Journal of KIIT. Vol. 23, No. 12, pp. 329-339, Dec. 31, 2025. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2025.23.12.329

템플릿 기반 공격 트리 자동 생성 도구의 문제점 및개선 방안에 관한 연구

최유정*[1], 조광수*[2], 곽지원*[3], 김승주**

A Study on the Limitations and Improvements of Template-based Attack Tree Auto-Generation Tool

Yujeong Choi*[1], Kwangsoo Cho*[2], Jiwon Kwak*[3], and Seungjoo Kim**

이 논문은 2025년도 정부(방위사업청)의 재원으로 국방기술진흥연구소의 지원을 받아 수행된 연구임
(KRIT-CT-24-001, 국방우주보안특화연구실)

요 약

공격 트리 자동 생성 기법은 시스템 설명을 규칙에 따라 변환하는 모델 기반, 보안 속성을 분석해 구성하는 분석 기반, 사전 정의된 취약점 패턴을 매칭하는 취약점 기반 접근 방식으로 구분된다. 본 연구는 취약점 패턴 매칭 방식 도구의 경로 누락과 중복 경로 문제를 개선한다. 기존 도구는 실제 공격 가능성을 반영하지 못하고 분석 과정의 명확성을 떨어뜨리는 한계가 있다. 본 연구는 이러한 문제를 해결하기 위해 경로 탐색 알고리즘을 개선하고 중복 제거 로직을 추가하여 기존 도구와 비교·검증하였다. 그 결과, 제안 방법은 공격자가 진입점에서 목표 지점까지 도달할 수 있는 경로를 기존 도구 대비 6.5배 더 많이 식별하여 약 84.6%의 잠재적 위협 누락 문제를 해결하였으며, 중복된 공격 단계를 효과적으로 제거하여 트리의 가독성을 크게 향상시켰다.

Abstract

Automated attack tree generation approaches can be classified into model-driven, analysis-driven, and vulnerability-driven methods. This study improves vulnerability-driven tools that rely on predefined attack patterns and suffer from missing attack paths and duplicated branches. Existing tools fail to sufficiently reflect actual attack feasibility, thereby reducing the clarity of the analysis process. To address these limitations, we enhance the path-exploration algorithm and incorporate a branch-deduplication mechanism, then validate the improvements through comparison with the baseline tool. As a result, the proposed method identifies 6.5 times more valid attack paths from entry points to target assets and resolves approximately 84.6% of the potential threat omission issue, while effectively removing redundant attack steps and significantly improving the readability of the generated attack trees.

Keywords

attack tree, automated generation, template-based method, threat analysis

* 고려대학교 정보보호대학원 정보보호학과
- ORCID[1]: https://orcid.org/0009-0007-6185-6434
- ORCID[2]: https://orcid.org/0009-0005-9863-5075
- ORCID[3]: https://orcid.org/0009-0008-1560-7442
** 고려대학교 정보보호대학원 정보보호학과 정교수(교신저자)
- ORCID: https://orcid.org/0000-0002-2157-0403

· Received: Nov. 20, 2025, Revised: Dec. 08, 2025, Accepted: Dec. 11, 2025
· Corresponding Author: Seungjoo Kim
Dept. of Cyber Security, Korea University School of Cybersecurity
Tel.: +82-2-3290-4897, Email: skim71@korea.ac.kr

Ⅰ. 서 론

현대 IT 환경은 클라우드, IoT, 차량 시스템 등 다양한 도메인에서 소프트웨어, 하드웨어, 데이터 서비스, 네트워크 요소가 복합적으로 구성되며, 하나의 아키텍처 내에 여러 기능 컴포넌트와 통신 프로토콜이 공존한다[1][2]. 복잡성이 증가함에 따라 공격자가 목표로 하는 공격 지점까지 도달하기 위한 공격 경로의 다양성 또한 함께 증가한다[3]. 따라서 복잡한 만큼 다양해진 보안 위협 경로를 체계적으로 분석할 방법이 필요하다.

공격 트리[4]는 시스템이 공격받을 수 있는 다양한 방법을 체계적으로 분석하는 그래픽 모델이다. 공격 트리는 개별 공격 단계가 어떻게 결합되어 복잡한 공격을 형성할 수 있는지를 보여주며, 대응책의 식별과 적용을 용이하게 한다[5]. 따라서, 공격 트리는 에너지[6], 헬스케어[7], 소프트웨어 시스템[8]과 같은 다양한 도메인에서 위협 모델링의 핵심 방법으로 인정받고 있다.

그러나 오늘날의 복잡한 IT 시스템 환경에서 공격 트리를 수동으로 구축하는 것은 시간이 많이 소요되고 오류가 발생하기 쉽다[9]. 더욱이 대규모 시스템에서 가능한 접근 경로의 방대한 수는 모든 공격 시나리오를 포착하는 것을 거의 불가능하게 만든다[10]. 결과적으로, 이러한 수동 방식의 한계를 극복하기 위해 공격 트리의 자동 생성이 연구자들의 주목을 받고 있다.

본 연구는 다양한 공격 트리 자동 생성에 관한 연구 중 J. Bryans et al.[10]이 제안한 템플릿 기반 공격 트리 자동 생성 도구의 한계점을 분석하고 이를 보완하는 방안을 제안한다. 해당 도구를 선택한 이유는 (1) 오픈소스이므로 도구를 수정하기가 쉽고, (2) 공격 템플릿이 정의되어 있으면 더 이상 사용자 개입 없이 자동 생성이 가능하며, (3) 실제 산업 시스템을 모델링한 사례를 기반으로 평가가 수행되어, 도구의 적용 가능성과 실용성이 입증되었기 때문이다. 우리가 분석한 해당 도구의 한계점은 다음과 같다. 먼저, 경로 누락이 발생될 수 있다. 또한, 중복으로 인한 가독성 저하가 발생될 수 있다.

본 논문에서는 J. Bryans et al.[10]의 2가지 문제점을 보완한다. 아래는 본 연구에서 제안하는 두 가지의 한계점 완화 방안이다. 첫 번째 방안은 경로 탐색(DFS) 알고리즘의 '조기 종료' 로직을 수정하는 방안, 동일한 노드가 연속으로 중복되는 현상을 제거하는 방안이다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장에서는 공격 트리 생성 도구를 포함한 관련 연구를 논의한다. 3장에서는 템플릿 기반 공격 트리 자동 생성 도구의 한계점에 대해 논의한다. 4장에서는 3장에서 제시한 한계점을 보완하는 방안을 제안한다. 5장에서는 구현한 두 가지 보완 방안을 적용한 도구와 기존 도구의 결과를 비교한다. 마지막으로 6장에서는 논문을 결론짓고 향후 연구 방향을 제시한다.

Ⅱ. 관련 연구

공격 트리를 자동으로 생성하기 위한 다양한 방법들이 지금까지 제안되었다. 먼저 R. Vigo et al.[11]와 M. G. Ivanova et al.[12]은 시스템과 공격자를 각각 그래픽과 수식 기반 모델로 표현한 뒤, 공격자의 목표 지점을 기준으로 트리를 자동 구성하는 모델 기반 접근 방식을 제안하였다. 이 방식은 모델로부터 트리를 자동 생성할 수 있다는 장점이 있지만, 생성된 트리가 실제 공격 시나리오를 정확히 반영하는지 보장할 수 없다는 의미론적 정합성 부족의 문제가 있다.

이를 해결하기 위해 S. Pinchinat et al.[9]은 시스템 모델로부터 공격 그래프를 도출하고, 사용자가 이를 기반으로 트리를 구성하는 ATSyRA라는 반자동 도구를 제안하였다. 앞선 문제였던 의미론적 불일치 문제는 사용자가 실제 공격 경로를 정제해 가며 트리를 구성함으로써 일정 부분 해결되었지만, 이 연구에는 사용자 개입이 필수적이라는 한계가 있다.

R. Jhawar et al.[13]에서는 의미론적 불일치 문제를 해결하면서도 S. Pinchinat et al.[9]에서의 수작업 정제 한계를 보완하기 위해 기존 공격 트리에 라이브러리를 활용해 누락된 단계를 자동 보완하는 증강 기법을 제안하였다. 그러나 여전히 초기 트리 품질에 따라 결과가 달라진다는 점에서 한계가 존재한다.

본 연구가 개선 대상으로 삼는 J. Bryans et

al.[10]은 위 문제들을 해결하고자 템플릿 기반 공격 트리 자동 생성 기법을 제안하였다. 이 방법은 공격 시나리오를 일반화된 템플릿으로 정의하고, 시스템 모델 내 자산 및 행위 요소에 템플릿을 자동으로 매핑하여 사용자 개입 없이 공격 트리를 구성한다. 이러한 접근은 기존 도구들의 의미론적 연계 부족과 낮은 자동화 수준의 한계를 해소하였다.

그러나 이 기법 또한 다음과 같은 한계를 내포한다. 첫째, 해당 도구는 템플릿을 적용할 때 시스템 모델 내 직접 연결된 두 자산 간의 관계만 탐색 대상으로 삼는다. 이러한 특징은 여러 컴포넌트를 거쳐 도달할 수 있는 간접적인 경로는 탐지 대상에 반영되지 않아, 실현 가능한 공격 경로가 트리에서 누락될 수 있는 한계가 있다. 둘째, 템플릿이 다양한 위치에서 반복적으로 적용되면서, 동일한 공격 시나리오가 중복 표현된다. 이는 불필요한 공격 단계가 트리에 포함되어 가독성을 저하시킨다.

이러한 한계는 방어 요소를 포함한 공격-방어 트리 생성 방법론을 제시한 후속 연구 [14]에서도 해결되지 못하였다. 한편, G. He et al.[15]은 CPS (Cyber-Physical Systems) 아키텍처를 SysML로 모델링하고 이를 기반으로 공격-방어 트리를 자동 생성하는 기법을 제안하였다. 그러나 이 연구는 공격 목표 달성을 위해 요구되는 논리적 조건들의 의존 관계를 나열하는 데 집중하여 공격자가 네트워크 내에서 실제로 어떤 장비들을 거쳐 이동하는지에 대한 구체적인 전파 경로를 알기 어렵다는 한계가 있다.

본 연구에서는 J. Bryans et al.[10]이 제안한 도구의 이러한 문제점을 해결하고자 한다. 해당 문제에 대한 구체적인 분석은 3장에서, 본 연구가 제안하는 개선 방안은 4장에서 다룬다.

III. 템플릿 기반 공격 트리 자동 생성 도구의 한계

본 절에서는 관련 연구에서 소개한 J. Bryans et al.[14]의 템플릿 기반 공격 트리 생성 도구의 구성 요소와 동작 절차를 상세히 기술하고, 한계를 분석한다.

해당 도구는 시스템 모델과 공격 템플릿 라이브러리를 입력으로 받아 동작한다. 시스템 모델은 차량 도메인의 ECU(Electronic Control Unit), 네트워크(NET, Network), 진입 지점 (AP, Access Point)을 컴포넌트 타입으로 정의하며, 라이브러리는 여러 템플릿 파일로 구성된다. 해당 논문의 공격 트리는 OR, AND, SAND의 3가지 연산자로 각 노드를 연결하고, 각 노드 라벨에는 공격 방법과 공격 대상이 '공격 방법(공격 대상)'의 형식으로 명시된다.

J. Bryans et al.[10]의 도구는 시스템 모델 파일과 템플릿에 맞게 사전 정의된 공격 라이브러리를 입력으로 받아 아래와 같은 단계를 거쳐 공격 트리를 생성한다. 먼저, 공격 라이브러리에서 사용자가 공격 목표로 지정한 하나의 템플릿 파일을 선택하여 트리의 루트 노드로 설정한다. 그다음 설정된 루트 노드의 자식 노드로 연결될 수 있는 공격이 라이브러리에 있는지 조사하고, 노드에 있는 변수를 시스템 모델에 기반하여 모든 후보 컴포넌트로 할당한다. 이 과정을 더 이상 확장할 템플릿이 없을 때까지 또는 진입 지점에 도달할 때까지 모든 리프에 반복하여 완성된 공격 트리를 얻는다. 이 방법은 (i) 모델-템플릿 매칭, (ii) 템플릿 재사용, (iii) 경로 탐색 바탕의 자동 전개를 통해 수작업 부담을 줄이면서 다양한 경로 분기를 합성할 수 있다.

그러나 해당 도구는 실제 구현된 알고리즘에서 두 가지 문제점을 가진다. 첫째, 실제로 가능한 공격 경로의 누락이 발생될 수 있다. 이는 도구에서 사용된 깊이 우선 탐색(DFS, Depth First Search) 기반 경로 탐색 로직의 설계적 한계에서 비롯된다. 구체적으로, 해당 로직은 특정 분기점에서 최초로 발견된 목표 도달 경로만을 유효 경로로 간주하여 기록하고, 해당 경로로부터 우회할 수 있는 다른 경로에 대한 탐색을 중단한다. 이로 인해, 시스템 내 존재할 수 있는 우회 경로가 탐색 과정에서 누락될 수 있다. 즉, 최단 경로 완성으로 인해 목적지를 발견했다는 내부 플래그가 활성화되면, 동일한 분기점에서 시작하는 다른 이웃 노드를 경유하는 우회 경로에 대한 전수조사가 이뤄지지 않는다. 이러한 설계는 공격자가 최종 목적지로 도달하기 위한 전체 경로를 모두 파악할 수 없게 만들며, 최단 경로만 방어하는 행위는 추후 보안 문제를 초래할 수 있다. 예를 들어, TCU(Transmission Control Unit)는 고신뢰

보안이 요구되는 핵심 부품으로, 보안 정책상 GW(Gateway)를 통해 유입되는 일반적인 외부 제어 명령은 무시하고, 오직 주행 안정성을 위해 긴밀하게 연결된 ESC(Electronic Stability Control)의 제어 명령만 신뢰하도록 설계되어 있다고 가정하자. 해당 시나리오에서는 공격자가 GW를 장악한 뒤, GW와 연결된 ESC를 추가로 감염시켜 제어권을 획득할 수 있다. 이후, 공격자는 신뢰받는 ESC인 척 위장하여 TCU에 악성 제어 명령을 보낼 수 있으며, TCU는 이를 정상적인 주행 안정화 요청으로 오인하여 수행하게 된다. 결과적으로 기존 방식은 이러한 실질적이고 치명적인 우회 공격 경로를 누락시키는 한계를 가진다. 그림 1은 J. Bryans et al.[10]가 제시한 템플릿 기반 자동 생성 도구를 통해 도출된 공격트리로, 앞서 언급한 경로 누락 한계를 보여준다[10].

두 번째로, 공격 트리 전개 과정에서 발생하는 공격 단계의 중복이다. 공격 트리 생성 도구는 리프 노드와 연결할 수 있는 공격 단계를 공격 라이브러리에서 탐색하고 리프 노드 아래에 이어 붙여서 공격 트리를 확장한다. 예를 들어, Compromise(TCU)라고 하는 공격 단계를 확장할 때 Compromise.xml 템플릿을 불러온다. 이후 기존 도구의 알고리즘은 공격 트리 리프 노드에 불러온 템플릿의 부모 노드인 Compromise(TCU)를 자식 노드로 추가한다. 그 결과 그림 1의 표시된 네모 박스에서 볼 수 있듯 동일한 공격 단계 노드가 부모-자식 관계를 가지는 수직적 중복이 발생한다. 본 예시는 간단한 예시지만 복잡하고 거대한 시스템을 분석하는 과정에서 이는 공격 트리의 가독성을 크게 저해한다.

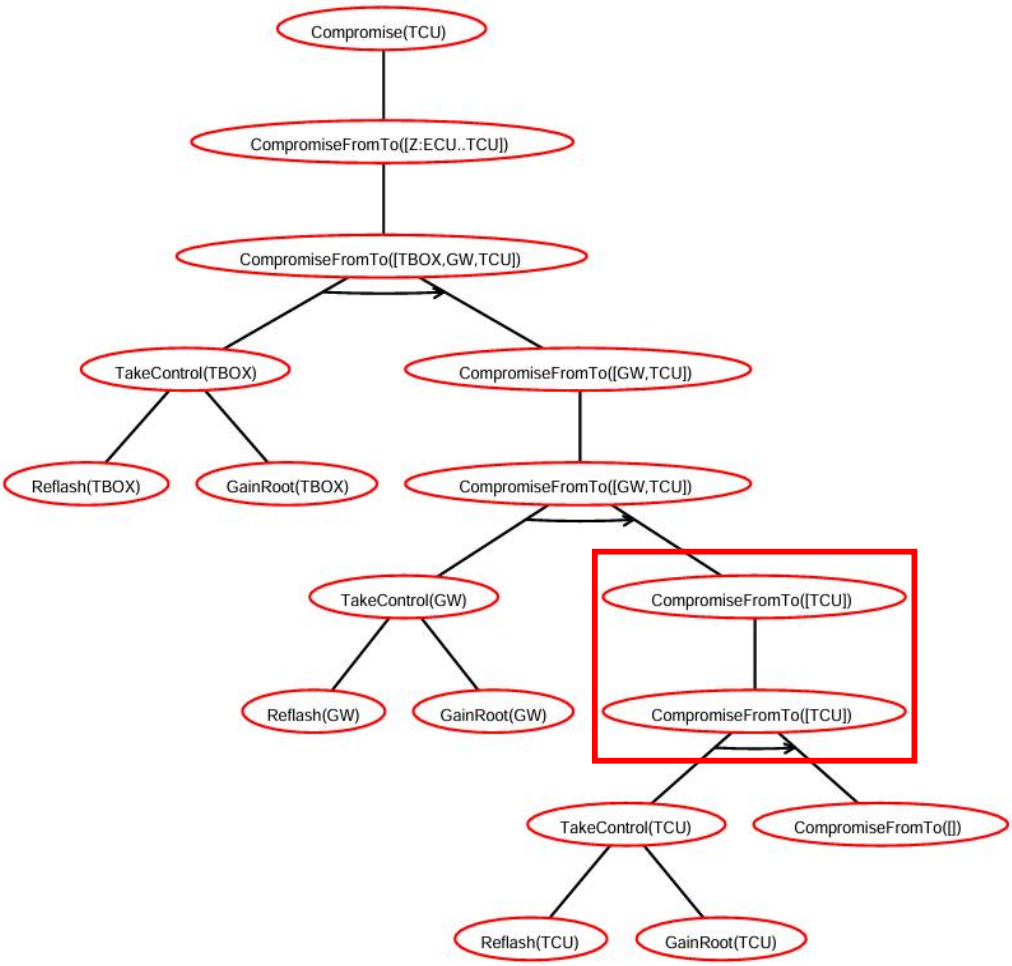


그림 1. 기존 도구로 생성된 공격 트리
Fig. 1. Output attack tree produced by the original template-based tool

우리는 J. Bryans et al.[10]의 도구에 대해 위와 같은 두 가지의 문제점들을 파악했다. 따라서 경로 탐색의 완전성과 트리 구조의 가독성을 확보하기 위한 보완이 필수적이며, 이후 4장에서는 이러한 문제점을 해결하기 위한 알고리즘 수준의 개선 방안을 제안한다.

IV. 보완점

본 장에서는 3장에서 분석한 2가지 문제점의 해결 방안을 설명한다. 상세한 해결 방안을 설명하기에 앞서, 표 1은 기존 도구의 문제점과 본 연구에서 제안하는 해결 방안을 요약한다.

4.1 경로 탐색 알고리즘의 완전성 확보

기존 도구는 특정 경로를 발견하면 탐색을 중단하는 방식의 DFS 알고리즘을 사용하여, 잠재적인 우회 공격 경로를 식별하지 못하는 조기 종료 문제를 지니고 있다. 본 연구에서는 이러한 경로 누락을 방지하기 위해 기존 알고리즘의 경로 발견 플래그와 우회 탐색 차단 조건문을 제거하였다. 개선된 알고리즘은 현재 위치한 컴포넌트의 이웃 노드를 탐색할 때, 최단 경로 탐색과 우회 경로 탐색을 독립적으로 수행한다. 최단 경로 탐색은 이웃 노드 중 최종 목적지가 존재할 경우, 즉시 결과 리스트에 추가하며, 우회 경로 탐색은 목적지가 아닌 노드를 경유하여 공격을 이어가는 모든 하위 경로를 지속적으로 추적한다. 이때, 네트워크 아키텍처의 구성과 복잡도가 상이하므로, 모든 시스템에 일률적인 탐색 깊이를 적용하는 것은 비효율적이다. 따라서 본 연구에서는 경유 횟수를 고정된 상수가 아닌 가변적인 파라미터로 설계하였다. 이를 통해 사용자는 대상 시스템의 규모와 보안 분석의 목적에 맞춰 탐색 깊이를 스스로 조정함으로써, 과도한 연산 없이 현실적인 공격 경로를 유연하게 도출할 수 있다.

4.2 조건부 템플릿 병합을 통한 수직적 중복 제거

두 번째로 J. Bryans et al.[10]의 도구는 템플릿 확장 과정에서 공격 단계의 중복이 발생하여 가독성이 저하되는 문제가 있다. 이는 동일한 공격 단계가 여러 경로에 나타나는 것이 아닌 한 경로에서 부모 노드와 자식 노드가 동일한 공격 단계를 나타내는 문제점을 말한다. 본 연구는 이 문제를 해결하기 위해 '조건부 템플릿 병합' 로직을 제안한다.

본 연구에서 말하는 '템플릿'은 각 공격 단계와 그에 대한 사전 조건(Pre-condition) 단계를 서브 트리 형태로 구성한 것이다. 공격 라이브러리는 이러한 템플릿이 여러 개 모인 집합을 의미한다. '조건부 템플릿 병합' 로직은 공격 트리 전개 과정에서 리프 노드를 이어 붙일 때, 다음 공격 단계 템플릿의 사전 조건, 즉, 부모 노드가 현재 전개 중인 공격 트리의 리프 노드와 동일하면 부모 노드를 추가하지 않고 템플릿의 자식 노드를 연결한다.

본 4장에서 제안한 아이디어를 알고리즘 단위로 비교하고 실제 코드에 반영하여 동일한 시스템 모델을 바탕으로 제안한 아이디어의 효과를 분석한 내용은 다음 5장에서 다룬다.

표 1. Bryans의 한계성 및 제안방안
Table 1. Limitations of Bryans and its solutions

Category	Limitations of the existing tool	Proposed method
Path completeness	• Missing feasible paths due to premature termination in DFS • Inability to capture insider-initiated attack scenarios	• Modified DFS algorithm • Support for internal attacker starting point
Redundancy & readability	• Repeated reapplication of templates leading to redundant nodes • Increased tree size and structural complexity	• Elimination of meaningless vertical duplication • Generation of a simplified and cleaner attack tree

V. 실험 및 검증

본 절에서는 4장에서 제안한 두 가지 보완 방안의 효과를 검증하기 위해, 기존 도구와 개선된 도구에 동일한 시스템 모델을 적용하여 생성된 공격 트리를 비교 분석한다. 분석 대상 시스템 모델은 J. Bryans et al.[10]이 검증에 활용한 차량 네트워크 모델을 동일하게 사용한다.

Bryans et al.[10]에서 가정한 차량 네트워크 모델은 그림 2와 같이 다수의 ECU로 구성되어 있다. 공격자의 진입 지점은 TBOX, OBD-II이며 공격의 최종 목표는 TCU이다.

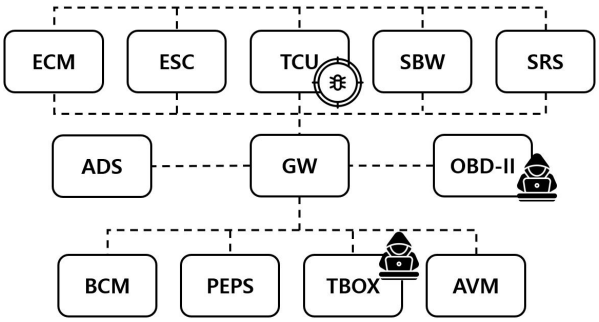


그림 2. 목표 시스템 모델 : 차량 네트워크
Fig. 2. Target system model: vehicle network

표 2는 기존 도구와 개선된 도구의 DFS 알고리즘을 각각 보여준다. 기존 도구의 알고리즘에서는 "if not found" 조건을 두어 최단 경로에 목적지가 있으면 우회 경로는 탐색하지 않는다. 반면 개선된 도구의 알고리즘에서는 해당 조건을 생략하여 최단 경로와 우회 경로를 모두 탐색한다. 또한, 경유 횟수를 가변적인 파라미터로 두어 사용자가 대상 시스템의 규모와 보안 분석의 목적에 맞춰 탐색 깊이를 직접 조정할 수 있게 한다.

또한 표 3은 기존 도구와 개선된 알고리즘을 적용한 도구 간 실행 결과의 차이를 보여준다. 이때, 본 실험에서는 현실적인 공격 가능성과 결과의 가독성을 고려하여 최단 경로 대비 최대 2회의 경유를 허용하는 조건으로 공격 트리를 생성하였다. 기존 도구에서는 TBOX → GW → TCU, OBD-II → GW → TCU의 경로 2개만 탐색되었다. 그러나 개선된 도구는 설정된 경유 횟수 범위 내에서 기존 대비 6.5배 증가한 총 13개의 공격 경로를 식별하였다. 구체적으로 앞선 최단 경로 2개 외에도, TBOX → BCM → GW → TCU와 같이 출발지 네트워크 내에서 인접 노드를 경유해 보안을 우회하거나, OBD-II → GW → ECM → TCU와 같이 목적지 네트워크 진입 후 타겟의 인접 ECU를 먼저 장악하는 등 동일 CAN 버스 내 수평 이동을 포함한 11개의 확장된 공격 경로가 추가로 탐색되었다. 이는 기존 도구가 네트워크 모델 내에서 발생 가능한 전체 공격 경로의 약 84.6%이상을 누락할 수 있다는 한계를 정량적으로 보여준다. 실제로 ESC와 같은 섀시 제어 시스템은 실시간성이 매우 강조되므로, 내부 통신 시 인증 절차를 생략하고 TCU와 데이터를 교환하는 경우가 빈번하다[16]. 반면, TCU는 게이트웨이를 통해 유입되는 외부 메시지에 대해서는 화이트리스트 기반의 필터링을 수행하여 직접적인 공격을 차단하도록 설계된다[17][18]. 따라서, 무조건 최단 경로를 식별하는 것이 아닌 TBOX → GW → BCM → TCU와 같은 우회 경로를 포함한 모든 공격 경로의 탐색이 필요하며, 개선된 도구가 경유 횟수 조정을 통해 이러한 한계를 보완할 수 있음을 확인하였다.

다음으로 수직적 중복된 공격 단계와 관련한 알고리즘의 변경점과 이에 따른 공격 트리 자동 생성 결과의 차이를 살펴본다. 표 4는 기존 도구와 개선된 도구의 공격 트리 전개 방법을 각각 알고리즘 수준으로 보여준다. 기존 도구에서는 공격 템플릿이 적용될 수 있는 대상이 한 개뿐이어도 템플릿 전체를 반복적으로 추가한다. 반면, 개선된 도구에서는 공격 템플릿이 적용될 수 있는 대상이 한 개일 경우, 템플릿을 그대로 추가하지 않고, 리프 노드와 템플릿을 병합하여 중복을 방지한다.

표 5는 공격 트리 전개 알고리즘이 각각 적용된 도구의 실행 결과를 보여준다. 기존 도구에서는 동일 노드가 중복해서 전개되는 것을 볼 수 있다. 반면, 개선된 도구에서는 동일 노드가 중복해서 전개되지 않고 바로 자식 노드로 전개되는 것을 확인할 수 있다. 결과적으로 기존 도구를 개선한 도구에서는 놓칠 수 있는 우회 경로를 보여주면서도 노드 간의 중복은 없애 가독성을 높일 수 있었다.

표 2. 개선된 DFS 알고리즘
Table 2. Improved DFS algorithm

Existing DFS algorithm[10]	Proposed DFS algorithm

```
Input: path, finals, model
Output: list of paths
r ← [];
nexts ← getNexts(path, model);
found ← false;
foreach n in nexts do
    foreach m in finals do
        if n.name = m.name then
            found ← true;
            path1 ← path + [n];
            r.append(path1);
        end if
    end foreach
end foreach
if not found then
    foreach n in nexts do
        cans ← { c | c ∈ keys(model[0]), n ∈ model[0][c].children };
        if |cans| > 1 then
            path1 ← path + [n];
            r.extend(dfs(path1, finals, model));
        end if
    end foreach
end if
return r;
```

```
Input  : path, finals, model
Output: list of paths
1  MAX_HOPS ← 2 ;                          // User Configuration
2  r ← [];
3  nexts ← getNexts(path, model);
4  finalNames ← {m.name | m ∈ finals};
   // 1.  Check direct paths
5  foreach n ∈ nexts do
6      if n.name ∈ finalNames then
7          path1 ← path + [n];
8          r.append(path1);
9      end
10 end
   // 2.  Check indirect paths with hop limit
11 currentHops ← length(path) − 1;
12 if currentHops < MAX_HOPS then
13     foreach n ∈ nexts do
14         if n.name ∉ finalNames then
15             path1 ← path + [n];
16             r.extend(dfs(path1, finals, model));
17         end
18     end
19 end
20 return r;
```

표 3. 공격 경로 완전성 비교
Table 3. Comparison of attack path completeness

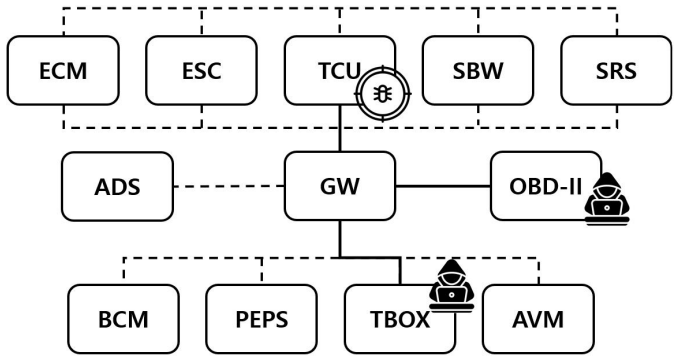
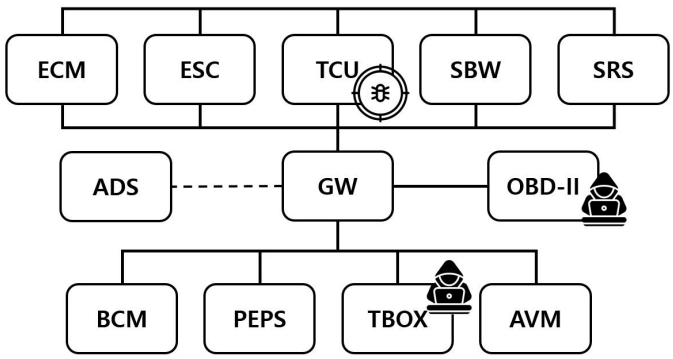
Existing tool[10]	
Proposed tool	

표 4. getAT 알고리즘에서 중복공격 단계 개선
Table 4. Improving duplicated attack step occuring in getAT algorithm

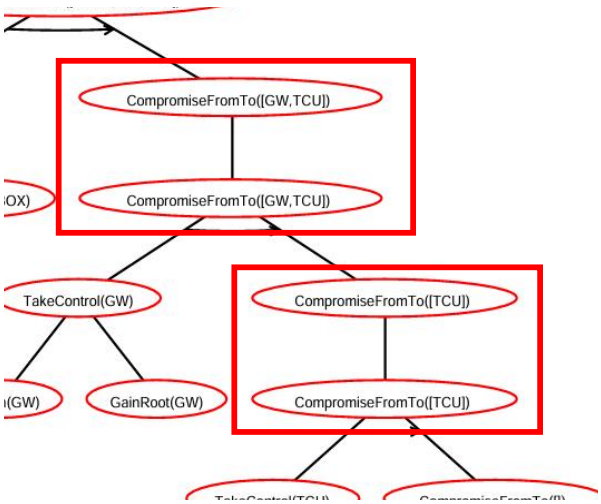
Existing attack tree generation algorithm[10]	Proposed attack tree generation algorithm

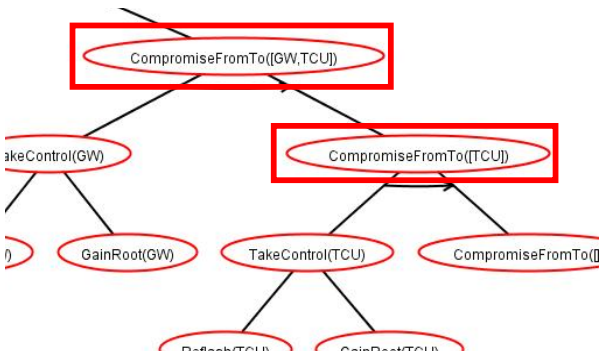
```
Input: tree, model, library
while true do
    leaves ← tree.getLeaves();
    expanded ← false;
    foreach leaf in leaves do
        leaf.type ← OR;
        assignments ← findAssignments(leaf, model);
        if assignments = [] then
            assignments ← [[]]
        end if
        foreach a in assignments do
            (T, sub) ← findTree(leaf.apply(a), library);
            if T ≠ None then
                expanded ← true;
                leaf.children.append(T.copy().apply(sub));
            end if
        end foreach
    end foreach
    if not expanded then
        break
    end if
end while
```

```
Input: tree, model, library
while true do
    leaves ← tree.getLeaves();
    expanded ← false;
    foreach leaf in leaves do
        assignments ← findAssignments(leaf, model);
        if assignments = [] then
            assignments ← [[]]
        end if
        if |assignments| > 1 then
            leaf.type ← OR
        end if
        foreach a in assignments do
            (T, sub) ← findTree(leaf.apply(a), library);
            if T ≠ None then
                expanded ← true;
                newT ← T.copy().apply(sub);
                if |assignments| > 1 then
                    leaf.children.append(newT);
                end if
                else
                    leaf.type ← newT.type;
                    leaf.children.extend(newT.children);
                end if
            end if
        end foreach
    end foreach
    if not expanded then
        break
    end if
end while
```

표 5. 중복단계표현 비교
Table 5. Comparison of duplicate step representation

Existing tool[10]	Proposed tool





본 연구에서 제안한 두 가지 알고리즘 개선 방안을 모두 적용한 최종 공격 트리[19]는 중복된 공격 단계를 제거함과 동시에 최단 경로와 우회 경로를 모두 포함하여, 제안한 방법의 개선 효과를 종합적으로 입증한다.

Ⅵ. 결론 및 향후 과제

현대 IT 환경은 다양한 도메인의 시스템이 복잡하게 연결된 구조로, 체계적인 위협 분석 방법론이 필요하다. 공격 트리는 시스템의 보안 위협을 체계적으로 분석하는 핵심 도구이지만, 수동 작성 시 많은 시간이 소요되고 오류가 쉽게 발생한다. 이러한 문제를 해결하기 위해 자동화된 공격 트리 생성 방법이 연구자들의 주목을 받고 있다.

본 연구에서는 J. Bryans et al.[10]의 템플릿 기반 공격 트리 자동 생성 도구를 보완하는 두 가지 방안을 제안하였다. 첫째, 경로 누락 가능성을 해결하기 위해 DFS 알고리즘에서 우회 경로 탐색을 차단하던 조건문을 제거하였다. 둘째, 동일한 노드가 구조적으로 반복되는 문제를 해결하기 위해 중복 제거 로직을 도입하였다. 두 가지 보완 방안을 실제로 구현하고 기존 도구와 비교 평가한 결과, 더욱 상세하고 가독성이 개선된 공격 트리를 생성할 수 있음을 확인하였다.

다만, 기존 도구는 차량 도메인을 중심으로 정의된 자산 구조와 공격 패턴에 강하게 종속되어 있어 다양한 도메인에 그대로 적용하기 어렵다는 한계가 존재한다. 이러한 도메인 종속성은 실제 산업 시스템 전반에 도구를 확장하는 데 제약으로 작용하며, 다양한 형태의 시스템 아키텍처를 지원하기 위해서는 알고리즘뿐 아니라 템플릿과 모델링 체계 전체의 개선이 필요하다. 향후 연구에서는 이러한 도메인 종속성을 완화하기 위한 템플릿 및 모델링 구조의 일반화를 단계적으로 진행할 예정이다.

향후 연구는 공격 트리 자동 생성의 적용 범위와 신뢰성을 강화하는 방향으로 진행될 예정이다. 첫째, 현재 도구의 공격 라이브러리는 제한된 자산 유형만을 포함하고 있어 복잡한 실제 시스템을 온전히 반영하기 어렵다. 따라서 우주, 로봇 등 다양한 분야의 자산과 프로토콜이 혼재하는 시스템을 분석할 수 있도록 공격 패턴의 표현 범위를 점진적으로 확장할 예정이다.

둘째, 시스템 규모가 커지고 구성 요소가 복합적으로 연결될수록 공격 트리의 크기가 기하급수적으로 증가할 수 있다. 이러한 문제는 다수의 자동 생성 연구에서 공통적으로 보고되는 한계로, 제안 알고리즘이 다양한 구조에서도 일관되게 동작하는지 지속적인 검증이 요구된다. 본 연구에서는 대규모 또는 비선형 구조의 모델을 대상으로 한 성능 평가 및 보완 방안을 단계적으로 마련해 나갈 계획이다.

마지막으로, 위성 및 지상국은 무선 통신과 네트워크 제어가 결합된 대표적인 복합 시스템으로, 향후 실험 환경으로서 중요한 사례가 될 것이다. 해당 도메인에서의 추가 평가를 통해 제안 기법의 실용성을 입증하고, 장기적으로는 다양한 산업 시스템에서 자동 위협 분석을 지원할 수 있는 확장 가능한 프레임워크로 발전시키는 것을 목표로 한다.

References

[1] S. S. Gill, et al., "Modern computing: Vision and challenges", Telematics and Informatics Reports, Vol. 13, pp. 100116, pp. 1-38, Mar. 2024. https://doi.org/10.1016/j.teler.2024.100116.

[2] A. M. Khasawneh, et al., "Service-centric heterogeneous vehicular network modeling for connected traffic environments", Sensors, Vol. 22, No. 3, pp. 1-19, Feb. 2022. https://doi.org/10.3390/s22031247.

[3] A. F. A. Musawi, S. Roy, and P. Ghosh, "Examining indicators of complex network vulnerability across diverse attack scenarios", Scientific Reports, Vol. 13, No. 18208, pp. 1-14, Oct. 2023. https://doi.org/10.1038/s41598-023-45218-9.

[4] B. Schneier, "Attack Trees", Dr. Dobb's Journal, Dec. 1999. https://www.schneier.com/academic/archives/1999/12/attack_trees.html.

[5] K.-A. Kim, D.-S. Lee, and K.-N. Kim, "ICS Security Risk Analysis Using Attack Tree",

Convergence Security Journal, Vol. 11, No. 6, pp. 53-58, Dec. 2011.

[6] R. Kumar, R. Kela, S. Singh, and R. Trujillo-Rasua, "APT attacks on industrial control systems: A tale of three incidents", International Journal of Critical Infrastructure Protection, Vol. 37, No. 100521, pp. 1-11, Jul. 2022. https://doi.org/10.1016/j.ijcip.2022.100521.

[7] C. E. Alozie, "Threat Modeling in Health Care Sector", International Journal of Engineering and Modern Technology, Vol. 10, No. 11, pp. 199-207, 2024.

[8] P. Karpati, Y. Redda, A. L. Opdahl, and G. Sindre, "Comparing attack trees and misuse cases in an industrial setting", Information and Software Technology, Vol. 56, No. 3, pp. 294-308, Mar. 2014. https://doi.org/10.1016/j.infsof.2013.10.004.

[9] S. Pinchinat, M. Acher, and D. Vojtisek, "ATSyRA: An Integrated Environment for Synthesizing Attack Trees", Proc. of Graphical Models for Security (GraMSec 2015), Verona, Italy,, Vol. 9390, pp. 97-101, Jul. 2015. https://doi.org/10.1007/978-3-319-29968-6_7.

[10] J. Bryans, et al., "A Template-Based Method for the Generation of Attack Trees", Information Security Theory and Practice (WISTP 2019), Paris, France, Vol. 12024, pp. 155-165, Dec. 2019. https://doi.org/10.1007/978-3-030-41702-4_10.

[11] R. Vigo, F. Nielson, and H. R. Nielson, "Automated Generation of Attack Trees", Proc. of the 27th IEEE Computer Security Foundations Symposium, Vienna, Austria, pp. 337-350, Jul. 2014. https://doi.org/10.1109/CSF.2014.31.

[12] M. G. Ivanova, C. W. Probst, R. R. Hansen, and F. Kammüller, "Transforming graphical system models to graphical attack models", Proc. of the Second International Workshop on Graphical Models for Security (GraMSec 2015), Verona, Italy, pp. 82-96, Jul. 2015. https://doi.org/10.1007/978-3-319-29968-6_6.

[13] R. Jhawar, K. Lounis, S. Mauw, and Y. Ramírez-Cruz, "Semi-automatically augmenting attack trees using an annotated attack tree library", Proc. of the International Workshop on Security and Trust Management, Lecture Notes in Computer Science, Barcelona, Spain, Vol. 11091, pp. 85-101, Sep. 2018. https://doi.org/10.1007/978-3-030-01141-3_6.

[14] J. Bryans, et al, "Formal template-based generation of attack-defence trees for automated security analysis", Information, Vol. 14, No. 9, pp. 1-25, Aug. 2023. https://doi.org/10.3390/info14090481.

[15] G. He, B. Xu, J. Zhao, and Y. Shi, "Architecture-oriented security strategy determination for cyber-physical systems", Concurrency and Computation: Practice and Experience, Vol. 35, No. 13, pp. e6781, Jun. 2023. https://doi.org/10.1002/cpe.6781.

[16] K. Koscher, et al., "Experimental security analysis of a modern automobile", Proc. of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, pp. 447-462, May 2010. https://doi.org/10.1109/SP.2010.34.

[17] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle", Black Hat USA, pp. 1-91, 2015.

[18] M. Wolf, A. Weimerskirch, and C. Paar, "Security in automotive bus systems", Workshop on Embedded Security in Cars, pp. 1-13, Nov. 2004.

[19] https://github.com/JJeong0627/attack-tree-pdf/blob/main/output.pdf.

저자소개

최 유 정 (Yujeong Choi)

2024년 2월 : 성신여자대학교 융합보안공학과(학사)
2024 3월 ~ 현재 : 고려대학교 정보보호대학원 석사과정
관심분야 : 보안공학, RMF A&A

조 광 수 (Kwangsoo Cho)

2019년 2월 : 호서대학교 컴퓨터공학과(학사)
2021년 8월 : 고려대학교 정보보호대학원(석사)
2021년 9월 ~ 현재 : 고려대학교 정보보호대학원 박사과정
관심분야 : 보안공학, RMF, AI, 임베디드 시스템

곽 지 원 (Jiwon Kwak)

2017년 2월 : 중앙대학교 전자전기공학부(학사)
2019년 8월 : 고려대학교 일반대학원 사이버국방학과(석사)
2019년 9월 ~ 현재 : 고려대학교 정보보호대학원 박사과정
관심분야 : 위협모델링, 정형기법, 고신뢰 시스템 개발

김 승 주 (Seungjoo Kim)

1999년 2월 : 성균관대학교 정보공학과(박사)
1998년 12월 ~ 2004년 2월 : 한국인터넷진흥원(KISA) 팀장
2004년 3월 ~ 2011년 2월 : 성균관대학교 정보통신공학부 부교수
2011년 3월 ~현재 : 고려대학교 정보보호대학원 정교수
2005년 1월 ~현재 : 한국정보보호학회 이사
2014년 9월 ~ 2015년 3월 : 육군사관학교 초빙교수
2016년 1월 ~ 2018년 1월 : 개인정보분쟁조정위원회 위원
2018년 11월 ~ 2020년 3월 : 대통령직속 4차산업혁명위원회 위원
2022년 7월 ~ 2024년 6월 : 고려대학교 스마트모빌리티학부 학부장
2023년 5월 ~ 2025년 5월 : 대통력직속 국방혁신위원회 위원
2024년 9월 ~ 2025년 7월 : 대통령직속 국가인공지능위원회 위원
2023년 3월 ~ 현재 : 고려대학교 디지털정보처 처장
2023년 9월 ~ 현재 : (사)국방혁신기술보안협회 협회장
2024년 2월 ~ 현재 : 중앙선거관리위원회 보안자문위원회 위원장
2025년 9월 ~ 현재 : 대통령직속 국가인공지능전략위원회 자문위원
관심분야 : 위협모델링, 보안성 평가/인증, DevSecOps, 암호학, 블록체인, Decentralized AI