情報保護學會論文誌
第18卷 第3號, 2008. 6

의미기반 취약점 식별자 부여 기법을 사용한 취약점 점검 및 공격 탐지 규칙 통합 방법 연구*

김 형 종[1†‡], 정 태 인[2]
[1]서울여자대학교, [2]한국정보보호진흥원

A Study for Rule Integration in Vulnerability Assessment and Intrusion Detection using Meaning Based Vulnerability Identification Method*

Hyung-Jong Kim[1†‡], Taeln Jung[2]
[1]Seoul Women's University, [2]Korea Information Security Agency

요 약

본 논문은 소프트웨어의 취약점을 표현하기 위한 방법으로 단위 취약점을 기반으로 한 의미기반 취약점 식별자 부여 방법을 제안 하고 있다. 의미기반 취약점 식별자 부여를 위해 기존의 취약점 단위를 DEVS 모델링 방법론의 SES 이론에서 사용되는 분할 및 분류(Decomposition/Specialization) 절차를 적용하였다. 의미기반 취약점 식별자는 취약점 점검 규칙 및 공격 탐지 규칙과 연관 관계를 좀 더 낮은 레벨에서 맺을 수 있도록 해주고, 보안 관리자의 취약점에 대한 대응을 좀더 편리하고 신속하게 하는 데 활용될 수 있다. 특히, 본 논문에서는 Nessus와 Snort의 규칙들이 의미기반 취약점 식별자와 어떻게 맵핑되는 지를 제시하고, 보안 관리자 입장에서 어떻게 활용 될 수 있는 지를 3가지 관점에서 정리하였다. 본 논문의 기여점은 의미기반 취약점 식별자 개념 정의 및 이를 기반으로 한 취약점 표현과 활용 방법의 제안에 있다.

ABSTRACT

This paper presents vulnerability identification method based on meaning which is making use of the concept of atomic vulnerability. Also, we are making use of decomposition and specialization processes which were used in DEVS/SES to get identifiers. This vulnerability representation method is useful for managing and removing vulnerability in organized way. It is helpful to make a relation between vulnerability assessing and intrusion detection rules in lower level. The relation enables security manager to response more quickly and conveniently. Especially, this paper shows a mapping between Nessus plugins and Snort rules using meaning based vulnerability identification method and lists usages based on three goals that security officer keeps in mind about vulnerability. The contribution of this work is in suggestion of meaning based vulnerability identification method and showing the cases of its usage for the rule integration of vulnerability assessment and intrusion detection.

Keywords : *Meaning-based Vulnerability Identification, Vulnerability Assessment, Intrusion Detection, DEVS-formalism*

접수일 : 2007년 11월 19일; 채택일 : 2008년 1월 15일
* 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장 동력핵심기술개발사업의 일환으로 수행하였음.
[2005-S-606-02, 차세대 침해사고 예측 및 대응 기술]
† ‡ 주저자, 교신저자, hkim@swu.ac.kr

Ⅰ. 서 론

인터넷 서비스가 가져야 하는 가장 중요한 특성 중 하나는 정상적인 사용자에게 언제든지 서비스에 접근할 수 있는 권한을 제공해야 한다는 것이다. 이러한 서비스에 대한 가용성의 보장은 서비스를 통해 수익을 창출하는 기업이나, 공공의 이익을 위하는 국가 기관에게 매우 중요한 고려사항이 되어 있다. 서비스에 대한 가용성의 보장에 있어서 중요하게 고려되는 두 가지 요소는 다음과 같다. 첫째, 동시 접속 사용자에 대한 예측으로, 해당 서비스에 같은 시간에 접속할 사용자의 숫자를 예측하고 이를 수용할 수 있는 수준의 서비스 제공 환경을 구축하는 것이다. 이를 위한 대책이 될 수 있는 것이 여러 서버의 복제를 통해 서비스에 걸리는 부하를 분산하는 방법이 있을 수 있다. 둘째, 서버에 대한 악의적인 공격에 대한 철저한 대비를 하는 것으로, 해당 서비스가 제공되기 위해서 존재하는 모든 하드웨어 혹은 소프트웨어에 존재하는 취약점을 제거하고, 제거하기 어려운 취약점에 대해서는 이를 공격자가 악용하지 못하도록 시스템을 적절히 보호하는 것이다. 이러한 두 가지 접근 방법은 네트워크 및 시스템을 보호하기 위한 기본적인 고려사항들이며, 제공하고자 하는 서비스를 준비하는 시점부터 고려되어야하는 사항들이다. 또한, 이 두 가지 고려사항은 어느 한쪽만 이루어져서는 안 되고, 동시에 이루어져야 시스템/네트워크의 신뢰성을 보장할 수 있다.

본 연구는 이러한 인터넷 서비스의 보호를 위한 접근 중 두 번째에 해당하는 시스템 및 네트워크의 악의적 행위에서부터 보호하고자 하는 측면에서의 관리에 초점을 맞추고, 특히 소프트웨어의 취약점을 악용하는 공격자의 행위에 대한 대응 방법을 고려하였다. 특히, 취약점을 발견하고, 이를 효과적으로 관리하기 위해서는 취약점에 대한 면밀한 분석이 필요하여 이를 기반으로 한 관리 정책이 정의되어 있어야 한다. 본 연구의 핵심은 취약점을 표현하는 방법에 있어서 기존의 단순 아이디 부여 방법을 보완하여, 소프트웨어의 취약점을 효과적으로 표현할 수 있는 의미기반 아이디 부여 방법에 대한 연구에 있다. 이러한 의미기반의 아이디 부여로 인해서 가져올 수 있는 몇 가지 효과 중 본 연구에서는 기존의 취약점 및 공격 기법을 탐지하기 위한 점검 규칙의 실질적인 점검 대상을 표현하는 영역에서의 효과를 제시하고자 한다. 특히, 본 연구에서는 공개용 취약점 점검 도구인 Nessus의 취약점 점검 규칙[12]과 공개용 침탐지 시스템인 Snort가 갖는 공격 탐지 규칙[11]에 이러한 의미기반의 취약점 아이디 부여 방법이 어떤 효과를 가져올 수 있는지를 실질적인 점검 규칙을 기반으로 제시하였다.

본 논문의 구성은 취약점의 표현 방법과 관련된 기존 연구에 대해서 분석하고 의미기반 아이디 부여 방법의 근간이 되는 단위 취약점(Atomic Vulnerability)에 대해서 소개하고자 한다. 또한, 본 연구의 핵심이 되는 단위 취약점의 개념을 통해서 도출되는 의미 기반의 취약점 아이디 부여 방법을 소개한다. 이러한 의미 기반의 취약점 아이디 부여 방법은 4장에서 그 활용의 측면에서 Nessus와 Snort의 탐지 규칙에 대해서 적용을 해보고 이의 효과를 논의하고자 한다.

Ⅱ. 기존 이론

단위 취약점의 표현에 근간이 되는 모델링 방법론인 DEVS(Discrete EVent System Specification) 형식론[1,2,3]은 B. P. Zeigler에 의해서 정의된 이산 사건 시뮬레이션 이론이다. DEVS 형식론은 시스템을 분석하여 모델을 만들기 위한 이론적 근간을 제공해준다. DEVS 형식론은 더 이상 나눌 수 없는 모델에 해당하는 atomic 모델을 정의하고 있고, 두개이상의 atomic 모델, atomic 모델과 coupled 모델 및 coupled 모델과 coupled 모델의 조합으로 구성될 수 있는 coupled 모델로 구성된다. 다음은 atomic 모델의 정의 이다.

$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$

단, X : 입력 사건의 집합.

S : 순차적 상태의 집합.

Y : 출력 사건의 집합.

$\delta_{int} : S \rightarrow S$: 내부 전이 함수

$\delta_{ext} : Q \times X \rightarrow S$: 외부 전이 함수

$\lambda : S \rightarrow Y$: 출력 함수

$t_a : S \rightarrow R+0 \rightarrow \infty$: 시간 진행 함수

단, $Q = \{(s,e) \mid s \in S, 0 \le e \le t_a(s)\}$

e : 최근의 상태 전이 이후로 흐른 시간.

다음은 coupled 모델의 정의이다.

$DN = < D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, select>$

단, D : coupled 모델의 구성요소 모델 i의 이름 집합.

M_i : 구성요소가 되는 basic 모델.

I_i : 모델 i의 영향을 받는 모델들의 집합.

Z_{ij} : I_i의 원소 각 j에 대해 i에서 j로 출력 번역 함수.

Select : 타이 브레이킹 함수.

Atomic 모델과 Coupled 모델을 통해 대부분의 시스템의 특성의 표현이 가능하다. 그러나, 시스템의 구조적인 특성을 표현하는데 있어서 Coupled 모델이 갖는 한계가 존재한다. 이것은 시뮬레이션 환경이기 때문에 존재하는 문제로서, 시스템을 구성하는 컴포넌트가 될 수 있는 후보 모델에 대한 정의와 이들 중 시뮬레이션을 수행하기 위한 선택이 가능한 구성이 필요하게 된다. 이를 위해서 정의된 모델링 이론이 SES/MB (System Entity Structure/Model Base)[1,2,3]이다. SES는 시스템의 구조를 나타내는 지식을 특정 형식으로 표현한 것으로써 모델들의 분할(decomposition), 분류(taxonomy)와 모델 간의 연결 관계에 대한 정보를 가지고 있다. SES는 트리 형태로 계층적인 모델들을 정의하고, 트리의 말단은 모델 베이스의 모델로 나타낸다. SES안에는 시스템의 구조를 표현하기 위한 지식으로 entity, aspect (decomposition), specialization의 3가지 형태의 노드가 있다.

취약점의 이름을 정하는 가장 대표적인 기법은 CVE (Common Vulnerability Exposures)에서 정의한 방법으로 단순히 취약점이 발견된 년도와 시점으로 고려해서 순차적으로 번호를 붙이는 형태로 정의된다.

CVE-ID는 취약점을 식별하는데 있어서 혼란이 생길 때 나타나는 문제를 해결하기 위해서 동일한 이름을 사용하도록 하기 위한 매우 간단한 명명법 이다. CVE-ID는 CVE라는 식별자 다음에 발견 연도와 일련번호를 붙여 관리를 하되, 발견 초기에는 CAN (Candidate)을 CVE 대신에 사용하고, 일정 수준이상 취약점이 일반적으로 알려진 후 CVE로 이름을 갱신해 준다. 2008년 5월 현재 약 31,000 여 개의 취약점이 존재하며, 매년 약 4~6,000개의 취약점이 NIST를 통해서 발표 되고 있다. 또한, NIST는 NVD (National Vulnerability Database)라고 하는 취약점 데이터베이스를 관리하며, 실시간으로 갱신되는 정보를 사용할 수 있도록 xml형태의 서비스를 제공해 주고 있다.

정보보호 제품의 개발 및 판매에 있어서 중요하게 고려되는 것이 해당 제품이 가지고 있는 적용범위(coverage)라고 할 수 있는데, 취약점은 정보보호 제품이 갖는 적용범위를 표시하는 데에 매우 좋은 지표라고 할 수 있다. 즉, 침입 탐지 시스템의 경우, 해당 기술이 탐지하는 공격의 영역을 표시하기 위해서 자주, 취약점을 그 지표로 사용하는데 이는 대부분의 공격이 특정 취약점을 악용하는 형태로 진행되기 때문이다[11]. 또한, 취약점 점검도구의 경우 해당 도구가 가지고 있는 적용범위를 표시할 때 관리대상이 되는 취약점의 범위를 표시하게 된다[12]. CVE-ID는 이러한 보안 제품이 갖는 탐지 영역을 표시하기 위한 공통적 기준으로 활용되고 있다.

단위 취약점은 특정 취약점의 더 이상 나눠 질수 없는 특성을 정의하기 위한 방법이다. 취약점은 단위 취약점과 이들 사이의 관계로 표현 된다. 다음은 단위 취약점 AV의 정의이다[4].

Atomic Vulnerability : AV = {Iav, Qav, δav, Type}
where,
Iav = {Iav1, Iav2, ..., Iavn}
Qav = Q(initial state) ∪Q(final state)
δav : Iav × Q(initial state) →Q(final state)
Type : {Fact, Deterministic, Non-deterministic}

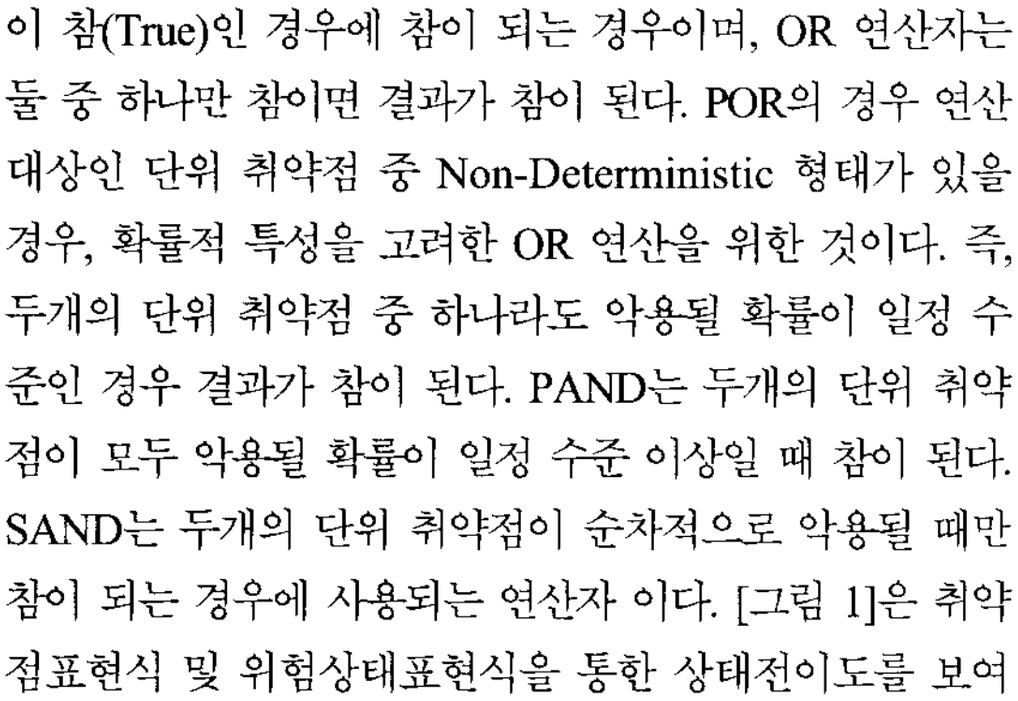
여기서, 입력 Iav에 의해서 시스템이 갖는 취약 특성이 활성화 되는 것이다. 이는 시스템의 취약점이 공격자의 특정 입력에 의해서 악용되는 상황을 모델링한 것이다. 일반적으로 취약점은 시스템의 잠재적 특성으로 공격자의 특정 입력을 통해서 활성화된다. 상태 집합인 Qav는 특정 입력이 시스템에 도착한 경우 전이하게 될 시스템의 상태를 정의하고 있다. 상태전이는 외부 입력 Iav와 상태집합 Qav로 정의된다. 단위 취약점은 3가지 유형으로 분류된다 첫째, 외부의 어떤 입력도 받지 않는 Fact 형태, 둘째, 외부의 입력을 받아 상태 변화가 생기는 경우는 확률적으로 상태변화가 발생하는 Non-deterministic 형태와 셋째, 외부의 입력에 의해 반드시 상태변화가 발생하는 Deterministic 형태이다. 이러한, 단위 취약점의 유형은 취약점을 분석하면서 도출된 것으로 대부분의 취약점의 특성을 포함 할 수 있다[4].

위에서 정의된 단위 취약점들과 이들 사이의 관계를 사용하여 취약점을 표현한다. 특히, 취약점 사이의 관계를 정의하기 위해서 취약점 표현식(VX : Vulnerability Expression)을 정의한다. 공격자의 행위는 결국 취약점을 악용하는 것으로 취약점의 악용이 성공할 경우 이를 참(True)으로 출력을 내주는 논리식을 취약점 표현식으

로 정의 하였다. 또한, 취약점 표현식은 공격자의 외부 입력에 대한 시스템의 상태 전이의 결정식으로 활용되어, 공격에 대한 시스템의 동적인 특성를 표현해 주게 된다. 취약점표현식과 함께 위험상태표현식(WSX : Warning State Expression)을 정의하여, 취약점이 악용되기 직전의 상태를 표현해 주도록 하였다. 아래는 단위 취약점을 사용한 취약점 표현식 및 위험상태 표현식에 대한 정의이다.

Vulnerability : V = {Icv, Qcv, δcv, WSX, VX}
where,
Icv = {Icv1, Icv2, …, Icvn}
Qcv= {Normal, Intermediate, Warning, Consequence}
δcv : Icv × Qcv → Qcv
WSX : Warning State Expression
VX : Vulnerability Expression

취약점 표현식은 취약점을 구성하는 단위 취약점들을 이진 연산자를 사용하여 관계를 표현하는 방법으로 정의된다. 여기서 사용되는 이진 연산자는 AND, OR, POR, PAND, SAND 이다. AND 연산자는 두개의 단위 취약점이 참(True)인 경우에 참이 되는 경우이며, OR 연산자는 둘 중 하나만 참이면 결과가 참이 된다. POR의 경우 연산 대상인 단위 취약점 중 Non-Deterministic 형태가 있을 경우, 확률적 특성을 고려한 OR 연산을 위한 것이다. 즉, 두개의 단위 취약점 중 하나라도 악용될 확률이 일정 수준인 경우 결과가 참이 된다. PAND는 두개의 단위 취약점이 모두 악용될 확률이 일정 수준 이상일 때 참이 된다. SAND는 두개의 단위 취약점이 순차적으로 악용될 때만 참이 되는 경우에 사용되는 연산자 이다. [그림 1]은 취약점표현식 및 위험상태표현식을 통한 상태전이도를 보여주고 있다.



〔그림 1〕 취약점의 상태 및 상태 전이

Ⅲ. 의미기반 취약점 식별자 부여방법

본 연구에서 제안하는 의미기반 취약점 아이디 부여 방법은 단위 취약점의 정의를 활용하는 것이다. 단위 취약점은 취약점을 표현하는 최소 단위 이며, 이를 재활용하여 비슷한 특성을 갖는 또 다른 취약점을 표현하는 데에 활용될 수 있다. [그림 2]의 CVE-2002- 0364는 버퍼 넘침 취약점을 단위 취약점을 활용하여 표현한 예이다.

단위 취약점을 사용하여 분석한 결과는 취약점 표현식으로 나타나며, 표현식의 각 단위 취약점들은 식별자(Identifier) 형태로 나타나고, 각각에 대해서 의미가 부여된다. CVE-2002-0364 표현식은 3개의 단위 취약점(AI1-202, KM1-002, KM2-002)과 논리연산자 (and)로 구성되어 있다. 이 취약점 표현식의 의미는, 외부에서 Chunked Encoding 된 잘못된 입력이 들어오는 경우(AI1-002), 이로 인해 메모리 영역 중 힙의 버퍼 넘침이 발생하여(KM1-002), 메모리의 특정 영역의 악성코드를 실행(KM2-002) 시키는 것이다. 이들의 연결 관계는 and로 되어 있는데 이는 외부의 입력 조건에 해당하는 AI1-202가 만족되는 경우 나머지 조건들이 순서와 상관없이 만족되어 취약점의 악용이 성공되는 것을 의미한다.

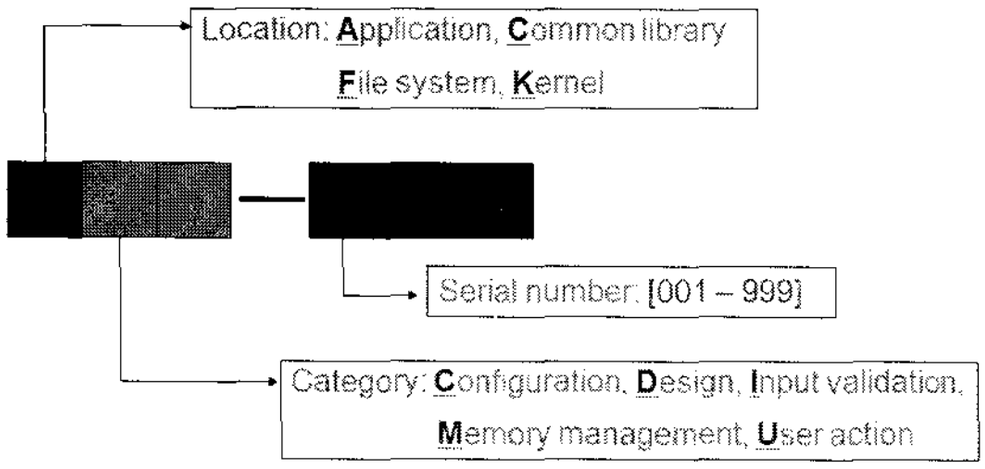
[그림 2]와 같은 의미기반 취약점 아이디 부여를 위해서는 각 단위 취약점의 이름을 지정하기 위한 명명법(Naming Convention)이 요구된다. [그림 3]은 본 연구에서 사용된 명명법을 보여주고 있다.

첫 자리는 취약점의 위치를 명시하는 것이고, 두 번째 자리는 취약점의 카테고리로 5가지 유형을 정의하고 있다. 같은 위치의 같은 유형의 취약점이지만 유형이 다른 것에 대해 순차적으로 번호를 지정하였고, 위치/유형

CVE-2002-0364
O 취약점 표현식 : AI1-202 and KM1-002 and KM2-002
- AI1-202 : no check input_remote_parameter _chunked encoding
- KM1-002 : heap overflow
- KM2-002 : Executed on Memory (Windows System Platform and certain OS version)

설명 : Chunked encoding 형태의 원격 입력 데이터의 길이를 점검하지 않는 문제로 인해, heap 넘침이 발생하고 이로 인해 메모리의 특정 영역의 악성코드가 실행되는 취약점

〔그림 2〕 의미기반 취약점 아이디 부여방법을 통한 분석 예



〔그림 3〕 단위 취약점의 명명법(Naming Convention)

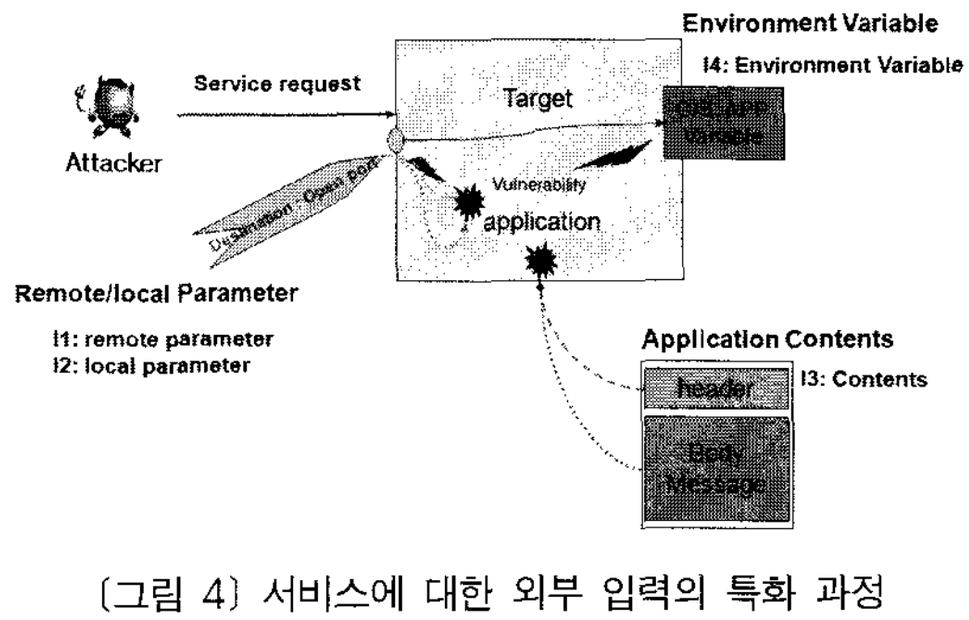
과 상관없는 단위 취약점 인스턴스들을 위해서 3자리의 순서번호(Serial Number)를 지정하였다. 이렇게 정의된 단위 취약점 명명법은 새로운 취약점이 나올 때 마다, 해당 취약점을 구성하는 구성 요소에 해당하는 단위 취약점을 식별하는 데에 유용하게 사용되었다.

[그림 3]과 같은 의미기반 취약점 아이디 부여를 위해 본 연구에서는 DEVS 형식론의 SES(System Entity Structure)에서 사용된 2가지 시스템 분류법인 분할(Decomposition) 및 특화(Specialization)를 사용하였다.

분할(Decomposition) : 취약점의 표현을 취약점 표현식으로 표현하는 것은 해당 취약점의 표시를 새분화된 수준으로 수행하고 이들 사이의 관계를 명시하는 것이다. 즉, 취약점을 여러 개의 조각으로 나누는 단위 취약점 개념은 SES의 분할과 같다. 이러한 취약점의 단위를 분할하여 단위 취약점으로 표현한 것은 단위 취약점의 표현에서 정의하고 있다. [그림 2]에서 제시되고 있는 버퍼 넘침 취약점 분석 결과를 보면 각 취약점을 여러 개로 분할하여 단위 취약점으로 표현한 것을 볼 수 있다. 취약점은 유형 별로 이와 같은 분할이 먼저 정의 된다. 본 연구에서 분석된 약 1,000개의 취약점에서는 7종류의 취약점 (버퍼넘침, 예외처리 오류, 과도한 서비스 요청, 약한 인증 기술 사용, Cross Site Scipt, 경쟁상태, 스푸핑, 설정 에러)에 대한 분할을 정의하였다.

특화(Specialization) : 단위 취약점 각각은 의미적으로 더 이상 나눌 수 없는 구성 요소 이지만, 이는 적용시에 특화된 형태로 나타낼 수 있다.

즉, 프로그램이 수용할 수 없는 긴 외부 입력에 대해



〔그림 4〕 서비스에 대한 외부 입력의 특화 과정

서 [그림 4]와 같이 특화가 이루어진다. [그림 4]의 경우는 외부 입력에 대한 4가지 유형을 보여주는 것으로, 외부에서의 서비스 요청 파라미터(I1), 내부에서 보내는 서비스 요청 파라미터(I2), 특정 소프트웨어에서 열어볼 수 있는 컨텐츠의 헤더 및 메시지 내용(I3) 그리고, 환경 변수(I4)로 특화 하였다.

Ⅳ. 취약점 및 공격 탐지 규칙 통합

본 연구에서 제안하는 의미기반 취약점 아이디 부여 방법의 활용에 있어서 어떻게 취약점 점검 규칙과 공격 탐지 규칙에 사용될지에 대해 고찰 하고자 한다. 일반적으로 취약점이 발견되면 다음과 같은 상황을 고려하여야 한다.

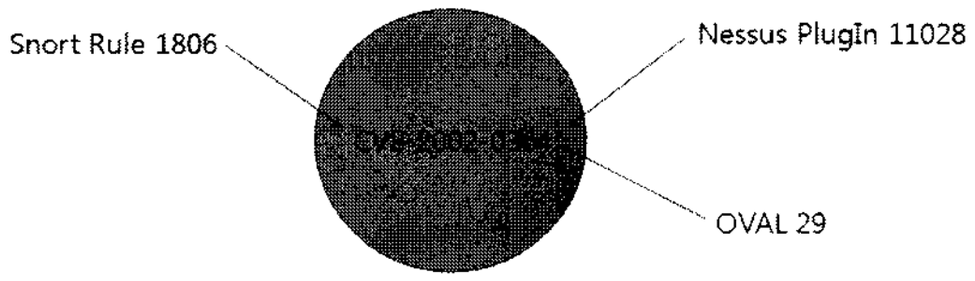
1) 해당 취약점이 정말로 존재하는 지에 대한 의문 즉, 오탐 여부를 고려한다.
2) 해당 취약점이 존재함으로 인해서 내 시스템에 어떤 일이 발생할 것인지를 고려한다.
3) 해당 취약점을 제거하거나 혹은 제거가 불가능하다면 우회할 수 있는지를 고려한다.

다음 시나리오는 위의 질문 중 첫 번째 질문에 해당하는 시나리오 이다.

시나리오 1 : 네트워크 보안 관리자는 취약점 점검도구를 통해 취약점을 발견하였다. 그는 해당 취약점이 정말로 존재하는 것인지 아니면 시스템에서 구동되는 특정 서비스로 인한 경고인지에 대해서 재확인을 하고 싶어 한다.

단위 취약점을 정의하지 않은 상황에서 위 시나리오를 달성하기 위해서 해야할 일은 다음과 같다.

먼저, 해당 취약점을 점검하는데 사용된 취약점 점검 규칙의 카테고리 및 점검 규칙의 내용을 확인하여야만



〔그림 5〕 CVE-2002-0364의 맵핑 정보

한다. 이를 통해 취약점이 어떤 방법으로 발견되었는지를 확인하게 된다. 취약점이 발견되었을 때 전달되는 대표적 취약점 식별자인 CVE-ID는 취약점 점검 규칙, 공격 탐지 규칙 및 취약점 데이터베이스를 연동해주는 정보로 활용되게 된다. [그림 5]는 CVE-ID (CVE-2002-0364)이 침입탐지 시스템 Snort의 SID (1806), 취약점 점검도구 Nessus PlugIn ID(11028) 및 시스템 취약점 점검 도구 OVAL의 ID (29) 사이의 관계의 중심에 있는 것을 확인 할 수 있다.

Example) 다음은 CVE-2002-0364가 발견 되었을 때 Nessus에서 전달해주는 경고 메시지이다.

```
The remote server is vulnerable to a buffer overflow in the .HTR filter.
An attacker may use this flaw to execute arbitrary code on this host (although the exploitation of this flaw is considered as being difficult).
Solution :
To unmap the .HTR extension :
1.Open Internet Services Manager.
2.Right-click the Web server choose Properties from the context menu.
3.Master Properties
4.Select WWW Service -> Edit -> HomeDirectory -> Configuration and remove the reference to .htr from the list.
See MS bulletin MS02-028 for a patch
```

위의 경고 메시지에 대한 취약점표현식은 위의 [그림 2]에서 보여준 것과 같다. 본 예에서는 해당 취약점표현식을 기반으로 Nessus의 취약점 점검 규칙의 연관을 제시하고자 한다.

다음은 Nessus 취약점 점검도구의 점검규칙(PlugIn 11028) 및 이에 대한 단위 취약점(AI1-202) 맵핑을 보여주고 있다. 이러한 맵핑은 Nessus의 취약점 점검 규칙이 취약점의 점검을 위해 외부에서 입력을 생성하여 80번 포트를 통해 전송(line,4 및 line,11)하고 이에 대한 반응을 본다(line,12 및 line,13). 이를 통해 취약점이 발견되고 나서 사용자는 취약점의 발견에 사용된 규칙이 어떤 단위 취약점과 연결되어 있는지를 보고, 어떤 방식으로 취약점을 점검하는지를 알 수 있게 된다.

Nessus PLUGIN 11028 → CVE-2002-0364 : AI1-202

```
1 : include("http_func.inc");
2 : req = string("POST /NULL.htr
    HTTP/1.1\r\n","Host : ",
3 : get_host_name(),
    "\r\n","Transfer-Encoding : chunked\r\n\r\n","20\r
    \n",crap(32), "\r\n","0\r\n\r\n");
4 : port = get_http_port(default : 80);
5 : sig = get_kb_item("www/hmap/" + port +
    "/description");
6 : if ( sig && "IIS" >!< sig )
7 : exit(0);
8 : if(!get_port_state(port))
9 : exit(0);
10 : soc = http_open_socket(port);
11 : if(soc){send(socket : soc, data : req);
12 : r = http_recv_headers2(socket : soc);
13 : if(egrep(string : r, pattern : "^HTTP/1.[01] 100
     Continue")){
14 : r2 = http_recv_body(socket : soc, length : 0,
     headers : r);
15 : if(!r2) security_hole(port);
16 : }
17 : http_close_socket(soc);}
```

시나리오 2 : 네트워크 보안 관리자는 취약점 점검 도구를 통해서 어떤 취약점이 관리대상 네트워크에 있는지를 확인했다. 이후 그는 해당 취약점이 악용하기 위한 공격이 어떤 것이 있는지, 특히 해당 공격의 패턴이 어떤 것인지를 궁금해 하게 된다.

상기 시나리오는 2번째 질문에 해당하는 "취약점으로 일어날 수 있는 시스템에 대한 공격 및 결과"에 대한 것이다. 이러한 질문에 대해 답변을 하기 위해서는 해당 취약점과 공격정보사이의 연결이 필요하며, 이를 단위 취약점을 활용할 경우 좀 더 자세한 수준의 상호 관계를 찾을 수 있게 된다.

Example) 다음은 취약점 CVE-2001-0241에 대한 Nessus의 경고 메시지이다. 이 취약점의 경우 웹 서버에 대한 외부 입력이 버퍼 넘침을 가져오고 이로 인해 시스템이 마비되거나 서비스가 종료되는 상황을 가져올 수 있다. 또한, 공격자가 교묘하게 버퍼 넘침 효과를 활용할 경우 악성 코드의 실행을 통한 권한 획득 및 정보 유출이 가능하다.

There is a buffer overflow in the remote IIS web server. It is possible to overflow the remote Web server and execute commands as the SYSTEM user. At attacker may make use of this vulnerability and use it to gain access to confidential data and/or escalate their privileges on the Web server.
See : http://www.eeye.com/html/Research/Advisories/AD20010501.html for more details.
Solution :
See : http://www.microsoft.com/technet/security/bulletin/ms01-023.mspx
Risk factor : High

위의 취약점에 대한 단위 취약점 기반 취약점 표현식은 아래와 같다.

CVE-2001-0241
O 취약점표현식 : AI1-220 and KM1-001 and KM2-001
- AI1-220 : no check input_remote_ parameter_ service specific string
- KM1-001 : Stack Overflow
- KM2-002 : Executed on Memory (Windows System Platform and a certain OS version)

취약점 표현식을 기반으로 Nessus의 점검 규칙과 연결관계는 다음과 같이 특정 문자열을 원격 파라미터로 전달될 때 이로 인한 스택의 넘침과 악성코드의 실행을 표현하고 있다. 특히, Nessus 점검 규칙에서는 외부 입력에 의해 해당 웹서버의 서비스가 중단 되는지를 확인(line,16)하는 것을 볼 수 있다.

Nessus PLUGIN 10657 → CAN-2001-0241 : AI1-220

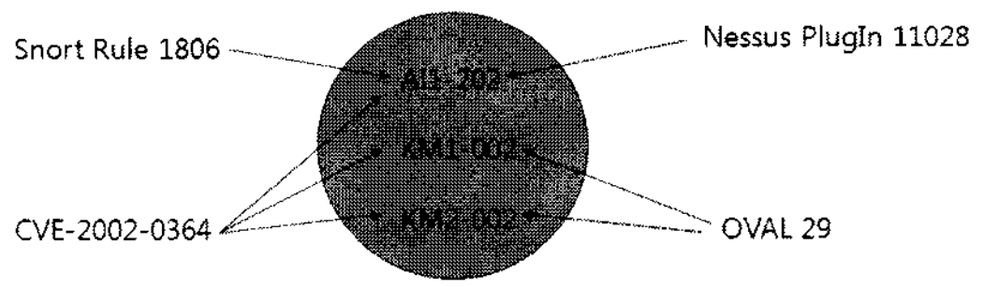
```
1 : include("http_func.inc");
2 : port = get_http_port(default : 80);
3 : sig = get_kb_item("www/hmap/" + port +
    "/description");
4 : if ( sig && "IIS" >!< sig ) exit(0);
5 : if(get_port_state(port)) {
6 : if(http_is_dead(port : port))exit(0);
7 : mystring = string("GET /NULL.printer
    HTTP/1.1\r\n");
8 : mystring = string (mystring, "Host : ",
    crap(420), "\r\n\r\n");
9 : mystring2 = http_get(item : "/", port : port);
10 : soc = http_open_socket(port);
11 : if(!soc) {exit(0);}
12 : else {
13 : send(socket : soc, data : mystring);
14 : r = http_recv(socket : soc);
15 : http_close_socket(soc);
16 : if(http_is_dead(port : port))
17 : {
18 : security_hole(port);
19 : exit(0);
20 : } } }
```

본 연구에서는 취약점을 악용하는 공격에 대해 단위 취약점을 기반으로 연결관계를 찾고자 한다. 다음은 Snort 침입 탐지 시스템의 규칙으로서, 위의 Nessus 취약점 점검도구가 점검하는 같은 내용에 대한 공격정보를 볼 수 있다. 이 공격 정보에서 단위 취약점 단위에서의 연결은 AI1-220과 연결되는 것을 알수 있다. 이는, 공격 정보가 갖는 특성이 외부입력에 대한 정의이기 때문에 이러한 연결이 된 것이다.

Snort ID 971 CAN-2001-0241 : AI1-220

```
alert tcp $EXTERNAL_NET any ->
$HTTP_SERVERS $HTTP_PORTS
(msg : "WEB-IIS ISAPI .printer access";
flow : to_server,established;uricontent : ".printer";
nocase;reference : arachnids,533;
reference : bugtraq,2674;reference : cve,2001-0241;
reference : nessus,10661;
classtype : web-application-activity; sid : 971;
rev : 9;)
```

이와 같이, Nessus 탐지 규칙과 Snort 탐지 규칙에 대한 맵핑 수준을 낮추게 될 때 [그림 5]의 CVE-ID 기반 매핑은 [그림 6]과 같이 좀 더 낮은 레벨에서의 매핑으로 나타낼 수 있다.



〔그림 6〕 의미기반 취약점 식별 기법을 사용한 맵핑 결과

시나리오 3 : 보안 관리자는 발견된 취약점을 기반으로 어떤 공격이 있을 수 있는지와 공격으로 인한 영향을 예측 했지만, 해당 취약점을 제거하기 어려운 상황이다. 관리자는 불가피하게 취약점을 보유하면서 서비스를 안정적으로 제공하고자 한다.

위 시나리오는 본 장의 처음에 제시한 3번째 질문으로 단위 취약점을 활용할 경우 위와 같은 시나리오에 대해서 보호대책을 고려할 때 좀 더 효과적으로 수행할

수 있게 된다. 취약점의 대책을 단순히 패치를 설치해야 한다는 형태로 제시하기 보다는 외부 입력 문자열의 길이를 점검하지 않는 것을 고치도록 하는 좀 더 세부적인 수준에서의 논리적 전개가 가능하게 된다. 예를 들어, 꼭 필요한 애플리케이션의 경우, 패치를 설치하지 않고 운영하면서, 임시적으로 특정 문자열을 포함하는 패킷을 차단할 수 있다.

V. 결 론

본 논문은 취약점을 표현하는 방법에 있어서 기존의 단순 아이디 부여 방법을 보완하여, 소프트웨어의 취약점을 효과적으로 표현할 수 있는 의미기반 아이디 부여 방법을 제시하였다. 의미기반의 아이디 부여로 인해서 가져올 수 있는 효과는 기존의 취약점 및 공격 기법을 탐지하기 위한 점검 규칙의 실질적인 점검 대상을 체계적으로 표현할 수 있다는 점이다. 취약점 점검 도구인 Nessus의 취약점 점검 규칙과 공개용 침입 탐지 시스템인 Snort가 갖는 공격 탐지 규칙에 이러한 의미기반의 취약점 아이디 부여 방법이 어떤 효과를 가져올 수 있는지를 실제 점검 규칙을 기반으로 제시하였다. 본 연구에서는 약 1,000개의 취약점을 분석하여 100여개의 단위 취약점을 얻어 냈고, 100여개의 단위취약점을 활용하여 모든 취약점을 표현하는 데이터베이스를 구축하였다.

향후 의미기반 취약점 표현법을 기반으로 NIST의 NVD 내의 모든 취약점에 대한 데이터베이스를 구축하고 이를 활용한 취약점의 발견 및 관리 기법을 제안하고자 한다. 특히, 취약점 데이터베이스가 의미기반 취약점 식별자를 활용하여 구축될 경우, 가독성이 높아지며, 대책을 좀 더 빠르게 제안하는 효과를 기대할 수 있다.

VI. 부록 - 용어 설명

용어	설명
NIST	미국 상무성(Department of Commerce) 산하의 기술개발 및 표준화 기관. 취약점의 이름인 CVE를 관리. NVD (National Vulnerability Database) 운영
CVE 호환성	CVE로 명명된 취약점을 어느 정도 커버하는 지를 알게 하기위한 목적으로 제공하는 인증 임.
단위취약점	취약점을 표현하는 최소의 단위로서, 더 이상 나눠 질 수 없고, 하나의 취약점을 표현하기 위한 논리연산자를 통해 연결됨.
취약점표현식	단위 취약점과 논리연산자로 표현되며, 참(true)로 검증될 경우 취약점이 악용되어 공격이 성공하는 상태임.
위험상태 표현식	취약점이 완전히 악용되기 직전의 상태의 표현식으로, 참(true)로 검증될 경우 취약점이 악용되기 직전의 위험상태임.
Cross Site Script	취약점의 한 종류, 악의적인 사용자에 의해 웹페이지에 악성 코드가 삽입되는 형태의 위변조가 가능하게 되어, 피해를 입는 사용자가 발생.
Chunked Encoding	하나의 메시지를 통신을 위해 여러 개의 조각으로 분할하는 경우 각 조각의 메시지 및 제어 정보를 일컬음.
버퍼넘침 (Buffer Overflow)	소프트웨어에 제공되는 입력 데이터의 길이를 점검하지 않는 문제로 메모리의 특정 허가된 영역 밖에 데이터 복사가 발생하는 취약점임.
Nessus PlugIn	Nessus에서 취약점 점검을 위해 사용되는 점검 규칙들로, NASL이라는 스크립트 언어를 사용.
Snort Rules	Snort에서 사용되는 공격 탐지 규칙으로, 공격 행위에 해당하는 패턴을 규칙으로 표현함, SID는 Snort Rule에 부여된 고유한 식별자임
OVAL (Open Vulnerability Assessment Language)	시스템의 취약점을 점검하기 위한 시스템 내부 정보의 XML 기반의 표현 방법을 정의함. OVAL의 점검 규칙들은 커뮤니티 포럼을 통해 개발된다는 특성이 있음

참고문헌

[1] B. P. Zeigler, *Theory of Modeling and Simulation 2nd Edition*, Academic Press, 2000.

[2] B. P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Orlando, FL : Academic, 1984.

[3] B. P. Zeigler, *Object-Oriented Simulation with Hierarchical, Modular Models*, San Diego, CA, USA : Academic Press, 1990.

[4] HyungJong Kim, "System Specification Network Modeling for Survivability Testing Simulation", Information Security and Cryptology ICISC 2002, LNCS Vol. 2587, pp. 90-106, November, 2002.

[5] Nancy R.Mead et. al., "Survivable Network Analysis Method", CMU/SEI-2000-TR-013, Sep.

2000

[6] Robert J. Ellison, David A. Fisher, Richard C. Linger, Howard F. Lipson, Thomas A. Longstaff, Nancy R. Mead "Survivability : Protecting Your Critical Systems", IEEE Internet Computing, November December, Vol 3, pp. 55-63, 1999

[7] F. Cohen, "Simulating Cyber Attacks, Defenses, and Consequences", Computer & Security, Vol.18, pp. 479-518, 1999

[8] M. Bishop, "Vulnerabilities Analysis", Proceedings of the Recent Advances in Intrusion Detection, pp. 125-136, September, 1999

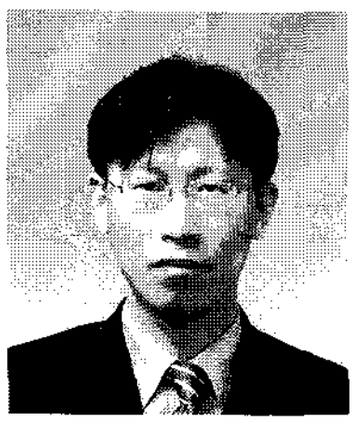
[9] N. Ye and J. Giordano, "CACA-A Process Control Approach to Cyber Attack Detection", Communications of the ACM, Vol.44(8), pp. 76-82, 2001.

[10] TaeHo Cho and HyungJong Kim, "DEVS Simulation of Distributed Intrusion Detection System", Transactions of the Society for Computer Simulation International, vol. 18, no. 3, pp. 133-146, September, 2001.

[11] Jay Beale, *Snort 2.1 Intrusion Detection 2nd Edition*, Syngress, 2004.

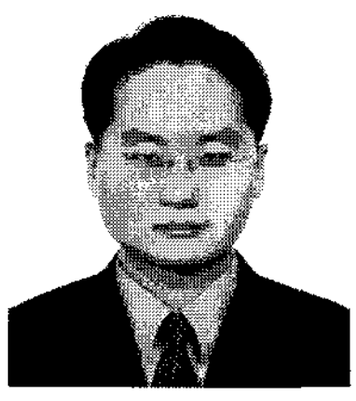
[12] Renaud Deraison, *Nessus Network Auditing*, Syngress, 2004.

〈著者紹介〉



김 형 종 (Hyung-Jong Kim) 종신회원

1996년 : 성균관대학교 정보공학과 학사
1998년 : 성균관대학교 정보공학과 석사
2001년 : 성균관대학교 전기전자 및 컴퓨터공학과 박사
2001년~2007년 : 한국정보보호진흥원 수석연구원
2007년~현재 : 서울여자대학교 컴퓨터학부 전임강사
<관심분야> 취약점 분석 평가, 네트워크 보안, VoIP 트래픽 분석



정 태 인 (Tae-In Jung) 정회원

1998년 : 한양대학교 전기공학과 학사
2000년 : 한국과학기술원 전기및전자공학과 석사
2000년~2001년 : 데이콤
2001년~현재 : 한국정보보호진흥원 수석연구원
<관심분야> 네트워크, 정보보호