supervisión y en *tiempo real*. Un ejemplo de un escenario posible que puede reforzar el mecanismo expuesto, es suponer que bajo una situación de ataque cada agente dispondría de algún mecanismo que, temporalmente, alteraría algún o algunos valores de las tablas confidenciales. Esto debería realizarse sin dejar rastros fáciles que permitan descubrir a un atacante como aconteció la alteración. De forma que si el ataque es satisfactoriamente repelido, se pueda restaurar sus contenidos a su forma original, pero si el agente cree que resultará sobrepasado debería enviar un mensaje de petición de auxilio y luego, eliminar cualquier posibilidad de recuperar la tabla. Se llegaría de ese modo a un estado que podría anular al agente por completo dentro del grupo. Otra variación es que bajo cualquier ataque, un agente notifica al grupo su estado y remueve sus tablas confidenciales, dejando en manos de algún nivel superior la restauración de su actividad. Debe recordarse que frente a sistemas críticos como puede ser una planta nuclear o un hospital, la *Política de Uso Aceptable* puede exigir que una inteligencia humana decida sobre la sanidad del sistema y sus componentes antes de que este pueda volver a operar con normalidad. El humano constituye así el último nivel de un esquema que se orienta con la idea de proveer una *defensa en profundidad*.

6. Simulación Discreta y Simple para la Propuesta

Con la intención de explorar, en modo simple y preliminar las posibilidades de fracaso con el proceso de autenticación propuesto, se construyó una *simulación de eventos discretos* [18]. Un procedimiento común que tiende a bajar costos y la literatura especializada desde hace tiempo refleja, por ello una muestra de la idea sobre esto se puede percibir al leer la siguiente frase:

"*Los resultados de una simulación con computadora pueden servir como un puente entre experimentos de laboratorio y cálculos teóricos. En algunos casos podemos esencialmente obtener resultados exactos a través de la simulación y un modelo idealizado que no tenga contraparte en laboratorio. Los resultados de un modelo idealizado pueden servir como un estímulo para el desarrollo de una teoría.*" [19]

No podemos dejar de señalar que esta simulación es apenas la primera de una serie y no pretende establecer conclusiones finales cuando el trabajo apenas inicia, aunque busca experimentar con las posibilidades de que un tercero, examinando el tráfico de comunicaciones entre los agentes, descubra elementos que sostienen lo secreto del mecanismo. Se descarta el hecho de que para ello el atacante debe sobrepasar las protecciones de las comunicaciones que viajan cifradas, con la intención deliberada de examinar cuán fácil es engañar al proceso de autenticación en sí. Otro aspecto relevante de la simulación es que tratando de apegarse a muchas potenciales implementaciones, que se sustentarían en los *generadores de números pseudo-aleatorios*, cosa que sigue siendo común en el mercado, se aplicó el mismo procedimiento en el caso de la simulación, a pesar de que se comprende la debilidad en materia de seguridad de semejante aproximación y que eso puede incidir en la fiabilidad del procedimiento. Este razonamiento no es ilógico, ya que en este campo, desde hace años, la viabilidad acerca de el uso de esos generadores ha sido expuesta. Una muestra de eso se percibe en este texto:

"*Primero debe ser señalado que los computadores estándares son máquinas determinísticas. Así que es completamente imposible generar directamente verdaderos números*

aleatorios. Uno podría, por ejemplo, incluir la interacción con el usuario. Esto es, por ejemplo, posiblemente medir el intervalo de tiempo entre pulsaciones sucesivas del teclado, que por naturaleza se distribuyen aleatoriamente. Pero los intervalos de tiempo resultantes dependen en gran medida del usuario que actúe, lo que significa que las propiedades estadísticas no se pueden controlar ... Además, en el caso de los verdaderos números aleatorios, la velocidad de generación de esos números es limitada cuando estos son baratos, o en caso contrario los generadores resultan costosos.
Esa es la razón por la que los números pseudo aleatorios usualmente se emplean. Ellos son generados por reglas determinísticas." [20]

Un punto de importancia para las pruebas realizadas fue que por razones de facilidad de disponibilidad se utilizó un generador de valores pseudo aletarios proveniente de una biblioteca del lenguaje C "stdlib.h" denominado "rand()". Este produce un entero entre el rango [0, RANDMAX] que posteriormente se normaliza entre (0,1); emplea además el reloj del sistema, como un parámetro que ayuda a generar una semilla de partida con otra función conocida como "srand()". En el computador donde se realizó la simulación el valor de la constante RANDMAX fue 32767 y hay libros universitarios, que señalan que ese generador puede producir secuencias diferentes en la misma máquina o con compiladores diferentes. Sin embargo, en el área también hay autores que dejan claro que lo que es suficientemente estocástico en una aplicación puede no serlo para otra, por lo que ese acotamiento de rango original del generador debía ser examinado en la práctica, a razón de ver si ello favorece las probabilidades de éxito de un atacante.

De manera que bajo ese enfoque fue que se plantearon los escenarios básicos, un *ataque de fuerza bruta* y un ataque basado en adivinar una respuesta binaria. Para ambos se presumía que anteriormente un atacante lograría comprometer la información clasificada de un agente y con base a esta, podría secuestrar al menos un diálogo entre agentes del grupo con el objeto de suplantar a alguno de ellos. Esto podría deberse a que estaba al tanto de que el agente comprometido había podido enviar un mensaje de alerta, cuando determinó que estaba siendo objeto de un ataque y en consecuencia, se pensó que ello inhabilitará al agente caído. En una situación convencional, si se comprometió la contraseña de acceso, ya es factible personificar a una parte de la comunicación. Los escenarios de ataques asumieron otra condición de protección más, que en la realidad le resta importancia a la seguridad que la propuesta ofrece en la realidad y es que se supuso que el atacante podía leer el tráfico cifrado. Eso bien podría suceder tras haber logrado comprometer a un agente, por lo que el atacante dispondría de suficientes parámetros, como son los elementos de negociación inicial de SSL/TLS.

6.1 Ataque de Fuerza Bruta sobre la Comunicación de los Agentes

Para este caso se supone que el atacante desconoce el valor de la variable "x", que es el "nonce" que se provee al plantear el reto. Se facilita la simulación dando por válido que el atacante tiene información acerca de la serie de potencia que se usará en la instancia de autenticación, así como la función "hash" *Algoritmo de Resumen Digital 5 (MD5)* que será empleada en el mismo diálogo. Ignora también la cantidad de decimales que incluirá el resultado, sin embargo puede intentar probar exhaustivamente con un rango de valores y buscar la coincidencia. En caso de que tenga éxito, puede proceder a aplicar otras acciones que le permitan secuestrar la conexión y de esa forma suplantar una de las partes. Bajo un ataque real toda la operación debe acontecer durante el reducido tiempo que tarda la autenticación, ya que los parámetros estarán vigentes únicamente durante esa sesión, lo que impone un nivel de prueba pericial más allá de la que posee un usuario que aplica un ataque común de fuerza bruta al esquema de una contraseña cifrada.

Otro aspecto que definió el tipo de experimento en esta simulación, refiere a que dada su naturaleza, se puede asegurar que de completarse un ataque de fuerza bruta este en algún momento alcanzará la coincidencia, por lo que la protección que contra ellos aplica se vincula con el tiempo que consumirá la búsqueda. Esto quiere decir que la consideración del rendimiento es lo importante con estos peligros, por lo que la elaboración de la prueba apuntó sobre ese tema.

En el Apéndice A se muestra el principal programa de esta simulación, que se construyó con un compilador "gcc"® de Free Software Foundation. Como la versión de la biblioteca de funciones OpenSSL no estaba actualizada y normalizada para el sistema operativo Ubuntu® GNU Linux 18.04.5 LTS que empleaba el desarrollador, se procedió a apoyarse con guiones shell y el comando "md5sum". En el Apéndice B se presenta uno de los códigos de este tipo de guiones que se empleó en esta experimentación.

6.2 Ataque sobre el Canal para Adivinar una Respuesta de dos Posibilidades

El diseño de esta otra prueba consideró la situación donde la consulta solicitada, que deberá ser respondida, es del tipo "verdadero o falso". Luego, se incorporó en la simulación la complicación de desconocer si la función "hash" de la instrumentación solicitada es MD5, *Algoritmo de Hash Seguro 256* (SHA256) o *Algoritmo de Hash Seguro 512* (SHA512). Supondremos que esa selección, así como el encadenamiento de consultas es lo que refuerza una operación que a primeras luce débil probabilísticamente. De forma que de entrada y en modo teórico, se puede calcular la probabilidad de que el atacante acierte. Un modo simple para hacer esto se muestra en (1) y se corresponde con la consideración de eventos independientes:

$$Prob_{acertar} = Prob_{pegar\text{-}la\text{-}respuesta\text{-}binaria} \times Prob_{obtener\text{-}el\text{-}algoritmo\text{-}de\text{-}hash} \quad (1)$$

Un cálculo que se traduce como:

$Prob_{acertar} = ½ \times ⅓ = 0.50 \times 0.33 = 0,16$ (2)

Es decir, que con (2) se deduce que teóricamente hay un 16% de probabilidad de que un atacante tenga éxito y cuantitativamente, ello es una mejora sobre un cálculo inicial del 50% que es lo impuesto por la naturaleza binaria de la pregunta. Ahora bien, si se piensa en encadenar al menos 3 preguntas de ese tipo, dentro de una misma verificación procedimental, por tratarse de eventos independientes, la probabilidad se reduce al 4,1%. Para este experimento de simulación, nuevamente se construyeron pogramas en C y Bash Shell.

Para este experimento de simulación, algunos de los programas empleados se muestran en el Apéndice C, mientras que el Apéndice D expone uno de los programas "bash" de este caso.

7. Resultados de la Simulación

Dado que las simulaciones fueron diseñadas, considerando que debían aproximarse a los recursos que realmente tendría un atacante convencional, ello significa que no pertenece a ninguna institución gubernamental, de ciberguerra o a un equipo bien organizado y dedicado a subvertir sistemas industriales, los resultados fueron simples, se orientaron al modelo de amenazas inicialmente establecido. Pero es necesario señalar que por eso únicamente muestran una primera aproximación, no conforman un amplio protocolo de pruebas de seguridad criptográfica y/o seguridad.

7.1 Evaluación Básica del Experimento 1

Esta simulación se apoyó en el utilitario "*time*" que es propio del *Sistema de Operación* y las mediciones fueron en tandas de 10 ejecuciones para cada orden de dígitos empleado. Los datos experimentales obtenidos se exponen en la Tabla V que seguidamente se expone:

Tabla V: Resultados de la Simulación Número Uno

Orden de dígitos que incluyó el valor pseudoaleatorio	Tiempo que requiere el atacante	Resultado para el atacante
1	3,238 segundos	Éxito
1	0,666 segundos	Éxito
1	0,830 segundos	Éxito
1	3,632 segundos	Éxito
1	3,555 segundos	Éxito
1	0,648 segundos	Éxito
1	3,570 segundos	Éxito
1	0,084 segundos	Éxito
1	2,493 segundos	Éxito
1	2,065 segundos	Éxito
2	El sistema se agotó al transcurrir 10.605 segundos y no consiguió la coincidencia	Fracaso
2	El sistema se agotó al transcurrir 8,738 segundos y no consiguió la coincidencia	Fracaso
2	El sistema se agotó al transcurrir 8,965 segundos y no consiguió la coincidencia	Fracaso
2	El sistema se agotó al transcurrir 8,908 segundos y no consiguió la coincidencia	Fracaso
2	El sistema se agotó al transcurrir 10.257 segundos y no consiguió la coincidencia	Fracaso
2	El sistema se agotó al transcurrir 7,321 segundos y no consiguió la coincidencia	Fracaso
2	El sistema se agotó al transcurrir 7,229 segundos y no consiguió la coincidencia	Fracaso
2	4,139 segundos	Éxito
2	El sistema se agotó al transcurrir 8,610 segundos y no consiguió la coincidencia	Fracaso

Para el caso donde el valor pseudo aleatorio estaba en el orden de dos dígitos, las prueba realizada en un sistema Ubuntu GNU Linux 18.04.5 LTS marcó un límite de archivos abiertos en el guión "shell" para el atacante, con un tope para el cálculo del valor de la serie de 113.000, es decir, con un valor aleatorio de la serie que sea mayor que 113.000 el atacante agota sus recursos en un sistema similar y no llega a la coincidencia. La Figura 1 muestra la pantalla de la ejecución de la simulación con dos dígitos, donde el atacante fracasa con "113.000" archivos abiertos, mientra comprueba el valor de la serie con apenas 3 cifras decimales.

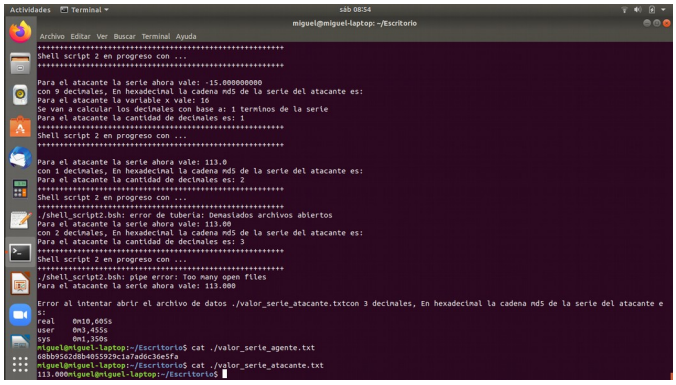

Figura 1: Simulación Número Uno y el Agotamiento de Recursos

Posteriormente, se realizó el mismo experimento pero en otro sistema menos reciente, tratando de ver si eran menos exigentes con ciertos límites, como fue Ubuntu GNU Linux 15.10 y el resultado fue similar con la restricción del tope, aunque se observó que el tiempo consumido estuvo en 6.916 segundos. Se dedujo entonces que la restricción de archivos abiertos señalada es propia del sistema operativo y que puede alterarse con ajustes en el "kernel", pero de esa experiencia ejecutada también fue notable observar que con un valor pseudo aleatorio de un único dígito, se obtuvo un promedio de 2,0781 segundos; luego se pasó a un valor pseudo aleatorio de 2 dígitos, con un tiempo de ejecución promedio de 7,4772 segundos. Ello indica más de 3 veces el valor medio de 1 dígito.

Es valioso señalar además, que para ese caso de 2 dígitos, de la tanda de 10 ejecuciones, solamente una vez el atacante llegó al éxito señalando una demora de 4,139 segundos. La Figura 2 presenta una captura de pantalla, cuando el cálculo está en el orden de 2 dígitos decimales y acontece la única

coincidencia que favorece al atacante. Los valores de la función “hash” coinciden.

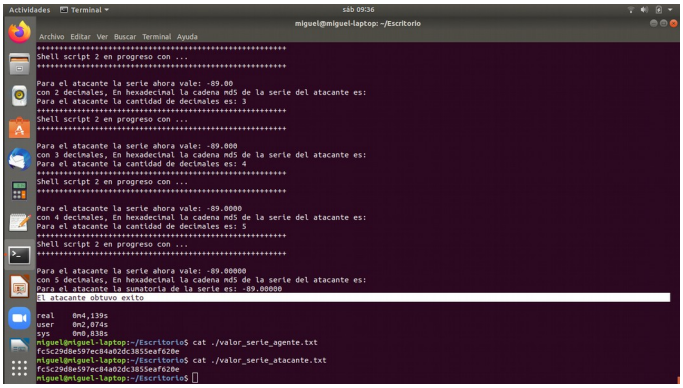

Figura 2: Simulación Número Uno donde el Atacante Triunfa

Para un incidente en la red, aunque el atacante logre su propósito debemos señalar que ese tiempo promedio puede resultar contraproducente para su éxito, si se piensa que en un enlace moderno, el *retardo de transmisión* de una comunicación en red está en el orden de los milisegundos. Esto obliga a experimentar con ataques reales que viajen en la red, ya que hay textos escritos como Estadística Computacional de James E. Gentle, editado en 2002 por la Universidad George Mason, donde se señala que la técnica que con esta simulación se empleó, a razón de mejorar la pseudo aleatoriedad del experimento, que obliga a ejecutar repetidas veces la generación de la semilla, arroja como consecuencia una demora adicional.

Por otra parte, a partir de esos resultados se puede observar un claro patrón de consumo exhaustivo de recursos, lo cual conduce a deducir que con un valor pseudo aleatorio de 5 dígitos, como comúnmente se emplea en muchos generadores de compiladores modernos, resultará más difícil que un atacante logre su meta. De nuevo hay que recordar que a diferencia de los esquemas tradicionales de seguridad, cuando el éxito de un atacante radica en romper con un ataque de fuerza bruta el contenido secreto de una contraseña, en esta propuesta lo que habría logrado es conocer la pregunta que se está haciendo, no la secuencia de todas ellas.

7.2 Evaluación Básica del Experimento 2

Con esta otra experiencia de simulación, los datos experimentales arrojaron los valores que expone la Tabla VI que abajo se presenta:

Tabla VI: Resultados de la Simulación Número Dos

Número de ejecuciones	Probabilidad de acierto del atacante
100	32,00%
1.000	34,00%
10.000	33,81%
100.000	33,35%
1.000.000	33.29%

A partir de esto resultados se puede observar que la probabilidad experimental duplica el valor de la teórica y aunque hacen falta más muestras para poder obtener un valor más confiable, es marcado la presencia de un patrón en ese orden. El autor del trabajo tiene la impresión de que resulta necesario explorar más pruebas con distintos generadores de valores pseudo aleatorios, ya que el cálculo teórico parte de la premisa de que los valores son aleatorios, más en la práctica esa condición no se cumple con exactitud. Para esta simulación es notable que el generador produce valores acortados entre el intervalo cerrado de 0 y 32767, cosa que es marcadamente diferente al cálculo teórico que se basa en un rango infinito de valores. A esto se debe agregar que desde hace tiempo el *Banco de Datos de Vulnerabilidades Comunes y Exposiciones (CVE)* ha emitido alertas donde señala al generador “rand()” como no recomendable para programar aplicaciones criptográficas, ya que este resulta débil ante ataques de fuerza bruta. Un ejemplo claro de ello es la debilidad de código CVE-2009-3278.

Dentro del compilador GNU C, una alternativa inmediata para probar es la función “random(3)”, de producción con re-alimentación no lineal, que el mismo sistema de desarrollo GNU recomienda como alternativa más idónea a “srand()” para aquellos casos cuando la distribución de valores pseudo aleatorios es de fuerte importancia para la aplicación; el autor de este trabajo la descubrió a partir de los resultados de esta simulación.

7.3 Lecciones Derivadas de la Simulación

En función de los dos experimentos realizados se pudo deducir lo siguiente:

- Es necesario y factible de diseñar un conjunto de pruebas por simulación que permitan establecer criterios acerca de la generación de valores pseudo aleatorios, la longitud mínima en los términos de la serie de potencias, el rango adecuado de la cantidad de decimales que deberá emplearse para los resultados, impacto del “hardware” en los ataques de fuerza bruta, los niveles de tolerancia ante respuestas incorrectas y otros parámetros más, de importancia para la instrumentación final del código del sistema. Para el caso de la generación de valores pseudo aleatorios, desde finales de la década de los 80's, se han descrito pruebas sobre la calidad de sus resultados que podrían ser incorporadas en la selección de los generadores como una fase previa a estos otros experimentos.
- Es imperioso experimentar la combinación de preguntas binarias con numéricas enmarcadas en ZKP, como parte de un esquema que refuerce la autenticación con base reto-respuesta. Y es que la fortaleza final de la propuesta radica en cómo se desarrolle todo el proceso, no en una única consulta que un atacante debería acertar.
- Es recomendable desarrollar un protocolo formal de pruebas de seguridad de la propuesta con agentes inteligentes de software que tenga como fase final, descartar las simulaciones y comprobar en redes de computadoras reales, para así incluir con diferentes cargas de tráfico el tema del rendimiento en la red.

8. Conclusiones

Al culminar este trabajo, aunque se tienen muchas nuevas interrogantes a tratar, se concluyó lo siguiente:

- Es factible construir nuevos mecanismos de seguridad que puedan enfrentar los sofisticados ataques de nuestro tiempo e integrarse también con los sistemas ya en uso. Sin grandes recursos económicos y computacionales, es posible trabajar en alternativas de autenticación, que superen la tradicional aproximación basada en una contraseña secreta, todavía de extenso uso pero también frecuentemente sobrepasada.
- Este trabajo describió una propuesta simple para mejorar la seguridad de los mecanismos de autenticación en sistemas críticos de computación, que deben ser automatizados con agentes de software. También se expuso con simulaciones simples el peligro que puede ocasionar tener una implementación pobre, incluso cuando se tenga una buena formulación teórica. En ese sentido es un aporte para ilustrar en la necesidad de profundizar más investigaciones del tema.

Referencias

[1] M. Woolridge. *An Introduction to Multiagent Systems*. John Wiley and Sons, LTD. 2002.

[2] A. Ross. *Security Engineering: A Guide to Building Dependable Distributed Systems.* Wiley Publishing Inc. 2001.

[3] ITU. *Data Communications Networks: Open Systems Interconnection (OSI). Security, Structure and Applications. Security Architecture for Open Systems Interconnection for CCITT Applications. Recommendation X.800*. 1991. Disponible en: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.800-199103-I!!PDF-E&type=items.

[4] J. Dooley. *History of Cryptography and Cryptanalysis. Codes, Ciphers, and their Algorithms***.** History of Computing. Springer International Publishing AG. 2018.

[5] A. Menezes, P. van Oorschot y S. Vanstone. *Handbook of Applied Cryptography*. CRC Press. 1996.

[6] M. Drašar. *Password Based Autentication*. Master Thesis. Universidad de Masaryk. República Checa. 2009. Disponible en: https://is.muni.cz/th/qkn59/thesis.pdf.

[7] M. Raza, M. Iqbal, M. Sharif y W. Haider. *A Survey of Password Attacks and Comparative Analysis on Methods for Secure Authentication*. World Applied Sciences Journal. Vol 19. 439-444. 10.5829/idosi.wasj.2012.19.04.1837.2012.

[8] Saisó y Martínez. *Historia de las Doctrinas Filosóficas*. Prentice Hall. Pearson Educación de México S.A. de C.V. 2009.

[9] Sampieri, R., Fernández, C y María del Pilar, Baptista. *Metodología de la Investigación*. Sexta Edición. McGraw-Hill / Interamericana Editores S.A. de C.V. 2014.

[10] J. Davies. *Implementing SSL/TLS Using Cryptography and PKI*. Wiley Publishing, Inc. 2011.

[11] S. Garfinkel, A. Schwartzand y E. Spafford. *Practical Unix and Internet Security*. Tercera edición. O'Reilly Media, Inc. 2003.

[12] Balasubramanian, K. y K, Mala. *Zero Knowledge Proofs: A Survey. Chapter 9. Algorithmic Strategies for Solving Complex Problems in Cryptography*. IGI Global. 2018.

[13] B. Christianson, B. Crispo, J. Malcolm y M. Roe (Eds). *Security Protocols.* Lectures Notes in Computer Science 2845. Springer. 2002.

[14] N. Fergunson, B. Schneier y K. Tadayoshi. *Ingeniería Criptográfica. Diseño, Principios y Aplicaciones Prácticas*. Wiley Publishing, Inc. 2010.

[15] P. Ceruzzi. *A History of Modern Computing*. Second edition. MIT Press. 2003.

[16] J. Saltzer y M. Schroeder. *The Protection of Information of Computer Systems*. Cuarto Simposio ACM sobre Principios de Sistemas operativos. 1973. Disponible en: https://www.cse.unsw.edu.au/~cs9242/20/papers/Saltzer_Schroeder_75.pdf.

[17] S. Gates. *Modern Defense in Depth. An Integrated Approach to Better Web Application Security*. O'Reilly Media, Inc. 2019.

[18] J. Banks, J. Caron II, B. Nelson y N. David. *Discrete-Event System Simulation*. Quinta edición. Pearson Education, Inc. 2010.

[19] G. Harvey, J. Tobochnik y Ch. Wolfgang. *An Introduction to Computer Simulation Methods Applications to Physical System*. Pearson, Addison-Wesley. 2016.

[20] A. Hartmann. *Practical Guide for Computers Simulations*. World Scientic Publishing Co. Pte. Ltd. 2009.

Apéndice A

CÓDIGO EN C

```
/*
ataque.c - Simulación simple de ataque tipo fuerza bruta para autenticación de MAS

Opera exclusivamente con la función: "1/e = 1 - x + x^2/2! - x^3/3! + x^4/4! - x^5/5! + ... + (-1)^(n-1)x[x^(n-1)/(n-1)!]"
Autor: mtorrealba@>usb.ve
Fecha: 05-08-2020
Versión: 1.0
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>
#define LONGITUD 33 /* 4bits/car. Salida de 128 bits se convierte en 32 car + 1 de NULL */
#define LONG_RESUMEN 16 /* Longitud del bloque de 16 bytes que usará el "hash" */
#define MAXCANTDDEC 10
#define MAX 10                  /* Valor máximo de la variable que el atacante usa para la serie */
#define ARCHIVO_SERIE "./valor_serie_agente.txt"
#define ARCHIVO_ATACANTE "./valor_serie_atacante.txt"

/* ------------------------------------------------------------------------------------ */
/** Función: genera_pseudo()-    Produce un valor aleatorio. **/
int genera_pseudo()
{
  int r;
  srand(time(NULL));   /* Inicialización de la semilla */
  r=rand();           /* Retorna un entero pseudo-aleatorio entre 0 y RAND_MAX */
  fprintf (stdout, "\ngenerando número pseudo aleatorio=%d", r);
  return r;
}

/** Función: factorial() -    Calcula el factorial de un número entero positivo **/
double factorial (int valor) {
  /* Precondicion: valor en un entero positivo con rango limitado */
  if (valor == 0)
                return (1);
  else
                return (valor*factorial(valor-1));
}

/** función compara_hashes () -    devuelve 1 si los hashes coinciden. Si no retorna 0 **/
int compara_hashes ()
{
  FILE *af1, *af2;
  unsigned char cadena_1[LONGITUD], cadena_2[LONGITUD];
  af1=fopen (ARCHIVO_SERIE, "r");
  if (af1==NULL) {
                fprintf (stderr, "\nError al intentar abrir el archivo de datos %s", ARCHIVO_SERIE);
                exit (101);
  }
  af2=fopen (ARCHIVO_ATACANTE, "r");
  if (af2==NULL) {
                fprintf (stderr, "\nError al intentar abrir el archivo de datos %s",
ARCHIVO_ATACANTE);
                exit (101);
  }
  fgets(cadena_1, LONGITUD, af1); fgets(cadena_2, LONGITUD, af2);
  if (strcmp(cadena_1, cadena_2) == 0) return (1);
  else return (0);
}

/** función: corre_agentes() -    Simula la corrida de los agentes que se autentican **/
double corre_agentes()
{
  FILE *apfile;
  int contad, cant_agentes, cant_decimales, variable_x, limite_inf;
  double serie;
  apfile=fopen (ARCHIVO_SERIE, "w+");
  if (apfile==NULL) {
    fprintf (stderr, "\nError al intentar abrir el archivo de datos %s", ARCHIVO_SERIE);
```

```
    exit (1);
  }
  /* Ejecución de los agentes */
  limite_inf=0;  /* Limite inferior del intervalo donde inicia identificación de los agentes */
  serie=1;          /* Suma del del polinomio -empieza con el término independiente -resto- */
  cant_agentes=3;       /* Cada agente es un término de la serie con la variable x */
  cant_decimales=5;  /* Cantidad de decimales que incluirá el cálculo */
  variable_x=genera_pseudo()%10;
  fprintf(stdout, "\nla variable x de los agentes para esta simulación es: %d\n", variable_x);
  for (contad=0; contad < cant_agentes ;contad++)
  {
    /* Si el término es par se resta el nuevo valor, en caso contrario se suma. Pero como
    el contador se inicia en cero entonces se suman los impares */
    if (contad%2 == 0)
      serie=serie-(pow(variable_x, contad+1)/factorial(contad+1));
    else
      serie=serie+(pow(variable_x, contad+1)/factorial(contad+1));
  }
 fprintf(stdout, "\nPara los agentes la sumatoria de la serie es: %.*f y ello se almacenará en %s",
cant_decimales, serie, ARCHIVO_SERIE);
  fprintf(apfile, "%.*f", cant_decimales, serie);
  fclose (apfile);
  return(serie);
}

/** función: corre_atacante() -    Simula la corrida del atacante. Devuelve "1" si el atacante descubre
el valor oculto **/
int corre_atacante()
{
  /* Variables del atacante */
  FILE *arch_atacante;
  int contad_atacante, cant_decimales_atacante, cant_agentes_atacante, variable_x_atacante,
limite_inf_atacante;
  double serie_atacante, serie_atacante_temporal;
  unsigned char cadena_md5_atacante[LONGITUD];
  unsigned char resumen_hash_atacante[LONG_RESUMEN];
  limite_inf_atacante=0;        /* Limite inferior del intervalo donde inicia la identificación */
  for (variable_x_atacante=0; ((compara_hashes() == 0) && (variable_x_atacante < MAX));
variable_x_atacante++)
  {
    cant_agentes_atacante=3;    /* Cada agente es un término de la serie con la variable x */
    serie_atacante=1;           /* Suma del polinomio -empieza con el término independiente -resto- */
    for (contad_atacante=0; (contad_atacante < cant_agentes_atacante) && (compara_hashes() ==
0); contad_atacante++)
    {
      fprintf(stdout, "\nPara el atacante la variable x vale: %d", variable_x_atacante);
      /* Si el término es par se resta el nuevo valor,
       en caso contrario se suma. Pero como el
       contador se inicia en cero entonces se
       suman los impares */
      if (contad_atacante%2 == 0)
        serie_atacante=serie_atacante-(pow(variable_x_atacante,
contad_atacante+1)/factorial(contad_atacante+1));
      else
        serie_atacante=serie_atacante+(pow(variable_x_atacante,
contad_atacante+1)/factorial(contad_atacante+1));
      serie_atacante_temporal=serie_atacante;                /* Se guarda temporalmente el cálculo de
la serie que se está calculando */
    fprintf(stdout, "\nSe van a calcular los decimales con base a: %d terminos de la serie",
contad_atacante);
    /* Se prueba con la cantidad de decimales */
    for (cant_decimales_atacante=1; cant_decimales_atacante < MAXCANTDDEC;
cant_decimales_atacante++)
    {
      /* Se vuelve a abrir para limpiar el archivo */
      arch_atacante=fopen (ARCHIVO_ATACANTE, "w+");
      if (arch_atacante==NULL) {
        fprintf (stderr, "\nError al intentar abrir el archivo de datos %s", ARCHIVO_ATACANTE);
        exit (2);
      }
      fprintf (stdout, "\nPara el atacante la cantidad de decimales es: %d", cant_decimales_atacante);
      fprintf (stdout, "\nPara el atacante la serie ahora vale: %.*f", cant_decimales_atacante,
serie_atacante);
      fprintf(arch_atacante, "%.*f", cant_decimales_atacante, serie_atacante);   /* Aquí se escribe la
respuesta */
      fclose(arch_atacante);
      system ("./shell_script2.bsh");  /* guion que deja en ARCHIVO_ATACANTE el hash */
      fprintf(stdout, "\ncon %d decimales, En hexadecimal la cadena md5 de la serie del atacante es:
%s", cant_decimales_atacante, resumen_hash_atacante);
      if (compara_hashes() == 1)
        break;
      }
      serie_atacante=serie_atacante_temporal;  /* Se restaura el valor de la serie que se está
calculando */
    }
  }
  fprintf(stdout, "\nPara el atacante la sumatoria de la serie es: %.*f", cant_decimales_atacante,
serie_atacante);
  return ((compara_hashes()==1));
}

/* ---------------------------------------------------------------------------- */
int main()
{
  double serie;
  serie=corre_agentes();       /* Ejecuta el cálculo de la serie de los agentes */
  system ("> ./valor_serie_atacante.txt");     /* Crea ARCHIVO_ATACANTE vacio */
  system ("./shell_script1.bsh");                  /* Se deja en ARCHIVO_SERIE el hash */
  fprintf(stdout, "\n**********************La serie %f de los agentes ya se procesó", serie);
  if (corre_atacante() == 1)  /* Ejecuta la corrida de un atacante */
    fprintf(stdout, "\nEl atacante obtuvo exito");
  else
    fprintf(stdout, "\nEl atacante ha agotado sus ejecuciones y aún no ha logrado la coincidencia");
  fprintf (stdout, "\n");
  exit (0);
}
```

APÉNDICE B

CÓDIGO EN BASH

```
#! /bin/bash
#
# shell_script1.bsh v1.0 - Guión Bash para complementar la simulación del experimento 1
# Autor: mtorrealba@usb.ve
# Version. 1.0
# Fecha: 05-08-2020
#
echo ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
echo Shell script 1 en progreso con ...
echo ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

ARCHIVO_SERIE="./valor_serie_agente.txt"
TEMPO="./foobar.txt"
head -n1 `echo $ARCHIVO_SERIE` | md5sum | awk '{print $1}' > `echo $TEMPO`
mv $TEMPO $ARCHIVO_SERIE
```

APÉNDICE C

CÓDIGO EN C

```
/*
simulacion-serie.c - Caso #2 de la Simulación simple para autenticación de MAS

Autor: mtorrealba@usb.ve
Fecha: 05-08-2020
Version: 1.0
Propósito: Simula un ataque sobre una respuesta del tipo binaria con 3 posibilidades de selección de
funciones hash que son MD5, SHA256 y SHA512.
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define ARCHIVO "./pseudo_aleatorio_legal.txt"
#define ARCHIVO2 "./pseudo_aleatorio_ilegal.txt"

/* ---------------------------------------------------------------------- */
/** Función: genera_pseudo()
   Produce un valor aleatorio.
**/
int genera_pseudo()
{
  int r;
  srand(time(NULL));   /* Inicialización de la semilla, debe ser invocada solo una vez */
  r=rand()%10;
  while ((r!=1) && (r!=0))
                r=rand()%10; /* Retorna entero entre 0 y RAND_MAX */
  return r;
}

/* ---------------------------------------------------------------------- */
int main()
{
  FILE *apfile, *apfile2;
  int valor, valor2;
  valor=0;
  valor2=0;
  apfile=fopen (ARCHIVO, "w+");
  if (apfile==NULL) {
    fprintf (stderr, "\nError al abrir el archivo de datos %s", ARCHIVO);
    exit (1);
  }
  apfile2=fopen (ARCHIVO2, "w+");
  if (apfile==NULL) {
    fprintf (stderr, "\nError al abrir el archivo de datos %s", ARCHIVO2);
    exit (-1);
  }
  valor2=genera_pseudo();
  fprintf(apfile2, "%d", valor2);
  fclose (apfile2);
  system("./foobar.bsh");
  exit (0);
}
```

APÉNDICE D

CÓDIGO EN BASH

```
#! /bin/bash
```

```
#
# foobar.bsh v1.0 - Guión Bash para complementar la simulación del experimento 2
# Autor: mtorrealba@usb.ve
# Version. 1.0
# Fecha: 05-08-2020
#
echo ++++++++++++++++++
echo MD5
echo ++++++++++++++++++
md5sum ./pseudo_aleatorio_legal.txt | awk {'print $1'} > pp.asc
SELECCION=`echo $(($RANDOM%3))`
echo la variable selección contiene: $SELECCION
if [ $SELECCION -eq 0 ]
then
  echo ++++++++++      echo MD5
  echo ++++++++++      md5sum ./pseudo_aleatorio_ilegal.txt | awk {'print $1'} > pp2.asc
else
  if [ $SELECCION -eq 1 ]
  then
    echo ++
    echo SHA256
    echo ++
    sha256sum ./pseudo_aleatorio_ilegal.txt | awk {'print $1'} > pp2.asc
  else
    echo ++
    echo SHA512
    echo ++    sha512sum ./pseudo_aleatorio_ilegal.txt | awk {'print $1'} > pp2.asc
  fi
fi

CADENA=`head -1 ./pp.asc`
CADENA2=`head -1 ./pp2.asc`
if [ "$CADENA" = "$CADENA2" ]
then
  echo "." >> ./coincidencias.txt
else
  echo "." >> ./diferencias.txt
fi
```