## Review 1: Penetration Testing: Practical Introduction & Tutorials

### Details

* + Penetration Testing: Practical Introduction & Tutorials
  + Stephen Watts
  + [Splunk Learn](https://www.splunk.com/en_us/blog/learn/penetration-testing.html?utm_campaign=google_amer_en_search_generic_dynamic_audienceonly_gpa&utm_source=google&utm_medium=cpc&utm_content=dynamic_search&utm_term=&_bk=&_bt=657063425256&_bm=&_bn=g&_bg=149493693980&device=c&gclid=Cj0KCQjw06-oBhC6ARIsAGuzdw1iNrREuL_84d-VV0bnUTjYHCeli2GqMKHgkk0AVcMDsK6rGTFcbgAaAsKgEALw_wcB)

### Introduction

* + S

### Summary

* + A

### Methodologies

* + A

### Main Findings

* + A

### Relevance to Your Course Content

* + A

## Review 2: Autonomous Security Analysis and Penetration Testing

### Details

* + Autonomous Security Analysis and Penetration Testing
  + Ankur Chowdhary, Dijiang Huang, Jayasurya Sevalur Mahendran, Daniel Romo, Yuli Deng, Abdulhakim Sabur
  + <https://ieeexplore.ieee.org/document/9394285>
  + local

### Introduction

* + Current network pentesting techniques involve a combination of automated scanning tools and manual exploitation of security issues to identify possible threats in a network. The solution scales poorly on a large network.
  + Pentesting Difficulties the paper does not address
    - Advanced Persistent Threats (APTs) are hard to detect with pentesting alone
    - Moving Target Defense
      * target environment has a robust adaptive defense mechanism where the network environment adapts to the attacker’s actions

### Summary

* + ASAP
    - autonomous security analysis and penetration testing framework (ASAP)
    - does two important things.
      * figures out how different vulnerabilities in a network are connected to each other and how they relate to the network's structure and complexity.
        + While there has been some research regarding the incorporation of RL in pentesting, they are focus on smaller networks and do “ not use the vulnerability information in the network, and they do not build a correlation between those vulnerabilities and the transition probabilities like this paper”

Vulnerability Information:

did not take into account the specific vulnerabilities present in the network.

relied on known sources, scans, or manual analysis to identify them

ASAP framework integrates vulnerability information into its decision-making process by

analyzing how vulnerabilities are related to each other and how they might be exploited based on the network's structure and connectivity

associates RL reward values with Common Vulnerability Scoring System (CVSS) scores so the framework can prioritize the exploitation of more critical vulnerabilities

considers how easy or difficult it is to move between different network elements and exploit vulnerabilities to help the agent decide the likelihood of success for different actions

Building Correlation

prior research did not establish a correlation between vulnerabilities and transition probabilities

didn't link the presence of vulnerabilities to the likelihood of an attack successfully transitioning through various states in the network

* + - * uses a system of rewards that are based on how serious these vulnerabilities are. Essentially, this system helps their AI understand how real-world security issues work.
    - creates a map of security threats and possible attack paths in the network using attack graphs.
      * “Does this by..”
      * generates autonomous attack plans and validates them against real-world networks
    - framework scales well on a large network
  + Author goal:
    - Address limitation of traditional AI models in the penetration testing domain
    - Unable to understand a network's intricacies
      * don't effectively account for the complex/dynamic nature of real-world networks
      * may not consider the specific network structure / distribution of vulnerabilities
    - Unable to “capture …real-world vulnerability distribution”
      * may not prioritize most critical vulnerabilities
    - can result in less accurate / efficient pt strategies
    - “The attack plans help in reducing the overhead of pentest significantly - order of seconds compared to manual pentest carried out over several hours/days.”

### Methodologies

* + RL
    - “This paper utilizes an artificial intelligence algorithm based on reinforcement learning to identify the attack path that maximizes the reward value for the pentester”
    - “The policy obtained from reinforcement learning formulation is the attack plan for the pentest.”
    - Mathematical Representation
    - *The research adopts reinforcement learning-based artificial intelligence algorithm to identify the optimal attack path that maximizes the reward for penetration testers. Reinforcement learning is a concept where an agent learns from interactions with an environment through trial-and-error. In the context of security assessments, it reflects the process of security professionals trying various attack strategies against vulnerabilities until they successfully exploit one. The RL approach consists of statistical techniques and dynamic programming that consider several key elements throughout the process including current user privilege level, actions corresponding to vulnerability exploitation, difficulty and probability of an action being successful, reward values, and the strategy the agent uses to determine the next action based on the current state. The resulting attack path, or progression of attacks to achieve the goal, is used as the “attack plan for the pentest [which]* *guides the security professional what next action to take” based on (??)' [needs work but good start]*
  + Framework
    - the actions or decisions made by the AI are influenced by considering both the severity of the vulnerabilities (how critical they are) and the difficulty associated with exploiting those vulnerabilities.
    - AI system is designed to prioritize actions that address the most critical and exploitable vulnerabilities in a given network setup.
    - ensures process focuses on the most significant security risks first, helping to strengthen the overall security of the network.
  + Domain-specific modeling
    - Authors use Common Vulnerability Scoring System (CVSS)
      * Used for tracking known vulnerabilities
      * Tracks severity of the vulnerability
      * Tracks how easy or difficult it is to exploit it, ranging from 0 to 10.
        + Access Complexity (AC)
    - *The authors' approach emphasizes Domain-Specific Modeling, which involves creating a customized representation of the network's security environment. This modeling approach focuses on integrating specific network details, including vulnerability information, access complexities, and vulnerability severity. To accomplish this, they utilize the Common Vulnerability Scoring System (CVSS), which not only tracks known vulnerabilities but also assesses their severity and the ease or difficulty of exploiting them, measured on a scale from 0 to 10. Access Complexity (AC) further refines this, categorized as LOW, MEDIUM, or HIGH, with corresponding probability values of 0.9, 0.6, and 0.3.*
    - *To visualize the “most optimal penetration testing policy for the underlying network,” they implement a state-transition diagram to represent the threat model, offering a visual representation of different network states and potential actions. In each state, there are various possible actions that a pentester can take. These actions are associated with probability values, which are determined based on the access complexity (AC) of the vulnerability. Additionally, a goal state is defined, and it is associated with a reward value, which corresponds to the CVSS score of the vulnerability.*
    - The authors then take this state-transition diagram, with its associated probabilities and rewards, and feed it into their DQN model.
    - By aligning the RL framework with the network's unique security landscape … the (unique/customized/tailored) series of actions that a pentester can follow to maximize the accuracy, effectiveness, and efficiency of their test.
  + Deep-Q Network (DQN) based RL model
    - [relate back to Author goal]
      * *To address the limitations of traditional AI models in the PT domain, the authors propose using a Deep-Q Network (DQN) based reinforcement learning model that can better handle high-dimensional state spaces and adapt to previously unseen network states. Additionally, they mention using domain-specific modeling to ensure that the reward distribution and transition probabilities are more aligned with the actual network's structure and the real-world distribution of vulnerabilities. This approach aims to enhance the effectiveness of AI-driven penetration testing in complex network environments.*
    - Evidence that Q-learning based solutions (in RL) for finding optimal attack plans may not work well with “a large scale network with multiple services and vulnerabilities”
      * So they used DQ, which “utilizes a deep neural network to parameterize the Q-learning function,” to “make policies learned more generalizable to highdimensional state space and unseen states.”
  + Attack Graphs
    - They take information about the availability/accessibility of network services (e.g., open ports, protocols) and vulnerabilities within those services to create a visual representation of potential attack paths and relationships between different elements of a network “and dependencies between the vulnerabilities”
    - They use host logs to model to represent the different states a network can be in and the actions that an attacker can take from those states in order to show how an attack could progress through various stages.
      * This representation helps the AI agent generate optimal attack plans that are then tested and validated by actually executing them on a real-world network
    - “The attack graph is parsed to obtain the parameters required for the RL framework.”
    - “The attack plans help in reducing the overhead of pentest significantly - order of seconds compared to manual pentest carried out over several hours/days.” [repeat]
    - help the network administrator to analyze the ‘situation’
  + Implementation
    - Vulnerability Scanning:
      * vulnerability scanners, Nessus and Openvas, to scan the vulnerabilities
    - Attack Graph Generation:
      * scan information, host configuration, and network topology is passed to the attack graph generator
      * attack graph generator generates an attack graph using MulVAL
    - State Graph Creation:
      * Key Information is converted from the attack graph into a structured format (state graph) for the RL algorithm for further analysis
        + which parts of the network give people certain powers, like 'execCode' or 'netAccess.' And how these different privileges are connected in the network
        + if they find a point in the attack graph where an attack is successful and connected to a vulnerability, they extract the CVSS and the AC information
    - Analyze Connections:
      * important points they found in the network are connected and labeled with the extracted information (CVSS and the AC)
      * this state graph serves as a visual representation for how these key points in the network are connected and what vulnerabilities are associated with them .
        + used to calculate transition probabilities and reward values in the reinforcement learning model
        + helps the AI system understand how to navigate the network efficiently and make optimal decisions during penetration testing
    - Attack Plan Generation
      * “The parameters of the state graph help” fill in the key elements of the RL approach mentioned earlier, including state of user privilege, actions, transition probability, reward values, and the agents decision policy.
    - Reward Matrix Population:
      * fill based on attributes of primary targets (i.e gaining specific levels of access or control within the network) within the state graph.
      * The reward value corresponds to the CVSS score of the vulnerability that led to a specific transition in order to reflect the severity and potential impact of the vulnerability
      * gives higher rewards to targets that are harder to achieve because they require exploiting more severe vulnerabilities. This encourages the AI system to focus on challenging but important security goals during penetration testing.
    - Verify Exploit
      * using logs from the ELK server to check if their exploitation attempts worked
      * also discovers any previously unknown vulnerabilities
      * combine this information with the data from state graph and provide it to the RL plan generator
    - Validation with Metasploit:
      * Once the attack plan is generated, it undergoes validation using a Python wrapper for the Metasploit framework.
      * This validation checks the effectiveness of the penetration test.
    - Relay Results
      * pentest results are used to suggest remediation plans for the target organization
      * If patches are deployed/changes made to the network, the attack graph can be updated, and the system can be retested
      * This process forms a complete, self-sustaining cycle for evaluating security and making improvements
        + Very important since PT is not a one-and-done thing
    - Case Study
      * created a simulated enterprise network that included an industrial control system and IoT devices
      * network consisted of 16 hosts distributed across three networks
      * used a mix of Windows and Linux systems for their experiments
      * included four key services: SSH, FTP, HTTP, and SMTP.
      * Goal: for the pentesting team to compromise email information, which required exploiting vulnerabilities on the SMTP service and infiltrating the IoT subsystem through a vulnerability present in the gateway machine.
      * varied the discount factor from γ = 0.6 to γ = 0.99.
        + future rewards compared to immediate rewards

A discount factor, usually denoted as γ (gamma), is a number between 0 and 1.

A higher γ (close to 1) makes the agent prioritize long-term rewards and consider the consequences of its actions in the distant future.

A lower γ (close to 0) makes the agent focus more on immediate rewards and be less concerned about long-term consequences.

* + - * batch size:Varied the amount of experiences/interactions the AI system uses at once to learn and improve its policy (how many examples or trials it looks at in each learning step)

### Main Findings

* + discount factor
    - had a significant impact on how quickly agent learned/made decisions
      * DF of 0.8 (γ = 0.8) - agent learned and converged to optimal solutions more quickly/ agent became proficient at making good decisions faster.
      * DF of 0.9 and 0.99 (γ = 0.9, γ = 0.99) - agent took more time to learn and make decisions. It spent more time exploring potential future outcomes.
      * The reward value diminishes considerably for higher values of γ for all batch sizes.
    - when using a discount factor (γ) of 0.8, the AI system achieved the best results in terms of reward values
  + batch size:
    - using a batch size of 16 (BS=16) helped the AI system learn faster/achieve better results in terms of convergence time and reward values
      * Convergence Time: This refers to how quickly the DQN algorithm (used for decision-making) reached a stable and effective solution during the experiments.
    - with larger batch sizes like 32 or 64, the AI's performance was not as good.
      * note: this observation might not hold true for very large networks due to the increased complexity and scale of such networks which can lead to different dynamics in the learning process for the AI system.
    - Different Strategy
      * AI's strategy for penetration testing is different from manual testing - as prioritizing the exploitation of certain vulnerabilities before others was a better approach for AI
        + sets apart from manual methods that might have a more rigid strategies
      * AI: acknowledging the characteristics of vulnerabilities and their relative difficulty.
        + sometimes starting with less challenging vulnerabilities can lead to a more efficient and effective overall penetration test.
        + data-driven
        + adapting to the unique network environment / vulnerabilities, which

### Relevance to Your Course Content

* + A

## MISC additional Information

* *AI difficulties [R2]*
  + not all vulnerabilities can be treated equally
    - “Some security configuration related issues may be relatively easier to exploit, whereas security problems related to native components such as memory overflow require domain expertise and effort to exploit.” [2]
  + Not all AI solutions scale well
    - Evidence that POMDP based solutions do not scale well on a large networks [2]
    - don't effectively account for the complex/dynamic nature of real-world networks[2]
    - may not understand the network's intricacies[2]
    - may not consider the specific network structure / distribution of vulnerabilities [2]
  + can result in less accurate / efficient pt strategies
  + may not prioritize most critical vulnerabilities or
  + may not understand the network's intricacies

## Ref

1. Penetration Testing: Practical Introduction & Tutorials
2. Autonomous Security Analysis and Penetration Testing

* *S*

In a large system – an in-depth analysis of each device on the system plus finding vulnerabilities for each device and then mapping each vulnerability to not only their exploit, but the likelihood of success for that exploit becomes exponentially time consuming

The reward and transition probability input for existing AI models used in the pentesting domain does not capture the network structure and real-world vulnerability distribution.

practice of creating models that are tailored to the specific characteristics, challenges, and requirements of pentesting

professionals utilize several variants of attack payloads against a vulnerability before identifying a valid proof of concepts

to showcase that it can be exploited. The reinforcement learning problem can be solved using statistical techniques and dynamic programming to estimate the utilities for taking actions in state space. Formally, reinforcement learning can be

defined using tuple (*S, A,R, τ,* Π).

* + - * + S=*{s*1*, s*2*, s*3*, .., sn}* represents the system states. The
* states in our model represent the privilege of the pentester,
* e.g., *s*1 = (userm ftp), *s*2 = (root, ftp), which means
* attacker had user privilege on FTP service in state *s*1,
* and root privilege in state *s*2.
  + - * + A=*{a*1*, a*2*, ..., ak}* represent the action taken in current
* state. We utilize vulnerability exploitation as actions in
* our model. For instance attacker can transition from
* (user, Web Server) to (root, Web Server) by taking action
* *a*1 = exploit (CVE-2010-3947), which corresponds to a
* vulnerability present of FTP service.
  + - * + *τ* (*s*1*, a*1*, s*2) represent the state transition probability. By
* taking action *a*1 in state *s*1 the pentester can transition to
* *s*2 with a certain probability, depending on how difficult
* it is to exploit a certain vulnerability.
  + - * + *R*(*s*1*, a*1) is the immediate reward obtained by an agent
* for taking action *a*1 in state *s*1. We utilize the Common
* Vulnerability Scoring System (CVSS) [27] to assign
* reward value for (state, action) pairs in our reinforcement
* learning model.
  + - * + Π known as a policy which is a strategy employed by
* the agent to learn next action based on current state. We
* employ a well-known model-free reinforcement learning
* algorithm Q-learning [16] to learn the agent’s policy.
* The policy obtained from reinforcement learning formulation
* is the attack plan for the pentest.