Universidad de Buenos Aires
Facultad de Ingeniería
Año 2018 - 2º Cuatrimestre

75.26 Simulación

Informe Trabajo Práctico Final:
”Markov Game Modeling of Moving Target Defense for Strategic Detection of Threats in Cloud Networks”

Apellido y nombre	Padrón	Email
Pichinini, Facundo	84906	facundo.pichinini@gmail.com

Índice

Presentación del paper estudiado

Abstract

El procesamiento y almacenamiento de información crítica en grandes sistemas de red en la nube, necesita soluciones de seguridad que sean escalables. Ha sido demostrado que implementar todas las medidas de seguridad posibles implica incurrir en un alto costo de performance en dichos sistemas, utilizando valiosos recursos que los proveedores prefieren destinar para vender sus servicios. En consecuencia, se ha suscitado un reciente interés en desarrollar mecanismos de Moving Target Defense (MTD)(*) que ayuden a optimizar el objetivo conjunto de maximizar la seguridad mientras se asegura que el impacto en el rendimiento del sistema sea mínimo. Frecuentemente, estas técnicas modelan los ataques en múltiples etapas como un juego de detección de ataques de una sola etapa, perdiendo por tanto información valiosa inherente a los modelos teóricos de grafos diseñados para grandes sistemas de redes en la nube, y resultando en ciertas estrategias que tienen impactos asimétricos en la performance. En este trabajo (1) lo que se realizó en cambio, es modelar el problema como un Juego de Markov (*) de suma cero (*), y usar el “Common Vulnerability Scoring System (CVSS)” para obtener como resultado valores representativos de la realidad. Luego, se mostró que la estrategia óptima (*) para colocar mecanismos de detección contra un adversario es equivalente a computar el Equilibrio Min-max mixto de un Juego de Markov (*). Se comparan las ganancias obtenidas con este método a las obtenidas con otras técnicas que se utilizan actualmente en la seguridad de sistemas de red en la nube, demostrando su efectividad. Y por último se remarca como este método fue usado en un pequeño sistema de red de la vida real.

Introducción

Un proveedor de servicios en la nube, provee hardware de almacenamiento y de procesamiento junto con recursos de red a sus clientes, obteniendo por ello un beneficio económico. Elementos como firewalls, Sistemas de Detección de Intrusos (o IDS por sus siglas en inglés), logs de monitoreo, etc. han sido la columna vertebral para detectar y/o detener el tráfico malicioso que intenta acceder a dichos sistemas. Desafortunadamente, la escala de los sistemas de red de hoy en día hace que la colocación de todos los sistemas de seguridad posibles sea inviable, debido al alto costo que insumiría de recursos que se necesitan a su vez para ofrecer servicios monetizables a los clientes. En consecuencia se vuelve relevante la cuestión de cómo colocar una cantidad de sistemas de detección limitada, de forma tal de minimizar el impacto en la performance del sistema a la vez que se asegura que la seguridad del sistema no se reduce drásticamente.

Ya se han realizado trabajos en pos de responder esta pregunta (2), y se ha concluido que la colocación de sistemas de detección estáticos es inevitablemente insegura, dado que los atacantes van a aprender eventualmente donde están colocados, consiguiendo así evitarlos.

En consecuencia, la colocación de sistemas de detección dinámicos se convirtió en el método estándar. Dicha práctica es conocida como Moving Target Defense (MTD) y se puede usar para ir variando los sistemas de detección utilizados de alguna manera aleatoria cada cierto intervalo de tiempo, introduciendo de esta forma un grado de incertidumbre acerca de si un ataque dado va a ser detectado o no.

En los métodos tradicionales, el tratamiento de los ataques en múltiples etapas como ataques independientes e individuales de una sola etapa, lleva a la colocación sub óptima de los mecanismos de detección al no poder detectar el verdadero riesgo de un ataque a largo plazo. Estas estrategias, por ejemplo, pueden priorizar la detección de ataques de alto impacto en un web server, minimizando el riesgo de una sucesión de ataques de bajo impacto que generen un camino de acceso hasta un servidor de almacenamiento de información. Además, pueden llevar a la colocación de múltiples sistemas de detección en un mismo host, llevando a una degradación considerable de su performance.

En este trabajo se trata de subsanar esos problemas modelizando el sistema en la nube como un Markov Game (*):

- Subsistemas de red dentro de la red general, determinados a partir del Attack Graph (AG) que modela al sistema, representan los **estados** posibles del juego.
- Las **acciones del atacante** se corresponden con ataques reales determinados usando las “Common Vulnerabilities and Exploits (CVEs)” halladas en la “National Vulnerability Database (NVD)”.
- Las **acciones del defensor** se corresponden con la colocación de mecanismos de detección que puedan detectar dichos ataques.
- Las **recompensas** se diseñaron en base a la información disponible en el “Common Vulnerability Scoring Sistems (CVSS)”, obteniendo un puntaje para cada vulnerabilidad en función de factores como tipo de nodo atacado (FTP, Web Server, etc), daño potencial del ataque en caso de resultar exitoso, complejidad del ataque (asociada a la probabilidad de éxito en caso de no ser detectado), etc.

Con todo esto, se pudieron diseñar estrategias de defensa que tuviesen en cuenta el impacto a largo plazo de los ataques a la vez que se asegura que el defensor escoja un número limitado de acciones de monitoreo para cada estado del juego, garantizando así una distribución homogénea de los mecanismos de detección en la red que no afecte asimétricamente a los distintos servidores involucrados.

Primer caso de estudio

En el siguiente diagrama, se ilustra un Grafo de Ataque para un escenario que servirá de ejemplo en el análisis del modelo propuesto.

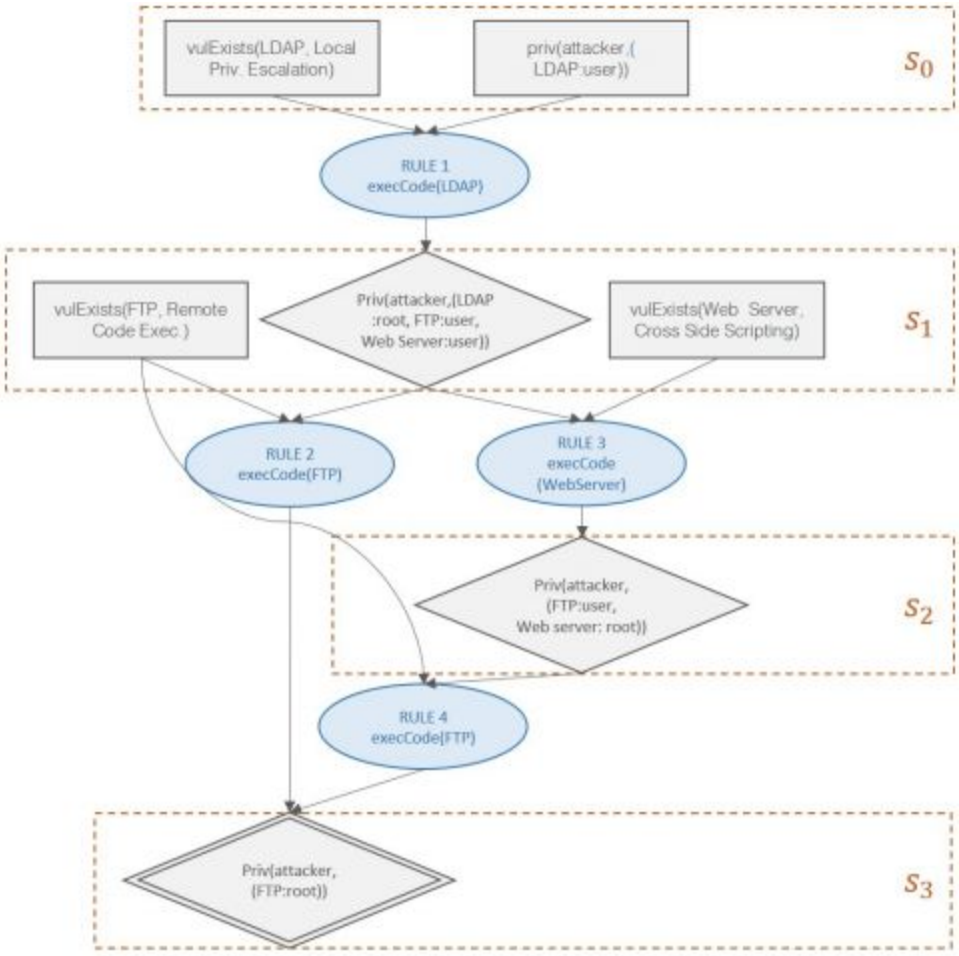

Figura 1

En el mismo se pueden observar encerrados en líneas punteadas los 4 estados posibles del juego, siendo el último aquel en el cual el atacante logró llegar hasta el servidor FTP con información crítica del negocio, significando esto su triunfo y la consecuente derrota del defensor. En las elipses celestes, se encuentran los posibles ataques a efectuar desde cada estado para pasar al siguiente, constituyendo las acciones tanto del atacante (explotar la vulnerabilidad) como del defensor (colocar un mecanismo de defensa que la monitoree).

De esta forma, el atacante tiene dos caminos para llegar a su objetivo (el servidor FTP):

- Camino 1: Explotar vulnerabilidad en LDAP → Explotar vulnerabilidad en FTP

- Camino 2: Explotar vulnerabilidad en LDAP → Explotar vulnerabilidad en Web Server → Explotar vulnerabilidad en FTP

Modelo

Para abordar esta problemática, se optó por modelarla como un Juego de Markov para 2 jugadores, donde uno de los jugadores va a ser quién intenta atacar el sistema, y otro quien intenta defenderlo.

Markov Game para dos jugadores

Para dos jugadores P_1 y P_2, puede definirse como la tupla $(S, A_1, A_2, \tau, R, \gamma)$, donde:

• $S = \{s_1, s_2, s_3, \ldots, s_k\}$ son los posibles estados del juego (finitos),
• $A_1 = \{a_{11}, a_{21}, \ldots, a_{m1}\}$ representa las m posibles acciones de P_1,
• $A_2 = \{a_{12}, a_{22}, \ldots, a_{n2}\}$ representa las n posibles acciones de P_2,
• $\tau\ (s, a_1, a_2, s')$ es la probabilidad de alcanzar el estado $s' \in S$ desde el estado s si P_1 y P_2 toman las acciones a_1 y a_2 respectivamente,
• $R_i\ (s, a_1, a_2)$ es la recompensa que obtiene P_i en el estado s si P_i y P_{-i} toman las acciones a_1 y a_2 respectivamente, y
• $\gamma^i \rightarrow [0, 1)$ es el factor de descuento (*) para el jugador i. En el resto del paper se asume $\forall i\ \gamma_i = \gamma$.

El concepto de **Política Óptima** (*) para el jugador P_i se define como la selección de la acción que optimiza el valor (asociado a la recompensa que se recibe) en cualquiera de los estados posibles, teniendo en cuenta:

- que se está en un contexto de toma de decisiones similar a un Markov Decision Process (MDP) (*) definido por τ
- el espacio de acciones posibles de los otros jugadores, lo cual se hace encontrando una política de min-max sobre el espacio de acciones de cada jugador en cada estado.

En un Juego de Markov cada estado representa un Matrix Game (*), y la política en cada juego busca no solo maximizar la recompensa en este juego sino también en los siguientes.
Como se usa un mecanismo de max-min para tomar la decisión del P_1 (defensor), y para evitar ser anticipado por el atacante (second-guessed), el max player debería jugar una **Mixed Strategy** (tener una distribución de probabilidades sobre las acciones que puede jugar)
La mixed strategy se formaliza definiendo los **Q-valores** para una acción a_1 tomada por P_1 en el estado s, siendo que P_2 elige a_2, como:

$$Q(s, a_1, a_2) = R(s, a_1, a_2) + \gamma \sum_{s'} \tau(s, a_1, a_2, s') \cdot V(s') \quad (1)$$

Estos Q-valores, pueden entenderse como una medida de la calidad de cada acción posible de P_1 en el contexto mencionado.
Definiendo la política mixta (como como se mencionó, se considera la política óptima a emplear) para el estado s como **π(s)**, un vector de longitud m que representa la distribución de probabilidades que P_1 tiene sobre las m acciones disponibles en el estado s, podemos definir el **valor del estado** s para P_1 usando la ecuación:

$$V(s) = \max_{\pi(s)} \min_{a_2} \sum_{a_1} Q(s, a_1, a_2) \cdot \pi_{a_1} \qquad (2)$$

El valor obtenido de esta forma, indica la recompensa total que se espera alcanzar en ese estado mediante la política óptima empleada. Es decir, nos indica que tan bueno es el desempeño del jugador en este estado. Es este por lo tanto, el valor que vamos a utilizar como referencia para evaluar la efectividad de las distintas políticas óptimas que se analizarán en el paper.

Estados

Los distintos estados del juego, se definen como abstracciones dentro del Grafo de Ataque. Si bien el análisis del grafo de ataque como herramienta excede los objetivos del presente trabajo, vale mencionar que dichos estados están formados a partir de los nodos que representan el progreso del atacante, y las vulnerabilidades que puede explotar en cada caso. De esta forma, como se observa en la Figura 1, el estado inicial del juego (S_0) es aquel donde el atacante tiene acceso con privilegio de usuario al servidor LDAP, y tiene conocimiento de un posible ataque que en caso de éxito le permitiría acceder a privilegios de administrador dentro del LDAP, pasado así al siguiente nivel, y “desbloqueando” nuevas vulnerabilidades a explotar. De forma análoga, el estado final (S_3), será aquel en el que el atacante haya accedido a nivel de administrador dentro el servidor FTP y por ende haya “ganado el juego” (accedido a hacer el mayor daño posible).
Las reglas para las abstracciones que definen los estados, son que cada uno de estos nodos puede pertenecer a un único estado, y que todos los nodos de este tipo tienen que estar contenidos en alguno de los estados.

Jugadores y Sets de Acciones

Las acciones para el jugador defensor (P_1), consistirán en colocar sistemas de detección específicos para cada tipo de ataque. Además, en cada estado también tendrá disponible la opción de no realizar ninguna acción (en el caso de que desplegar uno de los mecanismos sea más costoso que la ganancia potencial esperada).
Las acciones para el atacante (P_2), consisten en explotar las vulnerabilidades conocidas, dependientes del estado en el que se encuentra. Al igual que el defensor, tendrá siempre la opción de no realizar ninguna acción, si considera que el riesgo de realizarla es mayor a la ganancia potencial.
Ambos jugadores conocen todas las acciones disponibles tanto propias como de su oponente.

Transiciones

Las transiciones entre estados, a diferencia de las cadenas de markov vistas en clase, dependerán de una serie de factores. A saber, dependerá, de la acción elegida por cada jugador, y en el caso de que el atacante realice el ataque y no sea monitoreado, el mismo tendrá una probabilidad de éxito asociada a la complejidad de efectuarlo, que se obtiene normalizando las métricas presentes en el CVSS para cada ataque. En la imagen a continuación puede observarse el ejemplo gráfico de una transición:

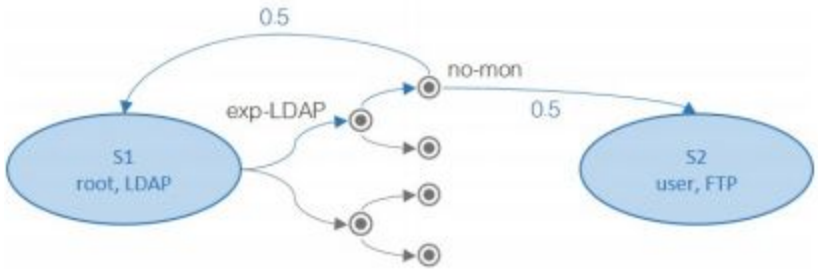

Figura 2

Aquí el atacante elige efectuar un ataque LDAP y el defensor elige no monitorear. La probabilidad de éxito al efectuar este ataque se determinó en 0,5, por lo que el juego pasará al siguiente estado con probabilidad de 0.5 y permanecerá en el estado actual con la probabilidad restante. El conjunto de todas estas probabilidades, se representa por la función de probabilidad τ (s, a_1, a_2, s') definida anteriormente.
Cuando un atacante intente ejecutar un ataque, y el defensor esté monitoreando y logre detectarlo, el atacante volverá al estado inicial. En caso de que no lo detecte, pasará al nivel siguiente en caso de éxito, o permanecerá en el estado actual en caso contrario.

Recompensas

En cada estado, cada par de acciones elegidas por los jugadores van a tener un valor de recompensa para cada uno, pudiendo ser este positivo (ganancia), negativo (pérdida) o cero. Como además se considera a este juego como de suma cero (zero-sum game), el par de valores para cada par de acciones posible será igual en módulo y opuesto en signo. Es decir que lo que gana uno lo pierde el otro. La siguiente tabla muestra las recompensas para este juego:

		P_1 (Defender)		
		no-mon	mon-Web	mon-FTP
P_2 (Atk.)	no-op	$0, 0$	$2, -2$	$3, -3$
	exp-Web	$7, -7$	$-5, 5$	$10, -10$
	exp-FTP	$10, -10$	$10, -10$	$-7, 7$

Table 2: Reward (R_2, R_1) for state s_1

	no-mon	mon-LDAP
no-op	$0, 0$	$3, -3$
exp-LDAP	$5, -5$	$-5, 5$

	no-mon	mon-FTP
no-op	$0, 0$	$2, -2$
exp-FTP	$10, -10$	$-8, 8$

Table 3: Reward (R_2, R_1) for states s_0 (left) and s_2 (right).

Figura 3

Estos valores se calcularon utilizando la Puntuación de Impacto obtenida en el CVE para cada ataque, y el costo de recursos insumido por cada mecanismo de detección a desplegar.
Para el caso del estado final S_3, se adoptó un “valor elevado” de (20,-20) para representar el daño que el atacante puede infligir al acceder a la información del FTP.

Implementación y Resultados

Para la implementación de este modelo, se utilizó en primer lugar el modelo de red ilustrado en la Figura 1 para probar la efectividad del tipo de política planteado como política óptima (mixed strategy), en comparación con otras dos políticas que son comunes en la comunidad de la seguridad informática

Estrategias de comparación

Min-Max Pure Strategy (MMPS)

Se basa en un mecanismo de selección determinístico, que ante cada acción posible del atacante, elige cual es la mejor respuesta posible y la elige con probabilidad 1. En escenarios de la vida real, eso facilita al atacante anticiparse a la colocación de los mecanismos de seguridad, que es la razón por la cual se introduce en este trabajo el concepto de Moving Target Defense (MTD), y se propone la Mixed Strategy como una estrategia superadora. Así y todo, la MMPS es la mejor opción de colocación de defensas estáticas en contextos de recursos limitados, y es mejor de lo que muchos administradores de redes utilizan.

Uniform Random Strategy (URS)

En este caso, el defensor utiliza una distribución de probabilidad uniforme entre todas las acciones disponibles en un estado dado. Si bien intuitivamente puede parecer un método efectivo, se ha demostrado (3) su ineficacia, pero se incluye como medida de comparación para enfatizar las consecuencias que puede tener tomar estas decisiones basado en la intuición.

Resultados

A continuación, se muestran los valores obtenidos con cada método en cada estado a medida que varía el factor de descuento (γ). Para obtenerlos, se realizó una simulación en python, utilizando el solver de optimización Gurobi (4) para el caso de la mixed strategy, y graficando los resultados mediante la librería Matplotlib (5).
A medida que el factor de descuento (*) se incrementa, ambos jugadores comienzan a evaluar qué recompensas futuras tendrán al momento de elegir las acciones, y por ende seleccionan estrategias que le dan mejores resultados a largo plazo. Cuando el factor de descuento es alto, la sucesión de partidas evaluadas se ve más influenciada por la alta recompensa del estado S_3 para el atacante (y penalización para el defensor). Al tratarse este de un estado atractor, a la larga siempre se llegará a caer en dicho estado. Esto explica por qué los valores aumentan en todos los estados al incrementar el factor de descuento.
Por el contrario, cuando el factor de descuento es bajo (el más bajo evaluado es 0,5), las recompensas futuras no tienen un impacto sustancial en los valores del estado evaluado.
En el valor del último estado (S_3), todas las estrategias muestran el mismo resultado puesto que ambos jugadores tienen una única acción posible que tomar.

Recordemos que las tres estrategias evaluadas son la Max-Min Pure Strategy (MMP), la Uniform Random Strategy (URS) y la Mixed Strategy (OPT), que es la propuesta en este paper como Política Óptima, y veamos cómo se comportan dichas estrategias en cada estado graficando el valor de cada estado V(s) al variar el factor de descuento:

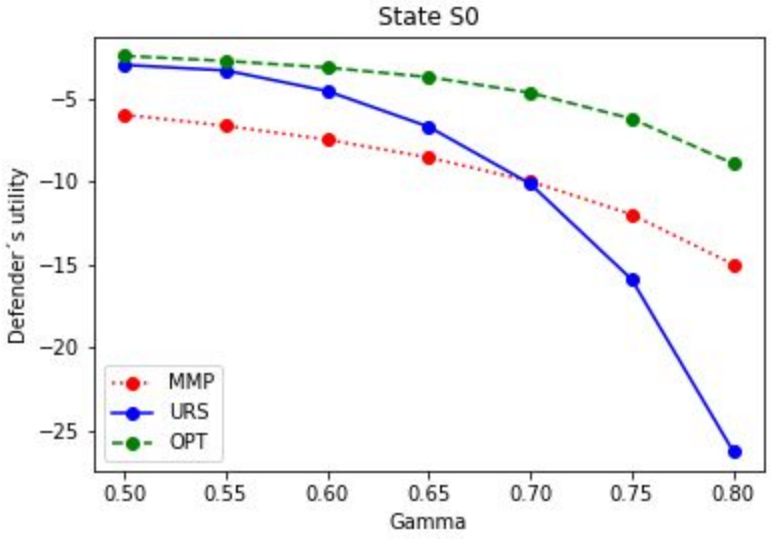

Figura 4

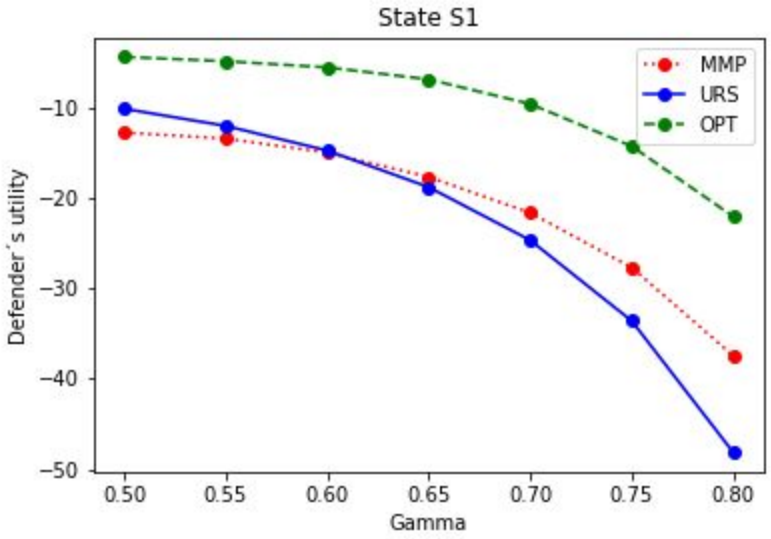

Figura 5

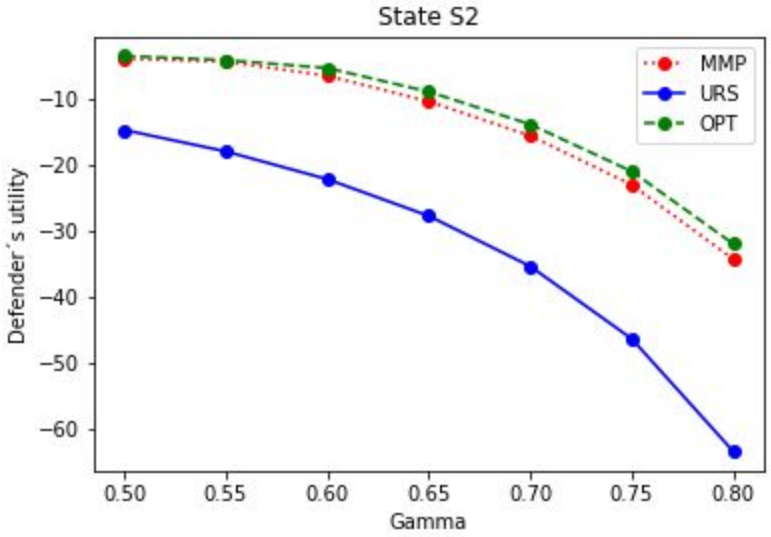

Figura 6

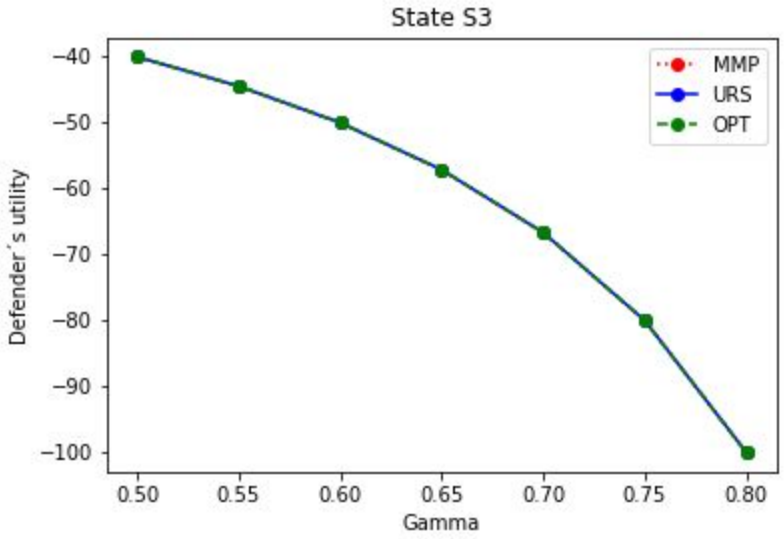

Figura 7

Efectivamente, y de igual forma que en la simulación realizada por los autores del paper, puede observarse que en cada caso la política óptima propuesta, la mixed strategy, es la que mejor se desempeña, y puede también verse que a medida que el factor de descuento aumenta, su performance también aumenta, debido a que es el único método que va ajustando las probabilidades asignadas a cada acción en función de cómo repercutirán en los juegos futuros (lo que en la simulación serían siguientes iteraciones). La determinación de estas probabilidades para este tipo de estrategia se resuelve también mediante la herramienta Gurobi.

Podemos apreciar esto un poco mejor viendo valores obtenidos durante la simulación sobre estas probabilidades. En todos los casos, los valores se obtuvieron al finalizar una partida de 300 iteraciones.

```
---------------
Discount Factor: 0.8
Strategy: MMP
---------------
π(0) : {'no-mon': 0.0, 'mon-LDAP': 1.0}
π(1) : {'no-mon': 0.0, 'mon-Web': 0.0, 'mon-FTP': 1.0}
π(2) : {'no-mon': 0.0, 'mon-FTP': 1.0}
π(3) : {'lost': 1.0}
```

Para la MMP, la acción a tomar en cada estado es siempre la misma, elegida deterministicamente mediante una Pure Strategy (*), independientemente del factor de descuento.

```
---------------
```

```
Discount Factor: 0.8
Strategy: URS
---------------
π(0) : {'no-mon': 0.5, 'mon-LDAP': 0.5}
π(1) : {'no-mon': 0.3333, 'mon-Web': 0.3333, 'mon-FTP': 0.3333}
π(2) : {'no-mon': 0.5, 'mon-FTP': 0.5}
π(3) : {'lost': 1.0}
```

En el caso de la URS, la probabilidad de tomar cada acción se distribuye uniformemente entre todas las disponibles en cada estado, independientemente del factor de descuento.

```
---------------
Discount Factor: 0.5
Strategy: OPT
---------------
pi(0) : {'no-mon': 0.5905, 'mon-LDAP': 0.4094}
pi(1) : {'no-mon': 0.1234, 'mon-Web': 0.4423, 'mon-FTP': 0.4341}
pi(2) : {'no-mon': 0.1067, 'mon-FTP': 0.8932}
pi(3) : {'lost': 1.0}

---------------
Discount Factor: 0.8
Strategy: OPT
---------------
π(0) : {'no-mon': 0.4044, 'mon-LDAP': 0.5955}
π(1) : {'no-mon': 0.0, 'mon-Web': 0.5465, 'mon-FTP': 0.4534}
π(2) : {'no-mon': 0.0, 'mon-FTP': 1.0}
π(3) : {'lost': 1.0}
```

Para la Mixed Strategy, la cual le asigna probabilidades a cada acción dependiendo de la conveniencia de realizarla, puede observarse que al variar el factor de descuento, su comportamiento también varía. Con un factor de descuento bajo (0,5), se centra más en los sucesos cercanos, y la posibilidad de no realizar acciones en los primeros 3 estados, aunque baja, recibe una probabilidad de ser elegida (el riesgo de perder aún es lejano y se ahorra los recursos que implica desplegar los mecanismos de detección). Por el contrario, con un factor de descuento más cercano a 1 (0,8), toma más peso el riesgo de perder a largo plazo, por lo que no realizar ninguna acción en los estados 1 y 2 se descarta como opción.

Caso de estudio de la vida real

A continuación, en el paper analizan un “caso de la vida real”, utilizando Virtual Machines de la competición del 2018 de la Western Region Cybersecurity Defense Competition (WRCCDC) (6). En esta competencia, había dos equipos, el rojo (atacantes) y el azul (defensores). El objetivo del equipo azul, era el de mantener el sistema funcionando a la vez que se detectaban y registraban los ataques del equipo rojo.

El esquema de la red en cuestión es el siguiente:

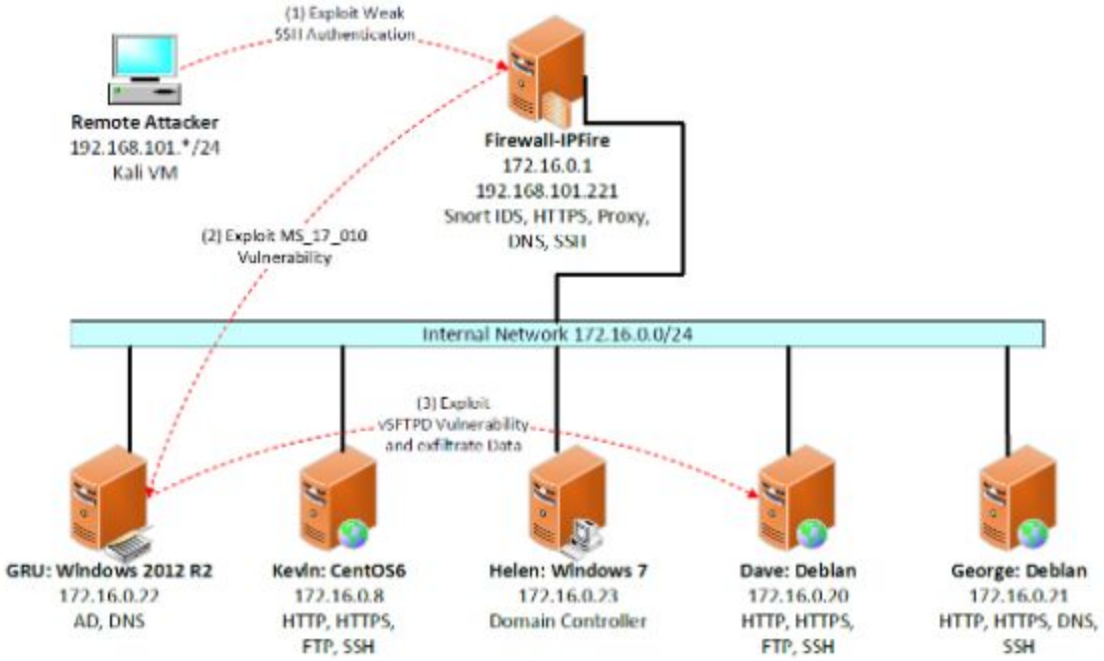

Figura 8

Donde las flechas rojas representan un posible camino de un ataque en múltiples pasos, para impactar en uno de los servidores de la capa final donde funciona un FTP. Se busca modelar y evaluar la efectividad de los métodos planteados para detectar y prevenir la ejecución de este ataque.

Para representar este caso, se realizó una simulación con el mismo código que la anterior, pero adecuando los parámetros para representar este modelo, y se comparó el desempeño de una una URS con el de la Política Óptima propuesta en el paper. Los resultados obtenidos son idénticos a los del paper:

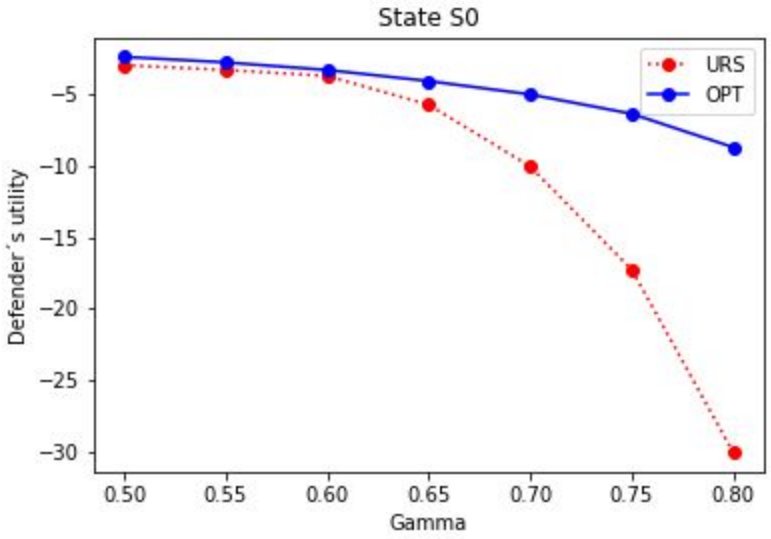

Figura 9

Estado inicial del sistema, que corresponde al atacante aún fuera de la red.

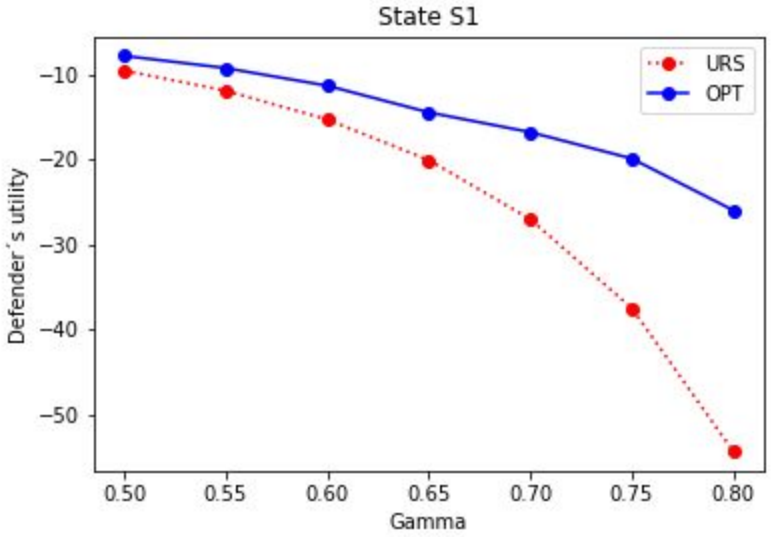

Figura 10

Siguiente estado, donde el atacante consiguió acceso al server donde funciona el firewall (destino de la primer flecha roja)

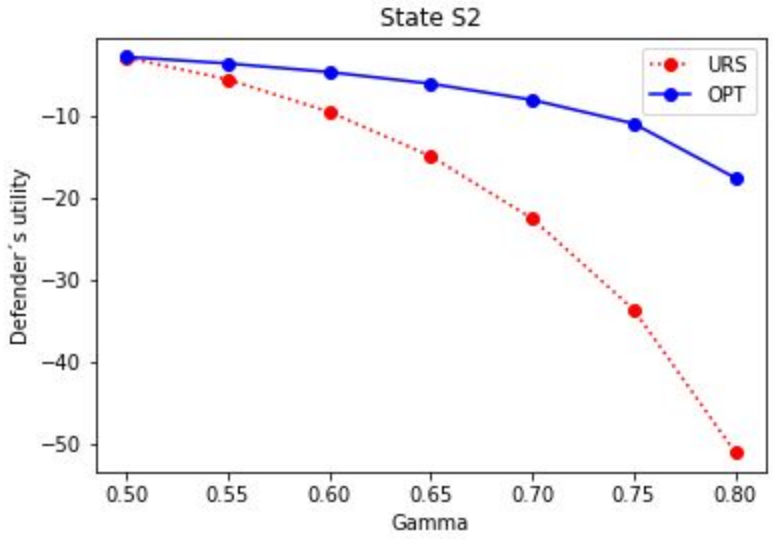

Figura 11

En el estado S2, el atacante logró acceder a la Internal Network, y se aloja en el servidor con Windows 2012.

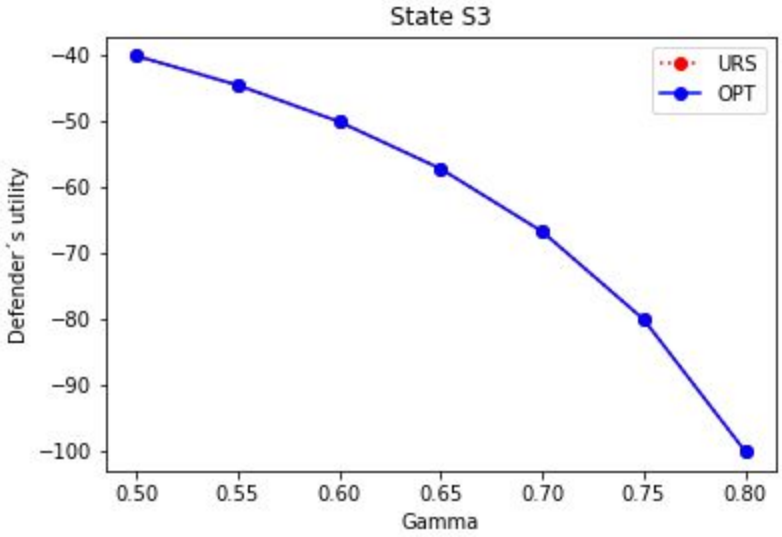

Figura 12

Estado final del juego, donde el atacante logró su objetivo de acceder al servidor Debian con el FTP. Nuevamente, al ser un estado final y haber una única acción disponible para ambos jugadores, el resultado de ambas estrategias es idéntico.

Al igual que en el caso anterior, la Mixed Strategy propuesta como política óptima demuestra ser más efectiva para mitigar las pérdidas por el daño causado por el atacante y/o por la pérdida de performance de colocar medidas de protección innecesarias.

Discusión

El paper demuestra que ante un escenario donde se conocen las posibles vulnerabilidades del sistema, y se tienen recursos limitados para la colocación de medidas de detección, modelar el problema de la colocación de las mismas como un Juego de Markov, y utilizar una Moving Target Defense con una Mixed Strategy como política óptima es la mejor herramienta para mitigar las pérdidas causadas por el atacante y/o por la pérdida de performance de colocar medidas de protección innecesarias. Con esta herramienta, se puede construir una cadena de Markov donde sabiendo el estado actual del juego podemos ir eligiendo en cada instante de tiempo qué mecanismos colocar para influenciar las probabilidades de que el sistema evolucione hacia alguno de los posibles estados deseados.
En ambos casos estudiados, la cadena de Markov analizada es no ergódica, ya que posee un estado final de carácter absorbente y estados intermedios transitorios. Debido a esto y a que en dicho estado final el defensor sufre una penalización elevada, su recompensa al final de cada juego es negativa, pero consigue disminuir considerablemente su pérdida con el método propuesto.
Si bien el concepto de Juego de Markov para 2 jugadores amplía la noción de Cadena de Markov vista en clase, el concepto de evolución de estados es en esencia el mismo, pudiendo apreciarse la efectividad y versatilidad de esta herramienta para modelar situaciones de la vida real. En particular, fue interesante descubrir cómo puede utilizarse en técnicas de Reinforced Learning, y lo relevante que es en el mundo de la Teoría de Juegos.
Por último, si bien el modelo evaluado es pequeño y limitado a una única acción por atacante y una única acción por defensor, los resultados son contundentes. Al evaluar el riesgo a futuro, consigue prevenir más eficazmente los ataques de múltiples etapas que otros mecanismos que se utilizan como norma en la actualidad. Los investigadores mencionan en el paper que esperan poder estudiar este método en modelos más complejos a pesar del costo computacional que implica esta evaluación en modelos de mayor envergadura.

Conclusión

Se pudieron reproducir los resultados mostrados en el paper de manera satisfactoria y utilizando las mismas herramientas para modelarlo. El repositorio original (7) al igual que el realizado en este análisis (8) pueden encontrarse en la sección de referencias.

Glosario

Markov Decision Process (MDP)

Un Proceso de Decisión Markoviano (En inglés Markov Decision Process), es un modelo estocástico estocástico de toma de decisiones a intervalos discretos de tiempo, donde un agente toma decisiones y donde el resultado de sus acciones es aleatorio y solo depende del estado actual del sistema y la decisión tomada.
Básicamente, en cada intervalo de tiempo el proceso está en un estado s, donde el agente puede elegir una acción a que esté disponible en este estado. Elegida la acción, el sistema responde en el siguiente intervalo de tiempo moviéndose aleatoriamente a un nuevo estado s' y otorgándole al agente la recompensa correspondiente.

Juego de Markov

Es un juego que puede ser modelado como una cadena de Markov, es decir, donde la toma de decisiones en un estado dado no depende de las decisiones que se han tomado antes, y cuyo mecanismo de decisión es un MDP.

Optimal Policy

La política de decisión en un MDP, denotada normalmente como $\pi(s)$, es aquella estrategia que le indica al agente que decisión tomar en cada estado en el que se encuentra. Se considera óptima cuando esta política le hace tomar la mejor opción posible en cada estado.

Min-Max Policy

El funcionamiento de minimax puede resumirse en cómo elegir el mejor movimiento para ti mismo suponiendo que tu contrincante escogerá el peor para ti.
El esquema a continuación ejemplifica el proceso de toma de decisión con una política Min-Max

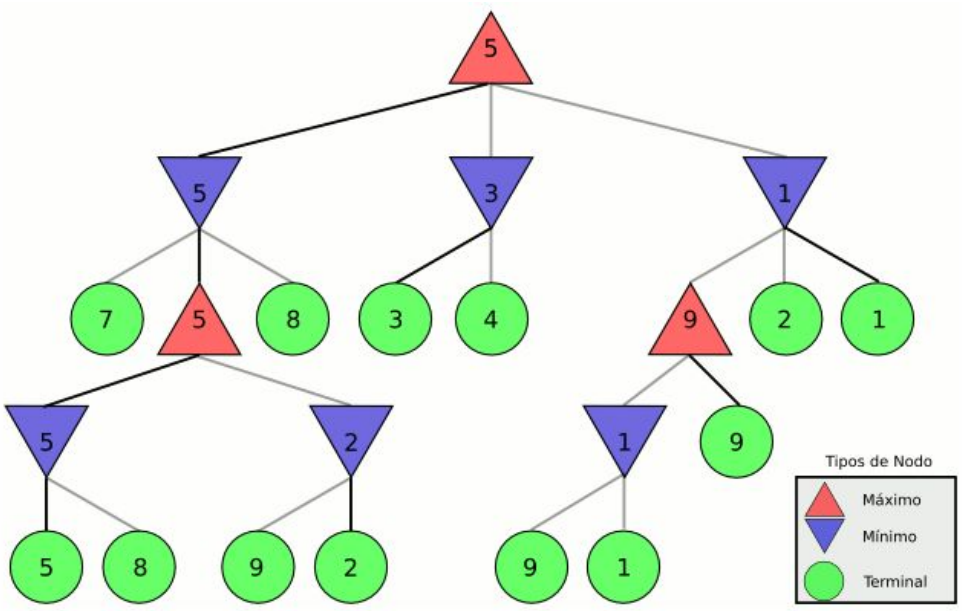

Figura 13

En este caso, un jugador (en color azul) asume el rol de Minimizar, y ante cada elección posible selecciona la de menor recompensa. El otro jugador (color rojo) asume el rol de Maximizar, y escoge en cada caso la mayor recompensa.

Pure Strategy

Consiste en evaluar de manera determinística cual es la mejor acción a tomar en cada caso (estado), o lo que es equivalente, asignarle una probabilidad de 1 a una acción en cada estado. Combinada con la Min-Max Policy descrita anteriormente, implicaría fijar cuál es la acción que me conviene tomar en cada estado asumiendo que mi oponente va a elegir la peor para mi.
El problema de esta estrategia en la vida real, para el caso estudiado, es que le permite al atacante anticiparse a nuestra respuesta, haciéndole perder efectividad.

Mixed Strategy

Consiste en definir una distribución de probabilidades sobre el vector de acciones a elegir por el jugador en cada estado. Al evaluar las recompensas que una acción me puede traer a futuro en lugar de únicamente la recompensa inmediata, tiene sentido otorgarle a cada acción una probabilidad de elección acorde a su posible utilidad futura. Además, de esta forma me evito ser anticipado por mi oponente.

Matrix Game

Un juego de matriz (en inglés Matrix Game) es aquel donde las acciones de los jugadores pueden representarse en una matriz, de manera que cada tupla de acciones elegida por los

jugadores pueda mapearse a un casillero de la matriz, de donde se obtiene la recompensa que dicha jugada le otorga a cada uno.
En el caso del Markov Game de dos jugadores planteado en este paper, la matriz de recompensas de cada estado (Figura 3) configura distintos Matrix Game.

Zero-Sum Game

Un juego de suma cero es aquel en donde la suma de recompensas para todos los jugadores da cero para cada jugada. En el caso del modelo presentado en este paper, significa que para cada par de acciones de atacante y defensor, la recompensa para ambos es igual en módulo y opuesta en signo. Esto puede apreciarse observando las tablas de recompensas en la Figura 3

Discount Factor

El factor de descuento es un recurso matemático con el que se obliga a converger a un valor a una sumatoria infinita. En nuestro caso, si bien las sumatorias son finitas (tanto los estados como las acciones son finitos), es lo que regula cuanto voy a tener en cuenta las siguientes jugadas a la hora de elegir mi acción. Un factor de 1 implica valorar de igual forma las recompensas futuras que las actuales, mientras que un valor cercano a 0 implica desestimar el impacto de mis decisiones a futuro.

Moving Target Defense

Consiste en un método de colocación de mecanismos de detección donde se van variando los sistemas de detección utilizados de manera aleatoria cada cierto intervalo de tiempo, introduciendo de esta forma un grado de incertidumbre acerca de si un ataque dado va a ser detectado o no. Caso contrario, de permanecer fijos, el atacante podría aprender a reconocerlos y evitarlos.

Referencias

(1) Chowdhary, Ankur & Sengupta, Sailik & Huang, Dijiang & Kambhampati, Subbarao. (2018). Markov Game Modeling of Moving Target Defense for Strategic Detection of Threats in Cloud Networks. https://www.researchgate.net/publication/329945354_Markov_Game_Modeling_of_Moving_Target_Defense_for_Strategic_Detection_of_Threats_in_Cloud_Networks

(2) [Venkatesan et al. 2016] Venkatesan, S.; Albanese, M.; Cybenko, G.; and Jajodia, S. 2016. A moving target defense approach to disrupting stealthy botnets. In Proceedings of the 2016 ACM Workshop on Moving Target Defense, 37–46. ACM. https://doi.org/10.1145/2995272.2995280

(3) Sengupta, Sailik & Gautam Vadlamudi, Satya & Kambhampati, Subbarao & Doupé, Adam & Zhao, Ziming & Taguinod, Marthony & Ahn, Gail-Joon. (2017). A Game Theoretic Approach to Strategy Generation for Moving Target Defense in Web Applications. https://www.researchgate.net/publication/317170288_A_Game_Theoretic_Approach_to_Strategy_Generation_for_Moving_Target_Defense_in_Web_Applications
(4) Gurobi - Mathematical Optimization Solver https://www.gurobi.com/
(5) Matplotlib https://matplotlib.org/
(6) Western Region Cybersecurity Defense Competition (WRCCDC) https://www.wrccdc.org/
(7) Repositorio original del paper https://github.com/sailik1991/MarkovGameSolvers
(8) Repositorio de este trabajo https://github.com/facundo-p/Markov-Game-Model