Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.2, Apr. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.2.485

동적 네트워크 환경에 적용 가능한 Attack Graph 모델 연구

문 주 연,[1†] 김 태 규,[2] 김 인 성,[2] 김 휘 강[1‡]
[1]고려대학교 정보보호대학원, [2]LIG넥스원

An Attack Graph Model for Dynamic Network Environment

Joo Yeon Moon,[1†] Taekyu Kim,[2] Insung Kim,[2] Huy Kang Kim[1‡]
[1]Graduate School of Information Security, Korea University, [2]LIG Nex1

요 약

시스템 및 네트워크 환경의 규모가 확대되고, 네트워크 구조 및 시스템 구성이 빈번하게 변화함에 따라 네트워크 관리자가 현황을 수동으로 관리하고 실시간 변동사항을 식별하는 데에 많은 어려움이 발생하고 있다. 본 논문에서는 동적인 네트워크 정보를 실시간으로 스캔하고, 사전에 수집한 취약점 정보를 바탕으로 네트워크 내 장치의 취약성 정도를 점수화하고 최종적으로 공격자의 입장에서 공격 가능한 모든 경로를 도출하여 네트워크 관리자에게 공격 가능성이 높은 경로 목록을 제공하는 알고리즘을 제안하였다. 또한 제안하는 알고리즘을 토대로 한 Attack Graph를 실제로 구현하였으며, Software Defined Networking (SDN) 환경이 포함된 동적으로 변화하는 가상 네트워크 환경을 구축한 후 시뮬레이션을 진행하여 Moving Target Defense (MTD) 개념이 반영된 시스템에도 적용이 가능함을 입증하였다.

ABSTRACT

As the size of the system and network environment grows and the network structure and the system configuration change frequently, network administrators have difficulty managing the status manually and identifying real-time changes. In this paper, we suggest a system that scans dynamic network information in real time, scores vulnerability of network devices, generates all potential attack paths, and visualizes them using attack graph. We implemented the proposed algorithm based attack graph; and we demonstrated that it can be applicable in MTD concept based defense system by simulating on dynamic virtual network environment with SDN.

Keywords: Attack Graph, Network Topology, Dynamic Network, Moving Target Defense, Network Security

I. 서 론

특수한 목적을 가지고 하나의 타겟에 대해 지속적으로 정보를 수집하고 취약점을 파악하여 이를 바탕으로 피해를 끼치는 공격을 Advanced Persistent Threat (APT) 공격이라고 한다. APT 공격은 정치적 또는 금전적 목적을 지닌 경우가 많고 주로 기업, 공공 기관 등을 타겟으로 하기 때문에 공격이 성공할 경우 피해 규모가 매우 크며, 탐지하기가 어려워 지속적인 공격이 가능하기 때문에 보안 관리자가 가장 경계해야 하는 공격 유형 중 하나이다[1]. 이러한 공격은 주로 시스템 내의 취약한 단말을 경유하여 확산된다. 즉, 중요한 자산이 포함된 장비가 공격자로부터 일차적으로 보호되어 있다고 하더라도 네트

Received(12. 27. 2017), Modified(04. 02. 2018), Accepted(04. 02. 2018)
† 주저자, jymoon94@korea.ac.kr
‡ 교신저자, cenda@korea.ac.kr(Corresponding author)

워크 시스템 내의 다른 장비를 통해 해당 장비에 접근 가능하다면 자산은 안전하다고 할 수 없다. 이 때 공격자로부터 자산을 보호하기 위해서는 공격자가 침투할 수 있는 경로를 파악하여 미리 예방하는 것이 중요하다. 네트워크 관리자가 공격자의 공격 경로를 미리 예상할 수 있다면 적절한 대응을 통해 침투 확률을 줄일 수 있다.

최근의 네트워크 시스템은 규모가 크고 복잡하여 관리자라 할지라도 네트워크의 토폴로지 및 구성 정보를 파악하고, 그 내부에서 발생할 수 있는 공격 경로를 수동으로 도출하는 데에는 많은 어려움이 따른다. 또한 무수히 많은 공격 경로를 모두 파악하였다고 하더라도 모든 경로에 대해 대응하는 것은 불가능에 가깝다. 네트워크 관리자는 유한한 시간과 자원 내에서 가장 효율적인 대응 방안을 탐색해야 한다. 공격자가 특정 경로로 자산에 침투할 때에, 경로에 포함되는 장치들이 얼마나 쉽게 위험에 노출되는지에 따라 공격 성공 확률 및 난이도가 달라진다. 공격자에 따라 공격 경로를 선정하는 방법이 다르지만, 일반적으로 공격 성공 확률이 높거나 공격 난이도가 낮은 경로일수록 침투 가능성이 높다고 볼 수 있다. 네트워크 관리자가 수많은 공격 경로 중 가장 침투 가능성이 높은 경로를 알 수 있다면 이를 차단하여 빠르고 효율적인 대응이 가능하게 된다. 이러한 공격 난이도는 공격자에 따라 상이한, 주관적인 요소이므로 이를 일반화하기 위해 주로 시스템이 취약할수록 공격 난이도가 높다고 판단하며, 시스템의 취약 정도는 Common Vulnerabilities and Exposures (CVE) 등 기존에 알려진 취약점 정보 또는 Common Vulnerability Scoring System (CVSS) 등 전문가에 의해 정량화된 수치를 이용해 추산할 수 있다.

Attack graph는 이러한 상황에서 공격자의 입장에서 공격 가능한 경로를 시각화하여 관리자가 잠재적인 공격 경로를 파악할 수 있도록 한다. Attack graph를 생성하기 위해서는 네트워크 시스템 전반적인 구조와 시스템 내 각 장치의 구성 정보 (운영 체제, 열린 포트 정보 등) 및 공격할 때에 악용될 수 있는 취약점 등의 정보 수집이 선행되어야 한다. 관리자가 모든 정보를 수동으로 입력하지 않으려면 자동으로 네트워크 내 장치를 스캔하여 정보를 수집하는 과정이 필요하다. 또한 네트워크의 구조 및 장치의 구성 정보는 가변적이므로 주기적인 스캔을 진행하여 최신의 정보로 갱신할 수 있어야 한다.

특히 최근에는 MTD가 사이버 보안 기술의 핵심 전략으로 부상하고 있다. 기존의 보안 전략은 시스템 자체의 구성에는 변화가 없기 때문에 공격자가 한 번 획득한 정보를 다음 공격에 재사용할 수 있다는 약점이 있다. MTD는 이를 보완하기 위해 제안된 보안 전략으로, 주기적으로 IP 주소 등 시스템 구성 정보를 변경하여 공격의 난이도를 증가시킴으로서 보안성을 높인다. 공격 방어 방법으로 MTD 전략을 취하는 시스템에도 적용하기 위해서는 시스템 구성 정보 변경 사항을 빠르게 업데이트하는 것이 매우 중요하다. 이러한 경우 변경 사항을 관리자가 모두 파악하고 있는 것은 현실적이지 않고, 따라서 네트워크 토폴로지를 주기적으로, 효율적으로 스캔하여 변경사항을 스스로 탐지할 수 있어야 한다.

본 논문에서는 attack graph를 통해 가능한 모든 경로를 제시하고, 알려진 취약점 정보를 기반으로 각 경로로 얼마나 쉽게 침투할 수 있는지 계산하여 공격자의 입장에서 가장 쉬운 공격 경로, 즉 가장 침투당할 확률이 높은 경로를 제시한다. 네트워크 관리자는 이를 통해 가장 먼저 대처해야 하는 경로를 알 수 있게 되고, 따라서 효율적이고 신속한 대응이 가능하게 된다. 또한 주기적으로 네트워크 시스템을 스캔하여 네트워크 토폴로지 및 시스템 구성 정보에 변경 사항이 발생할 경우 자동으로 업데이트하여, 네트워크 환경의 변화에도 빠르게 대처할 수 있다. 본 연구에서 네트워크 관리자에게 제공하는 attack graph는 호스트 기반의 방향 그래프 (directed graph)로, 각 노드는 네트워크 시스템 내에 존재하는 장치 중 하나를, 엣지는 시작 노드에서 타겟 노드로의 공격을 의미한다. 따라서 네트워크 관리자가 어느 장치의 보안성을 높이는 것이 효율적인지 직관적으로 파악할 수 있도록 한다.

2장에서는 attack graph와 MTD 등에 대한 관련 문헌 연구를 기술하고, 3장에서는 제안하는 시스템의 알고리즘 및 구현 방법을 설명한다. 4장에서는 3장에서 기술한 방법론을 동적으로 변화하는 네트워크 환경에 시뮬레이션하고, 제안한 시스템이 실제로 네트워크 환경의 변경 사항을 탐지하여 반영할 수 있다는 것을 입증한다. 마지막으로 5장의 결론에서 제안한 알고리즘 요약하고, 이점을 제시한다.

II. 관련 문헌 연구

본 연구에서는 attack graph를 생성하여 공격

경로를 도출하며, 도출한 경로를 관리자에게 시각화할 때에도 attack graph를 사용하기 때문에 attack graph에 관한 기존 문헌 연구를 진행하였다. 더불어, 동적으로 변화하는 네트워크 환경에서 변경 사항을 효율적으로 탐지하여야하므로 네트워크 토폴로지에 관한 문헌 연구도 진행하였다. Attack graph의 생성에 관한 연구는 기존의 문헌들에서 어떤 방법으로 attack graph를 생성 방안 및 시스템 취약성 평가 기준을 조사하였다. 또한 네트워크 토폴로지에 관한 연구는 SDN 환경 또는 동적 특성을 지닌 네트워크 환경에서의 연구를 중점적으로 조사하였다. Table. 1은 조사한 기존 attack graph 관련 문헌들의 attack graph 유형, 점수 산정 기준, 시뮬레이션 환경을 정리한 표이다.

2.1 Attack graph 생성에 관한 연구

네트워크 자원이 방화벽으로 보호되어 있다고 해도 중요한 자산에 도달할 수 있는 다른 호스트로 접근이 가능하다면, 그 호스트에 존재하는 취약점을 이용하여 네트워크 자원을 목표로 한 공격이 일어날 수 있다. 따라서 attack graph를 통해 모든 공격 가능한 경로를 확인하고 대응할 수 있는 방안이 필요하다. Jin B. Hong 등은 attack graph의 표현 방식은 다양하지만 일반적으로 노드와 엣지로 구성되며, 노드는 호스트, 권한, 취약점, 익스플로잇(exploit) 등을 나타낼 수 있고, 엣지는 노드 사이의 전이를 나타낸다고 기술하였다[2]. 최근에는 노드가 익스플로잇을 나타내는 유형의 attack graph에 대한 연구가 활발히 진행되고 있다.

Nirnay Ghos 등은 attack graph를 통해 네트워크 보안의 강도를 평가하는 방식을 제안했다[3]. 해당 논문에서는 네트워크의 토폴로지 (topology)와 네트워크 간 정책, 공격의 시작점과 목표지점, 취약점 정보를 기반으로 한 전제조건과 사후조건, 영향력에 대한 정보를 획득한 후 휴리스틱 알고리즘을 이용해 attack graph를 도출하였다. 제시한 방식의 한계점은 전문가들이 위험을 완화시킬 수 있는 비용 대비 효율적인 방법을 휴리스틱하게 제시하기 때문에 전문가들의 도움이 꼭 있어야 한다는 것이다.

이러한 attack graph 생성 시에 사용되는 취약점은 대부분이 알려진 취약점이며, 제로데이

Table 1. Related paper (Attack graph)

Title	Attack graph	Scoring	Simulation
NetSecuritas: An Integrated Attack Graph-based Security Assessment Tool for Enterprise Networks	Node: exploit, condition Edge: requirement of exploit	CVSS score, number of vulnerabilities (CVE)	Virtual network environment
k-Zero Day Safety: Measuring the Security Risk of Networks against Unknown Attacks	Node: exploit, precondition Edge: relation between exploit and precondition	Number of zero-day vulnerability	Virtual network environment
Automated Attack Path Enumeration Method based on System Vulnerabilities Analysis	Node: host Edge: (potential) attack	CVSS score, number of vulnerabilities (CVE)	Web Portal System of Korea University
Hybrid Attack Path Enumeration System based on Reputation Scores			
Validating and Restoring Defense in Depth Using Attack Graphs	Node: host Edge: (potential) attack	Vulnerability existence (CVE)	Virtual network environment
Evaluating and strengthening enterprise network security using attack graphs	Node: host Edge: (potential) attack	Vulnerability existence, open ports	Virtual network environment
Managing attack graph complexity through visual hierarchical aggregation	Node: condition Edge: exploit	x	Virtual network environment

(zero-day) 취약점은 대외적으로 알려지지 않은 특성으로 인해 정량적인 지표로 나타내기 어려워 이용에 많은 제약이 있다. 이러한 문제를 해결하기 위해 Lingyu Wang는 새로운 지표인 k-제로데이 안전성 (k-zero day safety)을 제안했다[4]. 본 논문에서는 알려지지 않은 취약점들의 랭크를 매기거나 발생 가능성을 측정하기보다, 주어진 네트워크 환경에서 목표 자산에 도달하기까지 필요한 제로데이 취약점이 몇 개인지 계산하여 가장 개수가 적은 경로를 선정하는 데에 초점을 맞추었다.

Ji Hong Kim, Young Hoon Moon 등은 자동화된 취약점 평가 및 공격 침투 경로 예측 시스템을 제안하였다[5][6]. 제안한 시스템은 attack graph를 생성하고, 운영 체제 취약점, 정보자산별 취약점, 사용자 인증을 고려하여 가장 공격 가능성이 높은 경로를 도출하여 네트워크 관리자에게 제공한다. 하지만 네트워크 및 시스템 정보를 관리자가 직접 입력해야 한다는 한계점이 있다.

Attack graph 생성 시 고려해야 할 사항 중 하나는 확장성 문제이다. 최근의 네트워크에 대해서 모든 공격 경로를 포함하는 attack graph를 생성하는 것은 비현실적이다. R. P. Lippmann 등은 처음 지정된 시작 위치에서 공격자가 손상시킬 수 있는 모든 호스트를 결정하며 취약점 제거 시 그 효과를 정확하게 예측할 수 있다는 특성을 지니는 predictive attack graph 생성 의사코드를 제안하였다[7][8].

또한 현재의 네트워크 규모가 크고 복잡한 만큼, 일반적으로 네트워크 attack graph를 시각화 할 때에도 확장성 문제가 큰 부분을 차지한다. Steven Noel 등은 익스플로잇을 엣지로, 조건을 노드로 하는 attack graph 표현 방법에 몇 가지 규칙을 적용하여 계층적인 구조의 시각화 방법을 제안하였다[9]. 해당 논문에서는 attack graph를 간략화하고 사용자가 필요에 따라 이를 해체할 수 있도록 하였다. 따라서 논문에서 제안한 그래프 표현 방법을 이용하면 사용자가 목적에 따라 부분적으로 자세히, 또는 전체적으로 간단하게 네트워크를 파악할 수 있도록 하였다.

2.2 네트워크 토폴로지에 관한 연구

네트워크 관리 및 성능 평가 등을 위해 네트워크 토폴로지의 정보를 얻는 것은 필수적이지만 최신 네트워크는 규모가 크고 복잡하며, 동적 특성을 지니고 있어 수동으로 정보를 알아내는 것이 어렵다. 네트워크 토폴로지는 논리적 토폴로지와 물리적 토폴로지로 나눌 수 있는데 논리적 토폴로지에 비해 물리적 토폴로지의 정보를 알아내는 방법 및 기술은 발달되지 않았다. Jing Jiang 등은 SNMP와 MIB를 기반으로 하는 물리적 토폴로지 디스커버리 방법을 제시하였다[10]. Simon Enoch Yusuf 등은 정적 네트워크만을 고려한 기존의 그래픽 보안 모델과 달리 동적 네트워크에서 시스템 패치 적용, 토폴로지 변화 등 네트워크의 구성이 변경되면 네트워크 변화를 포착하고, 사이버 보안 메트릭(공격 비용, 공격 경로 등)에 이를 반영하는 새로운 그래픽 보안 모델을 개발하였다[11]. Mengmeng Ge 등은 시스템에 보안 패치를 적용할 때에 가용성을 위해 사용하는 이중화(Redundancy)가 시스템의 보안성을 저하시킨다고 하며, 가용성과 보안성의 균형을 찾는 모델 설계 방안을 제시하였다[12].

일반 네트워크 환경뿐만 아니라 SDN 환경에서도 중앙 컨트롤러의 역할을 제공하기 위해 네트워크 토폴로지를 찾는 일이 선행되어야 한다[13]. 따라서 SDN 환경에서 효율적인 네트워크 토폴로지를 찾는 연구 또한 활발히 진행되고 있다. Suleman Khan 등은 현재 주요 SDN 컨트롤러 프레임워크에서 구현된 토폴로지 탐색 방법에서 발생하는 오버헤드에 대한 분석을 진행하였다. 이어서 검색 기능은 동일하지만 SDN 스위치와 컨트롤러에서 처리하는 컨트롤 메세지의 수를 줄여 오버헤드를 줄이는 방법을 제안하였다[14]. 또한 네트워크 애뮬레이터를 사용해 실험을 진행하여 제안한 네트워크 토폴로지 탐색 접근법이 기존 방식에 비해 오버헤드가 적게 발생하며 토폴로지가 복잡한 경우에도 효율적이라는 것을 입증하였다.

III. 방법론

본 논문에서 제안하는 시스템의 프레임워크는 Fig. 1과 같다. Attack graph 도출에 활용하기 위해 알려진 시스템 취약점 데이터베이스를 구축하고, 주어진 환경의 시스템 및 네트워크 정보를 수집한다. 수집한 정보에서 시스템 별 운영 체제 취약점, 정보자산별 취약점, 취약한 포트 정보를 도출하고, 이를 이용해 시스템 취약성 점수를 산정한다. 네트워크 환경은 주기적으로 스캔하여 변화가 발생한 경우

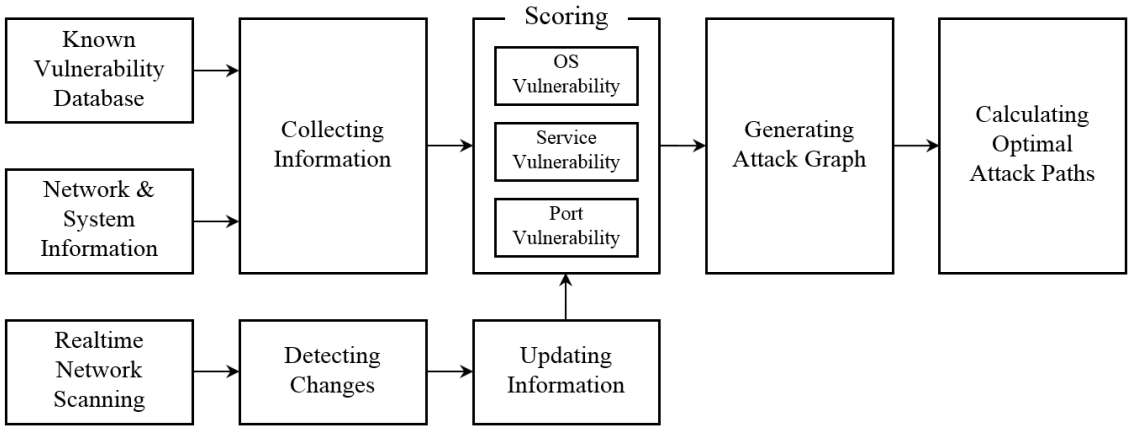


Fig. 1. System overview (information collecting, host scoring, attack graph generating)

시스템 정보를 업데이트한다. 마지막으로 가능한 공격 경로를 모두 포함하는 attack graph를 생성하고, 시스템 취약성 점수를 바탕으로 각 공격 경로의 점수를 계산하여 최적의 공격 경로를 탐색한다.

3.1 알려진 취약점 데이터베이스 구축

3.1.1 Common Vulnerabilities and Exposures (CVE)

CVE는 공격 취약점 관련 표준으로 미국의 비영리 단체인 MITRE에서 1999년 처음 만들어 운영하기 시작했고, 이후 미국 국립표준기술연구소(NIST)가 국가 취약성 데이터베이스 (NVD)를 만들어 구축하면서 체계화됐다[15]. CVE 식별 값은 CVE라는 문자에 취약점이 발견된 연도와 부여한 고유 값으로 구성되어 있고, 이를 이용하여 공개적으로 알려진 취약점을 구분할 수 있다. 본 연구에서 CVE는 장치가 얼마나 취약한지 판단 기준을 수립할 때에 사용된다.

3.1.2 Common Platform Enumeration (CPE)

CPE는 소프트웨어 응용 프로그램, 운영 체제 및 하드웨어 플랫폼의 이름을 지정하는 표준화된 방법이다. CPE는 part, vendor, product, version, update, edition, language 7개의 필드로 구성되어 있다. 본 연구에서 CPE는 사용 중인 프로그램의 제품명(product)과 관련된 CVE 정보를 검색할 때에 사용된다.

3.1.3 Common Vulnerability Scoring System (CVSS)

CVSS는 여러 조직 및 하드웨어/소프트웨어 플랫폼 환경에서 각 취약점의 취약 정도를 점수화한 오픈 프레임워크이다[16,17]. CVSS 점수는 exploitability와 impact 등 몇 가지 metric에 따라 산정된다. 점수의 범위는 0에서 10까지이며 10에 가까울수록 큰 리스크를 갖는 취약점이다.

본 연구에서는 CVSS 값을 취약점 위험도 점수 산정에 활용하고 있으며, 현재 NVD의 취약점 데이터베이스에서는 CVSS 값을 v2와 v3 두 버전으로 제공하고 있다. CVSS v3의 경우 보다 최신 기준으로 산출된 위험도 점수이나, 최신 취약점에 대해서만 한정적으로 산출되어 있고 2014년 이전 취약점 등 전체 데이터베이스에서 보았을 때 대다수의 취약점에 부여되어 있지 않은 실정이다. 아직까지 많은 기업에서 2014년 이전의 장비를 사용하고 있기 때문에, 본 연구에서 제안하는 모델에서는 CVSS v2 값을 기준으로 취약점의 위험도 점수를 사용하였다.

3.1.4 취약점 데이터베이스 구성 정보

NVD에서 제공하는 CVE 데이터 셋을 이용하여 각 CVE 별로 관련된 Common Weakness Enumeration (CWE), CVSS v2 점수, CPE 값, 발행 날짜, 마지막 수정 날짜, 요약, 참조 등 CVE의 기본 정보를 한 테이블에 저장하였다. 또한 CVE 식별값 별로 관련된 CPE 값과 CPE 값을 통해 얻을 수 있는 vendor, product, version, part 등의 CPE 정보를 한 테이블에 저장하였다.

두 테이블은 CVE 식별값을 통해 연동된 상태이며, 따라서 네트워크 스캔을 통해 수집한 운영 체제 및 응용 프로그램의 정보를 이용해 CVSS 점수를 검색하여 해당 시스템의 취약성 점수를 평가할 수 있다. 테이블의 컬럼 정보와 연동 상태는 Fig. 2와 같다. 네트워크 스캔을 통해 운영 체제, 실행 중인 프로그램의 제품명(product) 및 버전을 수집한 후 CPE 테이블에서 관련된 CVE 식별값을 검색한다. 이후 CVE 테이블에서 해당 CVE 식별값으로 상세 정보를 도출할 수 있다.

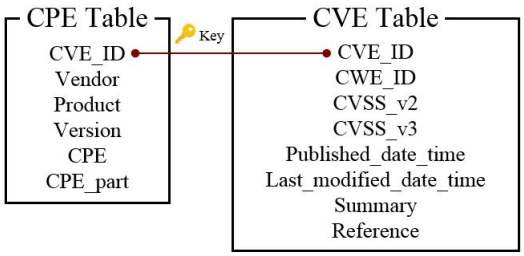


Fig. 2. CPE table and CVE table

3.2 시스템 정보 수집 자동화

3.2.1 일반 기업 환경

3.2.1.1 네트워크 존(Network Zone) 및 스캐닝 포인트 (Scanning Point)

한 스캐닝 포인트에서 네트워크상의 모든 장치를 스캔하기 위해서는 스캐닝 포인트에서 모든 장치에 접근할 수 있도록 보안 정책을 설정해야 한다. 하지만 이러한 보안 정책은 스캐닝 포인트가 공격당한 경우 시스템 전체가 매우 취약해지는 치명적인 약점이 존재한다. 본 연구에서 제안하는 네트워크 스캐닝 방법은 네트워크를 여러 개의 존으로 나누어 각 존 별로 스캔하는 것이다. 따라서 시스템 내 네트워크 정보 수집을 위해 우선 이러한 존 정보를 파악해야 한다. 하나의 존 내 호스트들은 IP 주소 대역과 네트워크 규칙을 공유하며, 존 내 호스트들끼리의 스캐닝은 모두 허용되어 있다고 가정한다. 각 존에는 하나 이상의 분산 관리 서버, 즉 스캐닝 포인트가 존재한다. 스캐닝 포인트란 자신과 동일한 네트워크 대역에 존재하는 시스템, 장비 등의 정보를 탐색하는 역할을 하며, 가급적 컴퓨팅 성능이 좋은 시스템을 선택하는 것을 권장한다. 중앙 관리 서버는 각 스캐닝 포인트에 존 별 스캔을 요청할 수 있으며, 존 별로 수집된 정보는 중앙 관리자용 데이터베이스에 전송되어 중앙 관리자가 정보를 이용할 수 있다. 수집된 정보는 각 스캐닝 포인트의 로컬에도 저장되어 존 내에서 발생하는 시스템 정보 변경 사항을 관리할 수 있다.

3.2.1.2 중앙 관리자

관리 대상이 되는 시스템에는 해당 네트워크 시스템에 대해 잘 이해하고 있는 중앙 관리자가 필요하다. 관리자는 각 호스트가 어느 존에 해당하는지 알고 있어야 하며, 스캐닝 포인트를 지정하여 네트워크 시스템 내 모든 호스트가 스캐닝 포인트의 스캔 영역 안에 포함되도록 해야 한다. 또한 네트워크 장비인 라우터, 스위치 등의 대역, VLAN 구성, 서브넷 정보 등을 비롯하여 네트워크 구성 앞단에 있는 방화벽, IPS 등의 접근제어정책 (access control rule)은 네트워크 스캔으로 파악하는 데에 한계가 있기 때문에 중앙 관리용 데이터베이스에 직접 저장해야 한다.

3.2.1.3 NMAP

NMAP은 대규모 네트워크를 빠른 속도로 스캔할 수 있는 도구로 네트워크 대역 내 호스트, 운영 체제, 시스템, 서비스 정보 등을 확인할 수 있다. 또 다른 스캔 도구인 OpenVAS와 비교하면 NMAP은 속도가 매우 빠른 대신 기본적인 네트워크 스캐닝 정보만을 제공하여 취약점 정보를 제공하지 않는다. 따라서 자체적으로 구축한 취약점 데이터베이스와 연결하여 호스트의 취약점을 평가한다. 본 연구에서 필요한 정보는 네트워크 대역 내 호스트와 호스트가 사용중인 포트, 운영 체제, 서비스 및 버전 정보이며, 따라서 해당 정보를 빠르게 수집할 수 있는 NMAP을 기반으로 하여 네트워크 정보를 수집한다.

3.2.1.4 운영 체제, 포트, 서비스 스캐닝 방법

NMAP으로는 IP 주소 대역 내 활성화된 호스트 및 MAX 주소, 호스트의 열린 포트 정보 및 포트에서 사용 중인 서비스와 버전 정보, 장치 모델 추측 정보 및 운영 체제 버전 정보를 수집한다. NMAP의 운영 체제 버전 정보는 정확도가 떨어지므로, 본 연

구에서는 운영 체제 추측 정확도를 높이기 위해 TTL 값 및 SMB 메시지 정보를 추가적으로 수집하여 활용하였다. TTL 값은 네트워크에서 패킷 데이터의 유효 기간을 나타내는 값이며 디폴트 값은 운영체제의 종류(Windows/Linux/MacOS X 등) 및 버전에 따라 상이하게 나타나므로 이를 이용해 운영체제를 추측할 수 있다. Table. 4는 운영 체제에 따른 TTL 값의 예시를 나타낸 표이다[18]. 또한 윈도우에서 주변 장치와의 데이터 공유를 위해 사용되는 SMB의 메시지 값을 이용해 윈도우에 한해 상세 버전 정보를 확인할 수 있다.

Table 2. Example TTL values of OS version

OS version	TTL value
Linux 2.4 kernel	255
Windows Server 2008	128
Windows 7	128
Windows XP	128
RedHat 9	64
FreeBSD 5	64
MacOS X (10.5.6)	64
AIX	60

3.2.1.5 스캐닝 주기 및 방법

정보 수집을 위한 최초 작업으로 시스템 및 네트워크 전체에 걸친 스캐닝을 진행한다. 모든 스캐닝 포인트는 각각 관리하는 존의 정보를 수집하여 중앙 관리자에게 전달한다. 이후 전체 스캔, 호스트 변경 사항 스캔, 존 별 스캔으로 나누어 각자 다른 주기로 스캔을 진행한다. 전체 스캔의 경우 다른 스캔에 비해 걸리는 시간이 길기 때문에 주 1회 정도가 적당하다. 또한 동적 네트워크에서도 빠르게 변경사항을 반영할 수 있도록 짧은 주기로 호스트 변경 사항을 스캔한다. 호스트 변경 사항 스캔은 새로운 호스트가 생성된 경우 이를 바로 반영할 수 있도록 존 별로 자주 진행되며, 호스트의 생성만 확인하여 걸리는 시간이 짧기 때문에 스캔 주기는 1분 이내가 적당하다. 이를 통해 새로운 호스트의 생성이 탐지된 경우 해당 존의 존 별 스캔을 통해 정보를 업데이트해 중앙 데이터베이스에 전달하며, 전체 스캔과 존 별 스캔은 관리자가 원하는 시간에 임의로 진행할 수 있다.

3.2.2 SDN 환경

3.2.2.1 SDN

소프트웨어 정의 네트워킹 (SDN, Software Defined Network)이란 소프트웨어 프로그래밍으로 네트워크 리소스의 관리를 편리하게 처리할 수 있게 하는 네트워킹 기술이다. SDN 환경에서는 컨트롤러(소프트웨어 기반 네트워크 제어 장치)가 네트워크 환경 내의 소프트웨어 및 스위치, 라우터 등의 모든 장치를 표준화된 API (Application Programming Interface)를 이용해 직접 제어할 수 있다[19].

3.2.2.2 SDN 환경에서의 시스템 정보 수집 자동화

SDN 환경에서는 일반 네트워크 환경과는 달리 중앙 관리자가 API를 통해 모든 호스트에서 소스코드를 실행시킬 수 있다. 따라서 스캐닝 포인트 없이 각 호스트가 스스로의 정보를 중앙 관리자용 데이터베이스에 전송하고, 중앙 관리자는 새로운 호스트의 생성이 탐지되었을 때에 해당 호스트에 소스코드를 전송하여 실행시킨다. 예로, 각 호스트는 netstat 명령어를 이용해 사용 중인 포트 정보를, python 라이브러리 중 platform을 이용해 운영 체제 정보를 수집할 수 있다. 또한 윈도우는 wmic 명령어로, 리눅스는 ps, dpkg-s, dpkg-query 명령어로 실행 중인 서비스 정보를 수집할 수 있다. 이렇게 수집한 정보는 중앙 관리용 데이터베이스로 전달되어 attack graph 생성에 사용된다.

3.3 시스템 평가 지표 기준 수집

보안성 평가 시 정확한 결과를 얻기 위해서는 대상 시스템 내부의 모든 구성 요소(장치)에 대해 취약점을 진단할 필요가 있다[20]. 각 장치의 취약성은 운영 체제, 사용 중인 서비스 등의 취약성에 따라 결정된다. 본 연구에서는 운영 체제, 사용 중인 서비스의 취약성과 함께 열린 포트 정보를 이용하여 시스템의 취약성을 평가한다. 운영 체제 또는 서비스의 취약성에 대한 평가 지표는 운영 체제와 서비스 제품명 및 버전 별 취약점 개수와 CVSS 점수를 통해 산정한 가중치를 통해 나타낼 수 있다. 운영 체제 또는 소프트웨어의 보안성은 개발 벤더의 시큐어코딩 지

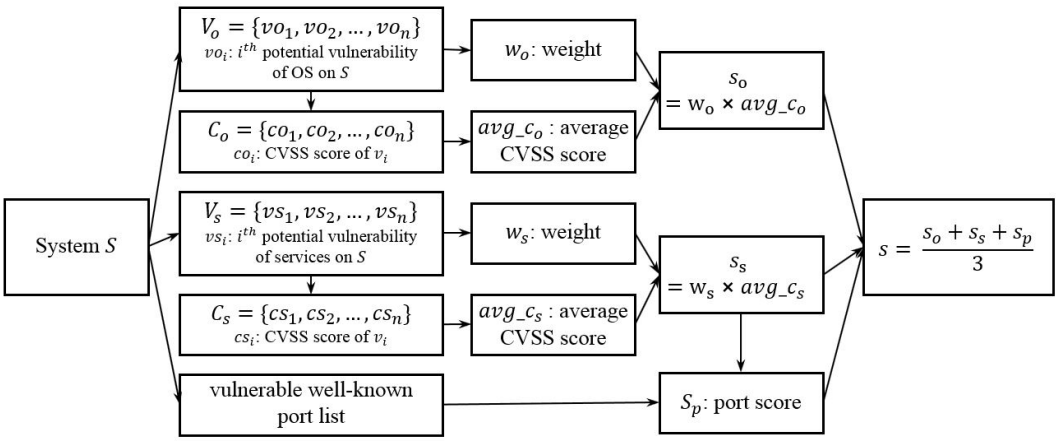


Fig. 3. Calculating vulnerability score of a system

침, 보안개발 프로세스 등 보안 성숙도에 의존성이 높다. 앞서 구축된 CVE 취약점 데이터베이스를 통해 운영 체제와 소프트웨어의 제품 및 버전 별 취약점 개수를 파악하고, 이를 평가 지표 중 하나로 이용한다. 또한 각 취약점 개수가 많은 운영 체제 및 소프트웨어라 하더라도, 해당 취약점의 CVSS 값이 낮으면 리스크가 큰 취약점은 적다는 의미로 해석할 수 있다. 단순히 취약점 개수만으로 특정 시스템의 위험성을 산출하면 오류가 생길 수 있기 때문에, 취약점의 점수를 같이 고려해야 한다. 이에 따라 운영 체제와 소프트웨어의 제품 및 버전 별 취약점들의 CVSS 점수 평균을 취약점 개수와 함께 평가 지표로 활용한다.

시스템에서 사용 중인 포트 중 잘 알려진 포트(well-known port)가 존재하며, 해당 포트로 이용 중인 서비스가 Internet Assigned Numbers Authority (IANA)에서 권고하는 포트-서비스 페어와 일치하다면 공격자가 서비스의 취약점을 알고 있을 때에 해당 포트로 공격을 시행하기 용이하다. 사용 중인 서비스의 취약성 점수가 높은 포트들 중 포트 번호와 사용 중인 서비스가 IANA의 권고안과 같은 포트를 본 논문에서는 '취약한 포트'로 간주한다.

Fig. 3은 하나의 시스템에 대한 취약성 점수 계산 방법을 도식화한 것이다. 각 시스템의 취약성 점수는 운영 체제의 취약성 점수, 사용 중인 서비스의 취약성 점수와 포트 취약성 점수의 가중치 합으로 나타나며, 이후 단계에서 최단 공격 경로 선정 시, 각 경로에 대한 공격 가능성을 확률적으로 산출해야 하기 때문에 0 이상 1 이하의 값으로 나타낸다. 운영 체제의 취약성 점수는 취약점 개수를 1 이상 10 이하로 나타낸 값과 CVSS 평균 점수를 1 이상 10 이하로 나타낸 값의 곱으로 계산하고, 이를 100으로 나누어 0~1 사이의 값으로 부여한다. 서비스의 취약성 점수도 같은 방법으로 사용 중인 모든 서비스의 취약점 개수와 CVSS 평균값을 이용해 계산한다. 포트 취약성 점수는 서비스 취약성 점수가 높은 포트들 중 포트 번호와 사용 중인 서비스가 IANA의 권고안과 같은 포트의 개수를 0~1 사이의 값으로 나타내어 부여한다. 마지막으로 운영 체제 취약성 점수, 서비스 취약성 점수, 포트 취약성 점수를 모두 동일한 가중치로 더하여 최종적으로 시스템의 취약성 점수를 0~1 사이의 값으로 나타낸다.

3.4 Attack graph 생성

3.4.1 최초 attack graph 생성 방법

가장 위험도가 높은 공격 경로를 계산하기 위해 우선 특정 타겟에 공격 가능한 모든 공격 경로를 추출해 데이터베이스에 기록해 놓는다. 공격 경로를 추출하는 데에 취약한 호스트를 손상시킬 수 있는 모든 경로를 보여주는 full attack graph를 사용하면 시간복잡도가 증가하므로 기존 attack graph 모델 중 하나인 predictive attack graph 방식을 응용하여 공격 경로를 생성해 나간다. Predictive attack graph란 모든 경로에 대해 점수를 계산하지 않고, 열린 포트가 존재하는 등 두 자산 간에 어

느 정도 접속이 발생할 가능성이 있는 경우에만 엣지를 그리는 방식이다[6]. Predictive attack graph는 처음 지정된 시작 위치에서 공격자가 손상시킬 수 있는 모든 호스트를 결정하는데, 이 때 취약점 제거 효과를 정확하게 예측하여 방화벽 등에 의해 취약점이 제거되는 경우 그래프에서 관련 엣지를 제거할 수 있어야 한다. 이를 통해 시작 위치에서 도달할 수 없는 노드를 알아내어 그래프 생성 비용을 줄일 수 있다. 본 연구에서는 포트, 서비스 등 두 자산 간에 정책적으로 연결된 통로가 없다면 공격 경로가 될 가능성이 매우 낮다고 간주하여 엣지를 그리지 않는 방식을 선택했다. 추가적으로, 최근 점점 많이 사용되고 있는 가상 네트워크 환경 상에 수시로 발생하는 변경사항을 반영할 수 있도록 하였다.

Attack graph를 생성하는 방법은 다음과 같다. 최초 그래프 생성은 Fig. 4와 같은 네트워크 구성을 가진 시스템을 기준으로 진행하였다. 타겟은 호스트 A로, 소스는 {호스트 B}로 지정하였다. 타겟은 관리자가 지정한 호스트이고, 소스는 관리자가 지정한 호스트 또는 외부망에서 접근이 가능한 호스트이다. 이 때 소스 호스트의 경우 다수가 지정될 수 있기 때문에 여러 호스트를 포함한 집합으로 표현하였다.

우선 타겟 호스트 A로 공격이 가능한 경로가 존재하는 모든 호스트들과 호스트 A를 엣지로 연결한다. 즉, '호스트 A에서 호스트 B로 연결 가능'하다는 것은 B에서 A로 공격 가능한 경로가 존재함을 의미한다. 호스트 A와 한 호스트의 연결이 새로 생성되었는데, 이 호스트가 소스에 속한 호스트라면 이미 공격 경로가 완성되었으므로 확장을 하지 않고, 호스트 A와 연결이 가능한 다른 호스트를 탐색한다. 새로 연결이 생성된 호스트가 소스에 속하지 않은 호스트라면 해당 호스트로 공격이 가능한 경로가 존재하는 모든 호스트들과 연결을 진행한다. 한 경로에서 한 번 지나온 호스트는 다시 지나지 않도록 하며 모든 경로를 리프 노드까지 확장한 후, 다시 호스트 A에서 연결 가능한 다른 호스트를 탐색하는 방식으로 depth-first로 그래프를 확장해 나간다. 이 때 만약 경로가 소스 호스트까지 확장되지 않더라도 네트워크에 변경사항이 발생하여 attack graph 업데이트 시 소스 호스트에 도달하는 경로가 생성될 수 있으므로 삭제하지 않고 데이터베이스에 기록해 둔다.

최초 그래프 생성 예시는 Fig. 5에 나타나 있다. 그래프 생성 방법은 아래와 같으며, 리프 노드의 경우 더 이상 공격 경로를 확장시킬 수 없는 노드 (호스트)를 의미하며, 생성된 모든 경로는 데이터베이스에 저장된다.

1) 네트워크에서 호스트 C는 호스트 A로 공격이 가능하므로 호스트 A → 호스트 C 엣지를 연결
2) 해당 경로에서 지나오지 않은 호스트들 중 호스트 C에서 연결 가능한 호스트를 탐색
3) 호스트 C → 호스트 B 엣지를 연결
4) 소스 호스트인 B에 도달하여 공격 경로가 완성되었으므로 더 이상 확장하지 않음
5) 다시 호스트 C로 돌아와 연결 가능한 호스트를 탐색
6) 호스트 C → 호스트 D 엣지를 연결하고 해당 경로에서 지나오지 않은 호스트들 중 호스트 D에서 연결 가능한 호스트 탐색
7) 호스트 D → 호스트 B 엣지를 연결하고 소스 호스트인 B에 도달하여 공격 경로가 완성되었으므로 더 이상 확장하지 않음
8) 호스트 C에서 리프노드까지 확장 완료하였으므로 다시 호스트 A로 돌아와 연결 가능한 호스트를 탐색
9) 호스트 A → 호스트 D 엣지를 연결하고 호스트 C에서 리프노드까지 확장한 방법과 같은 방법으로 호스트 D에서 리프노드까지 그래프 확장

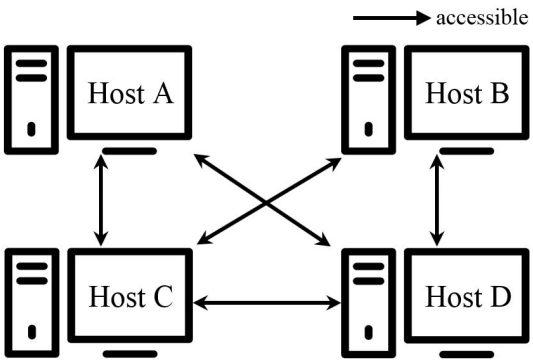


Fig. 4. Network policy

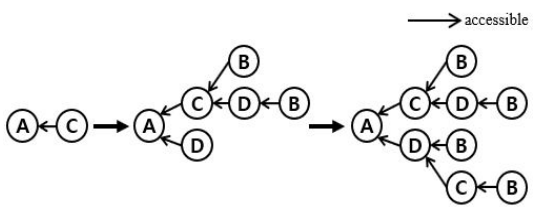


Fig. 5. Attack graph generation

10) 호스트 A로 돌아와 더 이상 연결 가능한 호스트가 탐색되지 않으므로 그래프 확장을 종료

최종적으로 완성된 attack graph는 Fig. 6과 같다.

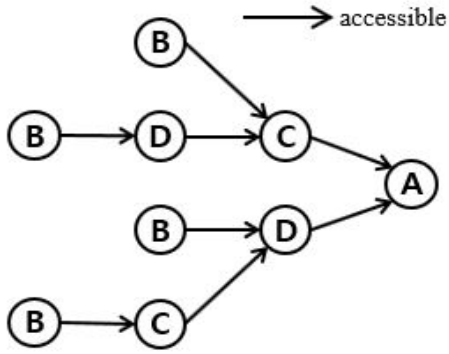


Fig. 6. Attack graph

3.4.2 최적의 공격 경로 선정 방법

3.4.1에서 구한 attack graph 내 여러 공격 경로들 중 최적의 공격 경로를 선정하기 위해 3.3에서 제안한 시스템 취약점 점수를 활용한다. 예를 들어, 각 시스템 별 취약 점수는 Table. 5과 같다고 가정한다.

공격 경로의 점수는 공격 경로 내 각 호스트들의 시스템 취약성 점수의 곱으로 나타낸다. 예를 들어, Fig. 6에서 공격 경로 {B→D→A}의 점수를 score(B→D→A)라 하면, score(B→D→A)는 호스트 D의 취약성 점수, 호스트 A의 취약성 점수의 곱으로 산출한다. 모든 공격 경로에 대해 같은 방식으로 취약성 점수를 계산하면 Fig. 6의 attack graph에서 각 공격 경로의 점수는 Table. 6와 같고, 이 중 가장 점수가 높은 경로인 {B→C→A}를 최적의 공격 경로로 선정한다.

Table 3. Score of each host

host	score
A	0.4
B	0.7
C	0.8
D	0.3

Table 4. Score of each path

path	score
B→C→A	0.7×0.8×0.4=0.224
B→D→C→A	0.7×0.3×0.8×0.4=0.0672
B→D→A	0.7×0.3×0.4=0.084
B→C→D→A	0.7×0.8×0.3×0.4=0.0672

3.4.3 호스트 변경 사항 발생 시 업데이트 방법

네트워크에 변경사항이 발생하면 attack graph를 업데이트해야 한다. 업데이트가 필요한 경우로는 기존 호스트가 사라지거나, 새로운 호스트가 추가되거나, 기존 호스트에서 다른 호스트와의 연결 정보가 변경된 경우가 있다. 본 연구에서는 연산의 효율성을 위해 변경사항이 탐지될 때마다 attack graph를 재생성하지 않고 기존 attack graph를 수정하는 방법을 취한다.

우선, 기존 호스트가 사라지는 경우 해당 호스트가 포함된 경로를 모두 제거한다. 새로운 호스트가 추가되는 경우, 해당 호스트가 연결될 수 있는 모든 호스트에 해당 호스트를 연결한 후 최초 그래프 생성 시 리프 노드까지 그래프를 확장시킬 때와 같은 방법으로 새로 연결한 호스트에서부터 그래프를 확장한다. 호스트 간에 새로운 연결이 생성되었다면 마찬가지로 새로 연결한 호스트에서부터 그래프를 확장한다. 또한 호스트 간의 기존 연결 중 끊긴 연결이 생기면 경로에서 해당 연결 이후를 모두 제거한다. attack graph에 대한 업데이트가 종료되면 공격 경로에 대한 위험도 점수를 재계산한 후 재계산 결과에 따라 취약한 공격 경로를 다시 제안한다.

3.5 알고리즘 구현

Fig. 7은 3.1장~3.4장에서 설명한 알고리즘을 구현한 시스템의 전반적인 프레임워크이다. 데이터베이스는 총 17개의 테이블로 구성되어 있으며, 크게 네트워크 토폴로지 정보, 취약점 데이터베이스 정보, 어택그래프 정보 세 범주의 정보를 담고 있다. 중앙 분석 시스템의 동작은 크게 스캐닝 포인트 관리, 취약점 데이터베이스 관리, attack graph 생성으로 나눌 수 있다. 네트워크 관리자에게 웹으로 모든 기능을 제공하여 네트워크 시스템 관리를 용이하게 하였다.

Scanning point manager는 flask를 이용하

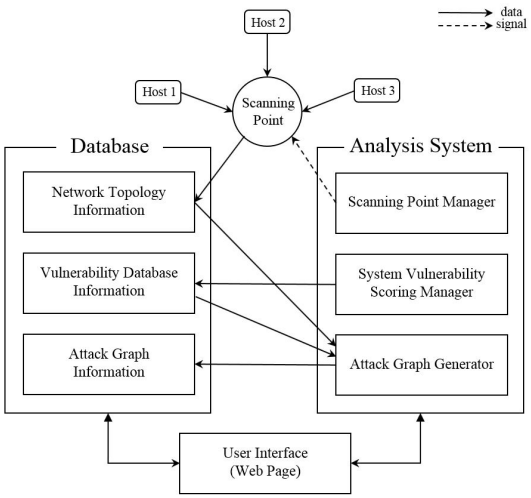


Fig. 7. Implementation

여 주기적으로 모든 스캐닝 포인트에 스캐닝 시작 신호를 전달하며, 네트워크 관리자가 전체 스캐닝을 실행하면 모든 스캐닝 포인트에, 특정 존 스캐닝을 실행하면 해당 존의 스캐닝 포인트에 스캐닝 시작 신호를 전달한다. 신호를 받은 스캐닝 포인트는 3.2장에서 기술한대로 존 내의 모든 호스트의 정보를 스캔하여 데이터베이스에 저장한다. 이 때 SDN 환경의 경우 스캐닝 포인트가 존재하지 않아 각 호스트가 모두 직접 신호를 전달받아 자신의 정보를 데이터베이스에 저장한다.

Vulnerability database manager는 CVE와 CPE, CVSS 정보를 수집해 데이터베이스에 저장한다. CVE 정보와 CVE에 해당하는 CVSS 정보는 NVD에서 제공하는 XML 형식의 데이터에서 추출하였고, CPE 정보는 NIST에서 제공하는 XML 형식의 데이터에서 추출하였다. 또한 추출한 데이터를 바탕으로 3.3장에서 기술한 방법대로 시스템 별 취약성 점수를 산정한다.

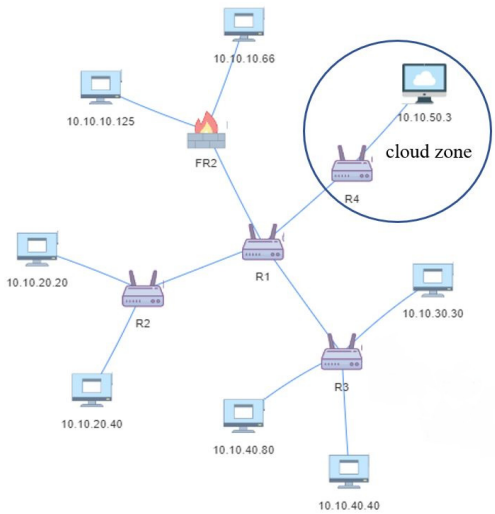
Attack graph generator는 데이터베이스의 네트워크 토폴로지 정보와 취약점 데이터베이스 정보를 이용하여 3.4장의 알고리즘대로 attack graph를 생성하고, 각 경로의 점수를 계산하여 데이터베이스에 저장한다.

네트워크에 새로운 호스트가 추가되거나, 취약성 점수가 변경되거나, 포트가 제거되는 등 동적 네트워크에서 변경사항이 발생하는 경우 전체 네트워크 스캔부터 다시 진행하는 것은 효율적이지 않으므로 경우에 따라 다르게 진행한다. 예를 들어, 특정 존에 새로운 호스트가 추가되는 경우, 해당 호스트만을 스캔하여 취약성 점수를 부여하고 attack graph를 재생성한 후 경로 별 점수 산정을 다시 진행한다. 또한 패치 등으로 특정 호스트에 변경사항이 탐지된 경우에는 해당 호스트를 스캔하여 시스템 취약성 점수를 다시 계산하고 경로 별 점수 산정을 다시 진행한다. 이 경우 attack graph에는 변화가 없기 때문에 attack graph를 재생성하지는 않는다.

IV. 시뮬레이션

4.1 환경 설정 및 사전 정보 입력

가상의 네트워크 환경을 구축하여 3장에서 구현한 시스템을 적용하였다. 가상의 네트워크 환경은 Fig. 8과 같으며, 4개의 존 중 1개의 존은 SDN 환경(클라우드 존)으로 VMware ESXi를 이용하여 구축하였다[21]. 이후 각 존마다 스캐닝 포인트를 선택하고, 라우터, 스위치의 정보를 입력하였다.

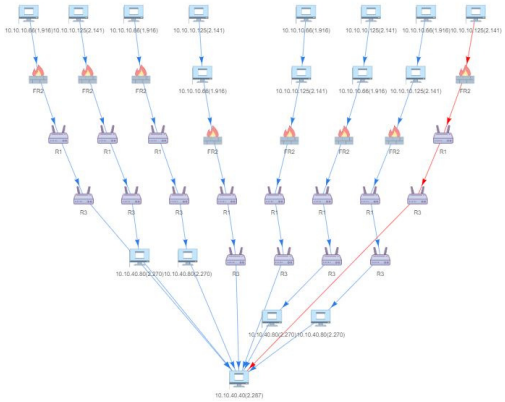


Fig, 8. Network topology

4.2 Attack Graph 생성

IP 주소가 10.10.10.125인 호스트와 10.10.10.66인 호스트를 소스 호스트로, 10.10.40.40인 호스트를 타겟 호스트로 설정하여 attack graph를 생성하였다. 생성한 총 공격 경로는 8개이며, attack graph는 Fig. 9와 같다. 이때

공격 경로가 10개를 초과하는 경우 모두 시각화하는 것은 부적합하기에 가장 점수가 높은 10개의 경로만 시각화한다. 각 공격 경로의 점수는 Fig. 10과 같이 웹으로 나타내어 관리자가 확인할 수 있도록 한다. Fig. 9에서 가장 공격 가능성이 높은 공격 경로는 {10.10.10.125-〉 FR2-〉 R1-〉 R3-〉 10.10.40.40}이다.



Fig, 9. Attack graph

Path	Score
10.10.10.125->FR2->R1->R3->10.10.40.40	0.04896
10.10.10.66->FR2->R1->R3->10.10.40.40	0.04382
10.10.10.125->FR2->R1->R3->10.10.40.80->10.10.40.40	0.01111
10.10.10.66->FR2->R1->R3->10.10.40.80->10.10.40.40	0.00995
10.10.10.125->10.10.10.66->->FR2->R1->R3->10.10.40.40	0.00938
10.10.10.66->10.10.10.125->->FR2->R1->R3->10.10.40.40	0.00938
10.10.10.125->10.10.10.66->FR2->R1->R3->10.10.40.80->10.10.40.40	0.00204
10.10.10.66->10.10.10.125->FR2->R1->R3->10.10.40.80->10.10.40.40	0.00204

Fig. 10. Score of attack path

4.3 네트워크 변경사항 탐지

네트워크에 변경사항이 발생한 경우, 이를 탐지하여 attack graph를 다시 생성한다. 새로운 호스트가 생성되거나 기존 호스트가 제거되는 경우 또는 네트워크 관리자가 취약한 호스트를 업데이트하는 경우, 주기적인 스캐닝을 통해 이를 감지하여 네트워크 토폴로지를 변경하고, attack graph를 재생성한다.

먼저, Fig. 11과 같이 IP 주소가 10.10.30.60인 호스트가 새로 생성되는 경우이다. 스캐닝 포인트에 새로운 호스트가 탐지된 경우 자동으로 해당 호스트의 정보를 수집하여 데이터베이스를 업데이트하며, attack graph도 다시 생성한다. Fig. 12는 변경

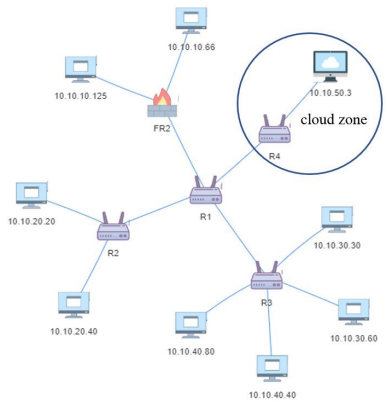


Fig. 11. Network topology (add host)

사항을 반영한 attack graph이고, Fig. 13은 각 공격 경로의 점수이다. 새로 생성된 호스트가 공격 경로에 추가되어 공격 가능한 경로가 8개에서 16개로 변경되었다. 가장 공격 가능성이 높은 공격 경로는 {10.10.10.125-〉 FR2-〉 R1-〉 R3-〉 10.10.40.40}로 변경되지 않았지만 타겟 호스트로의 침투 위험이 증가하였다.

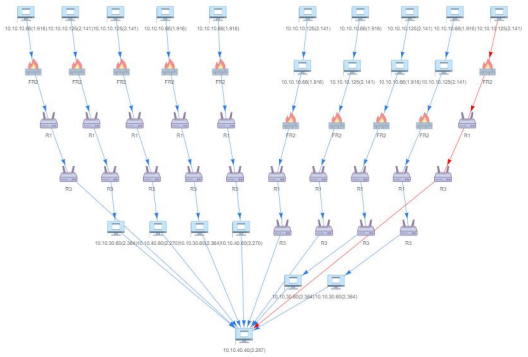


Fig. 12. Attack graph (add host)

Path	Score
10.10.10.125-> FR2-> R1-> R3-> 10.10.40.40	0.04896
10.10.10.66-> FR2-> R1-> R3-> 10.10.40.40	0.04382
10.10.10.125-> FR2-> R1-> R3-> 10.10.30.60-> 10.10.40.40	0.01158
10.10.10.125-> FR2-> R1-> R3-> 10.10.40.80-> 10.10.40.40	0.01111
10.10.10.66-> FR2-> R1-> R3-> 10.10.30.60-> 10.10.40.40	0.01036
10.10.10.66-> FR2-> R1-> R3-> 10.10.40.80-> 10.10.40.40	0.00995
10.10.10.125-> 10.10.10.66-> FR2-> R1-> R3-> 10.10.40.40	0.00938
10.10.10.66-> 10.10.10.125-> FR2-> R1-> R3-> 10.10.40.40	0.00938
10.10.10.125-> 10.10.10.66-> FR2-> R1-> R3-> 10.10.30.60-> 10.10.40.40	0.00222
10.10.10.66-> 10.10.10.125-> FR2-> R1-> R3-> 10.10.30.60-> 10.10.40.40	0.00222

Fig. 13. Score of attack path (add host)

본 연구에서 공격은 열린 포트를 통해 이루어지므로 열린 포트 중 취약한 포트로의 통신을 차단하는 경우에도 attack graph에 변화가 생길 수 있다. Fig. 11에서 IP 주소가 10.10.10.125인 호스트의 열린 포트 중 잘 알려진 포트인 21번과 80번을 차단하는 경우, 스캐닝 포인트에서 변경사항을 탐지하여 해당 호스트의 정보를 업데이트한다. Fig. 14는 새로 생성한 attack graph이고, Fig. 15는 각 공격 경로의 점수이다. 공격 경로는 16개로 포트를 차단하기 전과 동일하게 나타났지만 잘 알려진 포트를 차단하여 시스템 취약성 점수에 변화가 생겼기 때문에 가장 공격 가능성이 높은 경로가 {10.10.10.66-〉 FR2-〉 R1-〉 R3-〉 10.10.40.40}으로 변경된 것을 확인할 수 있다.

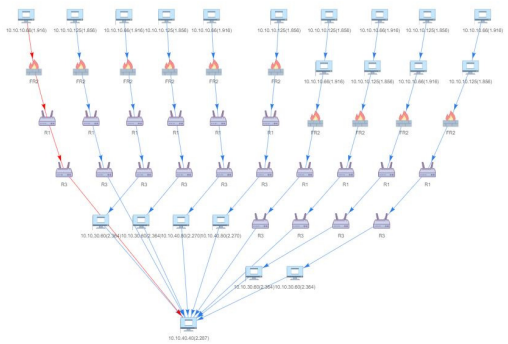


Fig. 14. Attack graph(block port)

Path	Score
10.10.10.66-> FR2-> R1-> R3-> 10.10.40.40	0.04382
10.10.10.125-> FR2-> R1-> R3-> 10.10.40.40	0.04245
10.10.10.66-> FR2-> R1-> R3-> 10.10.30.60-> 10.10.40.40	0.01036
10.10.10.125-> FR2-> R1-> R3-> 10.10.30.60-> 10.10.40.40	0.01003
10.10.10.66-> FR2-> R1-> R3-> 10.10.40.80-> 10.10.40.40	0.00995
10.10.10.125-> FR2-> R1-> R3-> 10.10.40.80-> 10.10.40.40	0.00964
10.10.10.125-> 10.10.10.66-> FR2-> R1-> R3-> 10.10.40.40	0.00813
10.10.10.66-> 10.10.10.125-> FR2-> R1-> R3-> 10.10.40.40	0.00813
10.10.10.125-> 10.10.10.66-> FR2-> R1-> R3-> 10.10.30.60-> 10.10.40.40	0.00192
10.10.10.66-> 10.10.10.125-> FR2-> R1-> R3-> 10.10.30.60-> 10.10.40.40	0.00192

Fig. 15. Score of attack paths(block port)

V. 결 론

본 연구에서는 동적으로 변화하는 네트워크 환경에 맞춰 네트워크 시스템 내 변경사항을 효율적으로 식별하고 위험에 대응할 수 있는 시스템을 구현하고, SDN 환경이 포함된 가상의 네트워크 환경에 적용하여 결과를 제시하였다. 기존의 attack graph 연구는 취약점 정보 및 네트워크 구성 정보가 모두 식별되었다는 것을 전제조건으로 하여 진행된 데에 비해 본 연구에서 제안한 시스템은 네트워크 구성 정보 스캐닝, 시스템 취약성 평가 지표 수립, attack graph 생성 모두를 포함한다. 또한 네트워크 관리자가 입력해야 하는 정보는 존 정보, 스캐닝 포인트 정보, 라우터 정보, 스위치 정보 네 가지로 최소화하여 편의성을 높이고, predictive attack graph를 통해 가능한 경로만을 고려하여 attack graph 생성 시 효율성을 확보하였다. 마지막으로 가상의 네트워크 환경에 적용하는 과정과 결과를 제시하여 실제 기업 환경에 적용하여도 유의미한 결과를 도출할 수 있다는 것을 입증하였다.

MTD 개념을 이용한 보안 전략에 대한 연구가 활발히 진행되고 있고, 근본적인 보안 전략으로 각광받고 있는 만큼 이에 적합한 네트워크 관리 시스템에 대한 연구 또한 진행되어야 한다. 본 연구에서 시스템의 동작이 정적인 정보를 이용하여 attack graph를 생성하는 데에 그치지 않고, 동적으로 변화하는 환경을 자동으로 스캔하여 결과를 제공하기 때문에 스캐닝 주기를 짧게 하거나 MTD의 구성 정보 변경 주기와 동일하게 한다면 MTD를 보안 전략으로 취하는 시스템에도 적용할 수 있다. 또한 이미 많은 기업에서 적용 중인 SDN 환경에도 적용할 수 있어 기존 네트워크 관리 시스템에 비해 활용 범위가 다양하다. 따라서 제안한 시스템이 실제 기업의 네트워크 관리자로 하여금 효율적으로 방어 전략을 수립하는 데에 많은 도움을 줄 것으로 기대된다.

References

[1] KISA, Monthly Special Report, Jul. 2012.

[2] Jin B. Hong, Dong Seong Kim, Chun-Jen Chung, Dijiang Huang, "A survey on the usability and practical applications of Graphical Security Models", Computer Science Review, vol. 26, pp. 1-16, Nov. 2017.

[3] Nirnay Ghosh , Ishan Chokshi, Mithun Sarkar , Soumya K. Ghosh , Anil Kumar Kaushik , Sajal K. Das, "NetSecuritas:

An Integrated Attack Graph-based Security Assessment Tool for Enterprise Networks", Proceedings of the 2015 International Conference on Distributed Computing and Networking, p.1-10, Jan. 2015.

[4] L. Wang, S. Jajodia, A. Singhal, S. Noel, D. Gritzalis, B. Preneel, M. Theoharidou, "k-Zero Day Safety: Measuring the Security Risk of Networks against Unknown Attacks", Proc. of the 15th European Symposium on Research in Computer Security (ESORICS 2010), Springer Berlin Heidelberg, vol. 6345, pp. 573-587, Dec. 2010.

[5] Ji Hong Kim, Huy Kang Kim, "Automated Attack Path Enumeration Method based on System Vulnerabilities Analysis", Journal of the Korea Institute of Information Security & Cryptology, Vol. 22, No. 5, pp.1079-1090, Oct. 2012.

[6] Moon, Y. H., Kim, J. H., Kim, D. S., Kim, H. K., "Hybrid Attack Path Enumeration System Based on Reputation Scores", Computer and Information Technology (CIT), 2016 IEEE International Conference, pp. 241-248. Dec. 2016.

[7] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, R. Cunningham, "Validating and Restoring Defense in Depth Using Attack Graphs", Military Communications Conference 2006. MILCOM 2006. IEEE, pp. 1-10, Oct. 2006.

[8] R. P. Lippmann, K. W. Ingols, C. Scott, K. Piwowarski, K. J. Kratkiewicz, M. Artz, R. K. Cunningham, "Evaluating and strengthening enterprise network security using attack graphs", MIT Lincoln Laboratory, Lexington, MA, Tech. Oct. 2005.

[9] S. Noel, S. Jajodia, "Managing attack graph complexity through visual hierarchical aggregation", VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pages 109-118, Oct. 2004.

[10] Jing Jiang, XiaoLi Xu, Ning Cao, "Research on Improved Physical Topology Discovery Based on SNMP", 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Jul. 2017.

[11] Yusuf, S. E., Ge, M., Hong, J. B., Kim, H. K., Kim, P., Kim, D. S., "Security Modelling and Analysis of Dynamic Enterprise Networks", Computer and Information Technology (CIT), 2016 IEEE International Conference, pp. 249-256, Dec. 2016.

[12] M. Ge, H. K. Kim and D. S. Kim, "Evaluating Security and Availability of Multiple Redundancy Designs when Applying Security Patches", 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Denver, CO, pp. 53-60, Jun. 2017.

[13] S. Khan, A. Gani, A. W. A. Wahab, M. Guizani, M. K. Khan, "Topology discovery in software defined networks: Threats taxonomy and state-of-the-art", IEEE Communications Surveys Tutorials, vol. 19, no. 1, pp. 303-324, Aug. 2016.

[14] Farzaneh Pakzad, Marius Portmann, Wee Lum Tan, Jadwiga Indulska, "Efficient topology discovery in software defined networks", 8th International Conference on Signal Processing and Communication Systems (ICSPCS), Dec. 2014.

[15] https://cve.mitre.org/

〔16〕 Joong Gil Park, "A development of weakness calculation method for information system", Journal of the Korea Institute of Information Security & Cryptology, Vol. 17, No. 5, pp.131-139, Oct. 2007.

〔17〕 P. Mell, K. Scarfone, and S. Romanosky, "A Complete Guide to the Common Vulnerability Scoring System Version 2.0", Jul. 2007

〔18〕 Ryo Yamada and Shigeki Goto, "Using abnormal TTL values to detect malicious IP packets", Proceedings of the Asia-Pacific Advanced Network 2012, Vol. 34, pp.17-34, Aug. 2012.

〔19〕 "Software-Defined Networking: The New Norm for Networks", ONF White Paper, pp.3-6, Apr. 2012.

〔20〕 Jongbin Ko, Seokjun Lee, Taeshik Shon, "Security Threat Evaluation for Smartgrid Control System", Journal of the Korea Institute of Information Security & Cryptology, Vol. 23, No. 5, pp.873-883, Oct, 2013.

〔21〕 Charu Chaubal, "The architecture of vmware esxi", VMware White Pap, Oct. 2008.

〈저자소개〉

문 주 연 (Joo Yeon Moon) 학생회원
2017년 2월: 고려대학교 컴퓨터학과 졸업
2017년 3월~현재:고려대학교 정보보호대학원 정보보호학과 석사과정
〈관심분야〉 데이터마이닝, 유저 행위 분석, 이상 탐지, 온라인게임 보안

김 태 규 (Taekyu Kim) 정회원
2000년 2월: 중앙대학교 컴퓨터공학 학사
2006년 5월: The University of Arizona 컴퓨터공학 석사
2008년 5월: The University of Arizona 컴퓨터공학 박사
〈관심분야〉 사이버보안 킬체인 및 TTP, 임베디드 시스템 보안, 시스템 모델링 및 시뮬레이션

김 인 성 (Insung Kim) 정회원
2000년 8월: 중앙대학교 산업정보학과 졸업
2008년 2월: 고려대학교 정보경영공학전문대학원 정보경영공학과 석사
2016년 3월~현재: 중앙대학교 융합보안학과 박사과정
〈관심분야〉 사이버보안, 위협 인텔리전스

김 휘 강 (Huy Kang Kim) 종신회원
1998년 2월: KAIST 산업경영학과 학사
2000년 2월: KAIST 산업공학과 석사
2009년 2월: KAIST 산업및시스템공학과 박사
2004년 5월~2010년 2월: 엔씨소프트 정보보안실장, Technical Director
2010년 3월~2014년 12월: 고려대학교 정보보호대학원 조교수
2015년 1월~현재: 고려대학교 정보보호대학원 부교수
〈관심분야〉 온라인게임 보안, 네트워크 보안, 네트워크 포렌식