수준별 신규 취약점 점검 모듈 개발 방법론

백승현* · 오형근* · 이도훈*

요 약

웜, 바이러스를 비롯한 최근의 사이버위협은 인터넷과 같은 주요 정보통신시설에 내제된 취약점을 악용하여 경제적 및 정치적 목적을 달성하기 위한 형태로 발전하고 있다. 이를 위해 취약점 발표 시 신속하게 취약점 점검 모듈을 개발하고 이를 적용하는 일이 매우 필요로 되고 있다. 본 논문에서는 신규 취약점 발생 시 이에 대한 점검 프로그램을 제작하는 방법론을 연구한다. 첫째, GFI LANGuard Network Security Scanner, Nessus 등을 포함한 7개의 취약점 점검 도구에서 신규 취약점 발생 시 대응하는 방법을 조사 및 분석하고 이에 대해 비교 분석을 수행한다. 둘째, 이를 기반으로 5가지 고유한 취약점 점검 모듈 제작 방법을 수준별로 정의하고 이를 제안한다. 마지막으로, 가상의 조건을 정의하고 5가지 취약점 점검 모듈 제작 방법 중 주어진 조건에 부합하는 최적의 방법을 결정하는 절차를 제시한다.

Study of Methodologies for New Vulnerability Checking Module Development Proper to User Level

Seung-Hyun Paek* · Hyung-Geun Oh* · Do-Hoon Lee*

ABSTRACT

Recent trends for cyber threat such as worm and virus exploit vulnerabilities inherent to main information communication infrastructures like the internet to achieve economical and political goals. It needs to develop checking programs for new vulnerabilities published in prompt and apply them to vulnerable systems for the defense of those cyber threats. In this paper, we study of methodologies for new vulnerability checking module development proper to user level. First, we analyze current 7 methodologies for the development of new vulnerability checking modules including GFI LANGuard and Nessus and then compare them. Second, We define and propose the 5 unique methodologies for the development of new vulnerability checking modules in depth. Finally, we induct the best methodology proper to a certain user level by assessing each methodology according to conditions which is set virtually.

Key words : Vulnerability, Vulnerability Checking, Development Methodology

* 국가보안기술연구소

1. 서 론

인터넷을 비롯한 정보통신기술의 발전으로 개인에서부터 민간 기업 및 공공 기관에 까지 정보통신기술에 대한 의존도가 날로 심화됨에 따라 사회 주요 시설에 존재하는 취약성을 이용한 전자적 침해와 사이버 위협에 대한 위험도가 높아지고 있다. 최근에는 인터넷 웜 및 바이러스와 해킹 공격이 정보통신기술에 내재해 있는 취약점을 악용하여 로컬 PC를 공격하는 추세에서 인터넷과 같은 정보통신기술 인프라로 또는 이를 기반으로 하는 서비스를 공격하는 추세로 전환하고 있다[1, 2].

이러한 추세에 있는 사이버 위협들로부터 정보통신기술 인프라 및 서비스 등의 자산을 보호하기 위해 대부분의 업체에서 방화벽 기술, 침입탐지/방지 기술, ESM(Enterprise Security Manager), 취약점 점검 기술 등과 같은 시스템 보안 도구에 대한 연구 및 개발이 활발히 이루어지고 있다[3, 4]. 이 중에서 특히 사이버 위협의 결정적인 공격 포인트가 되는 시스템의 취약점을 원천적으로 제거하는 기술을 개발하고 도입하는 사례가 많아지고 있다. 그리고 이를 자동화하기 위한 취약점 점검 도구들이 많이 제작되고 있다. 또한, 신규 취약점이 발표될 경우 이에 대한 신속한 점검을 수행해야 하는데, 이를 위해 신규 취약점에 대한 점검 프로그램을 제작하는 방법론에 대한 연구가 절실히 필요하다.

이를 위해 본 논문에서는 신규 취약점 발생 시 이에 대한 점검 프로그램을 제작하는 방법론을 연구한다. 첫째, 가장 기능상으로 우수하다고 평가한 GFI LANGuard Network Security Scanner, Nessus 등을 포함한 7개의 취약점 점검 도구에서 신규 취약점 발생 시 대응하는 방법을 분석한다. 둘째, 이를 기반으로 5가지 취약점 점검 모듈 제작 방법을 정의하고 이를 분석한다. 마지막으로, 가상의 조건을 정의하고 5가지 취약점 점검 모듈 제작 방법 중 주어진 조건에 부합하는 최적의 방법을 결정하는 절차를 예제로서 설명한다.

본 논문의 구성은 다음과 같다. 제 2장에서 관련 연구로 기존의 취약점 점검 도구에서 취하는 신규 취약점 대응 방식에 대해 조사하고 비교 분석을 수행한다. 제 3장에서 신규 취약점 점검 모듈 제작 과정과 본 논문에서 다루는 문제의 범주를 정의한다. 그리고 제 4장에서 앞에서 다룬 내용을 바탕으로 신규 취약점 점검 방법론에 대해 설명한다. 이를 바탕으로 제 5장에서 특정 조건에 대해 최적의 방법론을 도출하는 과정에 대해 사례 분석을 수행한 후 마지막으로 제 5장에서 결론을 맺는다.

2. 관련 연구

2.1 SecuGuard NSE(4)

SecuGuard NSE는 보안취약점 분석 시스템과 서비스를 전문으로 하는 국내 보안 업체인 나일소프트에서 개발한 상용 네트워크 취약점 점검 도구이다. SecuGuard NSE의 특이 사항은 하나의 콘솔 에이전트가 도구가 설치되어 있는 여러 서브 네트워크들에 대해서도 점검 기능을 수행하고 결과를 취합할 수 있다는 것이다. 또한, Crystal Report 9.0 기능을 이용하여 다양한 비주얼 양식을 보고서로 제공한다는 것이다. SecuGuard NSE는 다음과 같은 방식으로 신규 취약점에 대응한다.

- 엔진 내부에 점검하는 모듈을 내장하고 있기 때문에 매달 엔진 갱신
- 사용자에 의한 대응 방법은 없음
- 현재 1500여 가지 취약점 점검

2.2 Nessus(5,6)

Nessus는 1998년 Renaud Deraison이 공개 소스로 제작한 취약점 점검 도구로 2002년 이후 Tenable Network Security社가 공개 소스 라이선스를 가지고 있지만 사용상에 제약은 없으며 무료이다. Nessus

는 전 세계적으로 75,000여 조직에 사용되고 있으며, 유닉스/리눅스 기반 Nessus 서버, 유닉스/리눅스/MS 윈도우즈 기반 Nessus 클라이언트, 그리고 윈도우즈 전용 Nessus(이름 : NeWT)로 배포되고 있다. Nessus는 다음과 같은 방식으로 신규 취약점에 대응한다.

- NASL 스크립트 언어로 작성된 점검 스크립트를 정의된 스케줄에 따라 갱신
- 사용자가 NASL 스크립트를 작성하여 신규 취약점에 대해 대응 가능
- 현재 6000여 가지 취약점 점검

2.3 Internet Scanner(7,8)

Internet Scanner는 Internet Security Systems 社에서 개발한 네트워크 취약점 점검 도구로 네트워크로 연결된 서버, 데스크 탑 등의 시스템에 대해 원격 점검을 수행한다. Internet Scanner는 다음과 같은 방식으로 신규 취약점에 대응한다.

- DLL 형태로 작성된 점검 모듈을 정의된 스케줄에 따라 갱신(X-press Update)
- 사용자가 작성한 특정 취약점 점검 실행파일(exe)을 구동할 수 있는 환경 제공(FlexChecks)
- 현재 1200여 가지 취약점 점검

2.4 GFI LANGuard Security Scanner(9,10)

GFI LANguard Network Security Scanner (N.S.S.)는 패치를 관리할 수 있는 취약점 스캐너로 관리자에게 보다 직관적이고 관리하기 편리한 상용의 보안 스캐닝 및 패치 관리 도구로 GFI Software Ltd.에서 개발하였다. GFI LANguard N.S.S.를 통해 IP나 시스템, IP범위, 도메인을 지정하여 스캔할 수 있다. GFI LANguard N.S.S.의 왼쪽 창에 각 시스템이 나타나고, 오른쪽 창에는 프로세스 정보를 자세히 부여한다. 각 노드별로 취약점을 나타내며 그것을 확장하며 더욱 자세한 내용을 볼 수 있다. GFI LANGuard N.S.S.는 다음과 같은 방식으로 신규 취약점에 대응한다.

- VBS 형식으로 작성된 점검 스크립트를 스케줄에 따라 최신의 항목으로 갱신
- 사용자에게 VBS 형식의 점검 스크립트 편집 및 디버깅 도구 제공
- 사용자에게 OS/IIS/배너 등의 조건 및 스크립트 조합을 통한 취약점 점검 항목 제작 추가 기능 제공
- 현재 1500여 가지 취약점 점검

2.5 CyberCop(11,12)

CyberCop는 McAfee社에서 제작한 네트워크 취약점 도구로 현재는 도구의 형태의 배포는 구하기가 어렵고 CyberCops ASaP라는 취약점 점검을 수행하는 웹 서비스를 제공한다.

- 실행파일로 작성된 점검 파일을 스케줄에 따라 최신의 항목으로 갱신(AutoUpdate)
- 사용자에게 CASL(Custom Audit Scripting Language)를 제공하여 점검 항목의 추가 가기능 제공
- 사용자에 NTCASL이라는 패킷 생성 및 스크립트 제작 GUI 제공
- 2000년도 당시 700여 가지 취약점 점검

2.6 기타(13,14,15)

Shavlik Technologies 제작하고 MS가 배포하는 윈도 시스템에 특화된 취약점 점검 도구인 Microsoft Baseline Security Analyzer(MBSA)와 AXENT가 개발하고 2000년 7월 Symantec과 합병하여 현재는 Symantec의 취약점 점검 도구인 NetRecon은 신규 취약점에 대하여 점검 항목 갱신 기능은 제공하지만 사용자에 의한 항목 추가 기능은 제공하지 않는다.

2.7 비교 분석

7개의 도구에서 취하는 신규 취약점 대응에 대한 현황을 종합 분석하면 신규 취약점 발생 시 업체가 점검 모듈을 제작하여 사용자들에게 배포하는 방식과 사용자에게 점검 모듈의 제작 환경을 지원함으로써 사용자들이 직접 항목을 추가하는 방식으로 나누어 볼 수 있다.

- 신규 취약점 항목 갱신
  - 대부분이 취약점 점검 항목을 주기적 갱신하여 신규 취약점에 대응
  - 취약점 항목 갱신 : 스크립트 갱신, 엔진 패치, DLL 갱신
- 점검 모듈 제작 환경의 제공
  - 몇몇 업체에서 사용자에게 점검 모듈 제작 환경을 제공
  - 사용자는 제작 환경을 기반으로 점검 모듈 제작 및 추가
  - 제작 방식 : 직접 프로그래밍(실행파일, DLL 등), 스크립트 언어, GUI 형식

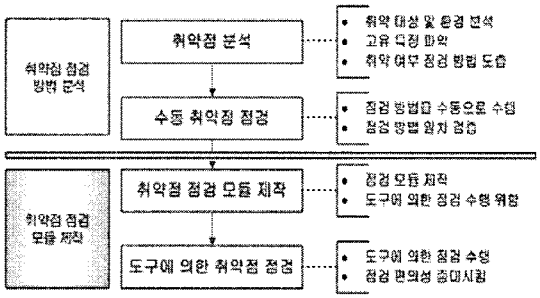
또한, 기존의 취약점 분석 도구들은 특정 취약점 점검 모듈을 만들기 위해 취약점에 따른 독립 실행코드를 계속 만드는 방식을 채택하지 않고 있다. 이유는 1000가지 이상의 취약점을 점검하기 위해 1000개의 독립 실행코드를 만드는 것보다는 공통적인 기능을 수행하는 엔진(구동기)을 개발하고, 그 엔진에 구동될 있는 취약점 점검 모듈들 선택하여 점검을 수행하는 것이 높은 확장성을 제공할 뿐만 아니라 관리도 용이하기 때문이다.

마지막으로, 취약점 점검 모듈을 제작하는 방식으로는 Nessus, CyberCops, GFI LANGuard NSS는 C++, Java와 같은 일반 프로그래밍 언어보다 상위 언어인 스크립트 언어를 제공한다. 또한 각 도구는 자신이 제공하는 스크립트 언어로 작성된 스크립트를 해석하고 실행하는 인터프리터(Interpreter)를 가지고 있다. 스크립트 언어를 사용하면 프로그램 언어를 이용한 점검 프로그램 작성보다 모듈 작성이 용이하다.

3. 문제 정의

3.1 신규 취약점 점검 모듈 개발 과정

신규 취약점 발생 시 일반적으로 (그림 1)과 같은 절차로 점검 항목을 제작하게 된다.



(그림 1) 신규 취약점 점검 항목 제작 과정

- 취약점 분석
  먼저, 발생한 신규 취약점에 대한 분석을 수행한다. 이때 취약 대상 및 환경을 분석하고 해당 취약점의 고유 특징을 분석한다. 그리고 해당 취약점을 점검할 수 있는 방법을 도출한다.
- 수동 취약점 점검
  전 단계("취약점 분석")에서 도출한 취약점 점검 방법을 수동으로 수행하여 도출한 점검 방법이 맞는지를 일차적으로 검증한다.
- 취약점 점검 모듈 제작
  검증된 취약점 점검 방법을 일반 프로그래밍 언어 혹은 스크립트 등의 다양한 방식으로 구현한다.
- 도구에 의한 취약점 점검
  제작한 취약점 점검 모듈을 구동(또는 실행)하여 다른 대상에 취약점이 있는 지의 여부를 점검한다. "수동 취약점 점검"에서 일차적으로 검

증된 점검 방법을 여러 호스트를 대상에 대해서 확대 검증하여 오탐율이 어떻게 되는지 검증한다. 이 단계에서는 제작한 점검 도구를 사용하기 때문에 점검 편의성이 증대될 뿐만 아니라 점검 시간도 단축된다.

3.2 신규 취약점 점검 모듈 개발 전제 조건

(그림 1)의 취약점 점검 모듈의 제작 과정은 크게 "취약점 점검 방법 분석" 부분과 "취약점 점검 모듈 제작"의 두 단계로 구분할 수 있다.

- 취약점 점검 방법 분석
  - 새로운 취약점이 발표되면 그 취약점에 대한 상세 사항 분석
  - 새로운 취약점에 대한 점검 방법 모색 및 검증
  - 점검방법을 찾는 문제는 정형적인 방법론이 존재하지 않음
  - 따라서 분석 전문가(사람)에 의해서만 달성 가능한 과정임
  - 분석 완료 기간은 분석자의 능력에 따라 달라지는 사항
    (분석자의 능력 : 분석 경험의 존재 여부, 취약점 이해 정도, 취약 대상이 되는 소프트웨어(운영체제, 응용 소프트웨어 등) 이해정도)
- 취약점 점검 모듈 제작
  - 분석한 결과에 따라 점검모듈을 만드는 과정
  - 제작에는 여러 방식(프로그래밍 언어에 의한 제작, 스크립트 언어에 의한 제작 등)이 존재
  - 사용자 편의와 점검 방법의 자유도에 따라 적절한 방식을 선택하여야 함

본 논문에서는 "취약점 점검 방법 분석"에 대해서는 정형적인 분석 방법론이 없기 때문에 "취약점 점검 모듈 제작"에 대한 방법론에 대해 분석하고 설명한다.

4. 신규 취약점 점검 모듈 개발 방법론

4.1 개요

제 2장에서 살펴본 것과 같이 각 도구들이 제공하는 제작 환경을 일반화하여 정리하고 몇 가지 추가하면 다음과 같은 제작 방식들을 정리할 수 있다.

- 방법0 : 실행파일 프로그래밍 방식
- 방법1 : DLL API 프로그래밍 방식
- 방법2 : DLL 코드 템플릿 프로그래밍 방식
- 방법3 : 스크립트 프로그래밍 방식
- 방법4 : 특정 조건 지정 GUI 방식

본 장에서는 5가지 방식에 대한 점검 방법론을 보다 상세히 정의하고 정의한 각 방식에 대해 장단점을 분석한 후, 특정 환경에서 어떤 방식이 최적의 방식인지를 선택하는 예를 들어 설명하겠다.

4.2 방법0 - 실행파일 프로그래밍 수준

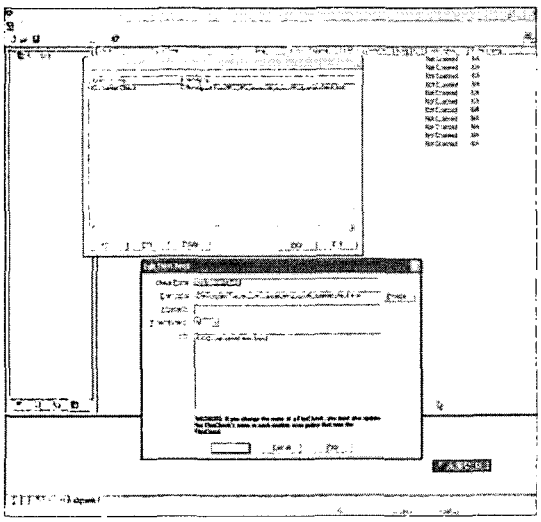
실행파일 프로그래밍 방식은 특정 취약점에 대한 점검을 수행하기 위한 단독 실행 파일을 사용자가 C++, Java 등과 같은 일반 프로그래밍 언어로 제작하는 방식이다. 보통 특정 취약점에 대한 점검을 수행하는 부분이 엔진에 포함되어 있으며, 경우에 따라서 사용자가 제작한 점검 실행 파일을 엔진이 호출하여 점검하도록 하는 경우도 있다.

- 특징
  - 편의성 : 보통 취약점 점검 구동 부분과 실행 부분이 분리되어 있지 않아 점검 실행 파일 개발 시 구동 부분을 매번 구현해야 하는 불편함이 존재
  - 난이도 : 취약점 점검 프로그램 작성 기술
  - 사용성 : 매번 엔진 업데이트를 하거나 별도의 취약점 점검 실행파일 개발

- 융통성 : 어떤 제약 조건도 없으므로 최고의 융통성을 지님

• 사례
- 나일소프트 SecuGuard : 엔진에 점검 수행 모듈 통합
- ISS Internet Scanner : 사용자에 의해 작성된 점검 실행 파일 추가 가능
- McAfee CyberCop Scanner : 매일 갱신되는 특정 취약점 점검 모듈이 실행 파일 형식임

• 평가
- 점검 모듈 개발 시 구동 모듈을 매번 구현해야 하는 부담으로 인하여 신속한 대응이 어려움



(그림 2) 방법0-실행파일 프로그래밍 방식 사례 : Internet Scanner의 FlexChecks 방식

4.3 방법1-DLL API 프로그래밍 수준

DLL API 프로그래밍 방식은 취약점 점검 모듈을 C++, Java와 같은 일반 프로그래밍 언어로 제작해야 한다는 점에서는 "방법0 - 실행파일 프로그래밍 방식"과 같다. 그러나 DLL API 프로그래밍 방식에서는 취약점 점검 구동 모듈과 실제 점검 모듈이 분리되어 있기 때문에 매번 구동에 필요한 부분을 제작할 필요가 없다. 이를 위해 특정 취약점 점검 모듈을 구동하기 위한 최소한의 API를 정의하고 이를 사용자에게 제공한다. 사용자는 제공된 구동기와의 인터페이스인 API를 구현하여 실제 점검 모듈을 컴파일하여 제작한다. 윈도우즈 환경에서 컴파일된 특정 취약점 점검 모듈은 구동기에서 동적으로 호출하여 실행할 수 있는 DLL (Dynamic Linking and Library) 형식을 가지며, 리눅스 및 유닉스 환경에서는 SO(Shared Object) 형식을 가진다. 마지막으로 작성된 DLL(혹은 SO) 파일을 실행 시 점검 항목이 될 수 있도록 구동기에 추가한다.

• 특징
- 편의성 : 구동기에서 실행되는 점검 모듈 작성을 위한 최소의 인터페이스 제공
- 난이도 : 구동기와 특정 취약점 점검 실행기 분리 구조
- 사용성 : 정의된 API 및 점검 수행 로직을 직접 구현하고 컴파일하여 모듈 제작
- 융통성 : 최소한 API 정의를 통한 정형화로 사용자에게 많은 자유도 제공

• 사례
- Ineternet Security Systems의 Internet Scanner : DLL 형식의 취약점 점검 모듈 존재, 그러나 사용자에게 API는 미공개
- 기타 분야로 리눅스/윈도우즈 디바이스 드라이버가 이러한 방식을 사용하고 있음

• 평가
- 취약점 점검 구동기와 실제 취약점 점검 모듈의 분리를 통해 응용 가능성을 높임
- 점검 로직 이외 공통되는 코드를 매번 중복하여 작성해야 하는 부하가 존재함

〈표 1〉 방법1의 API 예제

함수 명	설 명
char* Get_Name(void)	취약점의 이름
int Get_TargetProtocol(void)	점검하고자 하는 프로토콜(TCP, UDP 등)
int Get_TargetPort(void)	점검하고자 하는 포트
int Get_SecurityLevel(void)	점검하고자 하는 취약점의 중요도(주의/경고/위험)
char* Get_Description(void)	취약점에 대한 설명 및 대처방법
char* Get_CVEid(void)	취약점의 CVE 식별자
void Set_ProgressFunction(void (*Progress) (int percentage, char *msg))	점검 상황을 구동기로 상세하게 보고하기 위한 함수 포인터 지정
int Execute(char *target_addr)	취약점 점검 명령

4.4 방법2-DLL코드템플릿 프로그래밍 수준

"DLL API 프로그래밍 방식"은 점검 로직뿐만 아니라 API와 이에 부수적으로 수반되는 코드들을 모두 구현하는 반면에, "DLL 코드 템플릿 프로그래밍 방식"은 부수적으로 수반되는 코드들을 사용자에게 코드 템플릿 형식으로 제공하고 점검 모듈에 핵심 부분인 점검 로직은 직접 구현하게끔 코드 상에 명시하는 방식이다. 즉, API 구현을 위한 코드 프레임워크(framework)를 제공하고 사용자는 점검 로직에 해당하는 코드 세그먼트(segment)를 구현하게 함으로써 프로그래밍 언어로 코딩하는 데 있어서 최소의 노력만을 하도록 유도한다. DLL 코드 템플릿 프로그래밍 방식은 "방법 1"에서 정의한 API를 구현하는데 있어서 코드 템플릿 이라는 코딩 편리성을 제공한다.

- 특징
  - 편의성 : 실행에 필요로 되는 점검 모듈 제작에 있어서 프로그래밍 최소화
  - 난이도 : 방법 1 난이도 + 제공 가능한 코드 템플릿 정의 및 생성
  - 사용성 : 점검 수행 로직에 대해서만 직접 구현하고 컴파일하여 사용
  - 융통성 : 코드 프레임워크를 통한 반정형화로 사용자에게 비교적 많은 자유도 제공(코딩 스타일 제약)
- 사례
  - Microsoft Foundation Class(MFC) GUI 프로그래밍에 있어서 사용자게 전체 코드 템플릿과 "//TODO"라 표시되는 작성해야할 부분을 제시함
- 평가
  - 취약점 점검 구동기와 실제 점검 모듈의 분리를 통한 응용 가능성 높임
  - 점검 모듈 작성시 점검 로직 코드의 작성만으로 공통 코드의 중복 작성 부하 제거

4.5 방법3-스크립트 프로그래밍 수준

스크립트 프로그래밍 방식은 일반 프로그래밍 언어 대신에 취약점 점검에 적합한 고수준 언어(high-level language)인 스크립트 언어를 이용하여 취약점 점검 스크립트를 작성하고 이를 실행하는 방식이다. "방법 0 - 실행파일 프로그래밍 방식"에서 "방법 2 - DLL 코드 템플릿 프로그래밍 방식"까지는 C++, Java와 같은 일반 프로그래밍 언어를 이용하여 점검 모듈을 구현하고 이를 컴파일하여 제작하는 방식을 취한다. 그러나 스크립트 프로그래밍 방식은 스크립트에 대한 인터프리터(Interpreter)가 내부적으로 존재하여 사용자가 작성한 스크립트 해석하고 직접 실행함으로써 취약점을 점검한다.

- 특징
  - 편의성 : 취약점 점검에 자주 사용되는 기능을 위주로 한 상위 스크립트 언어 방식으로 사용

자 편의 수준이 높음
- 난이도 : 컴파일러 기술, 스크립트 인터프리터 기술 필요
- 사용성 : 취약점 점검에 적합하게 정의된 언어를 이용하기 때문에 점검 수행 로직을 신속하고 편리하게 구현할 수 있음, 컴파일 없이 인터프리터가 스크립트를 직접 해석하고 수행함
- 융통성 : 스크립트 언어가 지원하지 않는 기능은 사용자가 구현할 수 없음

• 사례
- Nessus : NASL(Nessus Attack Scripting Language) 스크립트 지원
- CyberCops : CASL(Custom Audit Scripting Language) 스크립트 지원
- GFI LANGuard : VBS(Visual Basic Script) 언어 지원

• 평가
- 고수준의 스크립트를 통해 신규 취약점 발생 시 신속한 대응이 가능함
- 스크립트 언어 제작을 통해 본 방식을 적용할 경우 해당 인터프리터 개발 등과 같이 부수적인 기술 개발이 수반됨
- 기존의 스크립트 언어를 사용할 경우 기존 도구의 주요 엔진 및 인터프리터를 적용 환경으로 포팅해야 하나 현재 관련 사례가 없어 개발 타당성을 검증 필요
- 신규 취약점 점검 스크립트를 업데이트하기 위해 구동 엔진 등 부수적 환경까지 동반 업데이트를 해야 하는 등의 많은 기술 노력이 소요

4.6 방법4-특정 조건 지정 GUI 수준

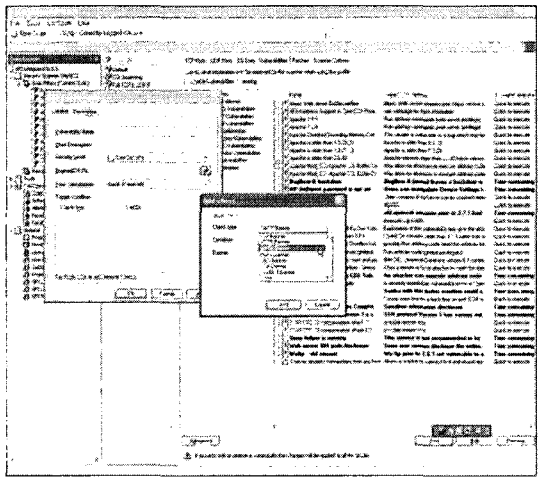
특정 조건 지정 GUI 방식은 점검할 필요가 있는 몇 가지 데이터 항목을 정의하고 사용자가 각 데이터 항목에 대한 조건을 지정하고 조건들을 조합할 수 있도록 GUI를 제공하는 방식이다. 지금까지의 방법은 사용자가 신규 취약점 점검 모듈을 제작하기 위해 프로그램을 작성하였으나 "특정 조건 지정 GUI 방식"을 이용하게 되면 자신이 점검하고자 하는 항목을 GUI 방식의 조건 조합을 통해 만들 수 있다. 그렇기 때문에 사용자의 노력을 최소화 하지만 지정 및 조합을 위해 제공하는 데이터 항목과 조건이 특정 범위에만 해당되기 때문에 조금이라도 범위가 벗어나는 취약점에 대해서는 실제적으로 활용이 불가하다는 치명적인 단점을 가지고 있다. 특정 조건 지정 GUI 방식을 지원하기 위해서는 취약점 점검에 근간이 되는 기본적인 점검 패턴을 추출하여야 하고 이를 GUI 방식으로 표현할 수 있어야 한다.

• 특징
- 편의성 : 특정 범위의 조건에 대해서는 GUI 방식을 이용한 최고의 사용자 편의성 제공
- 난이도 : 특정 조건 지정을 위한 기본 데이터 항목 추출 기술, 지원 범위에 따라 패킷 편집 및 생성 기술, 코드 생성 기술, 컴파일 기술 등을 필요로 함
- 사용성 : 점검 수행 로직의 구현 없이 마법사에 해당 항목 입력만으로 점검 가능
- 융통성 : 특정 조건의 범위를 벗어나는 경우에는 점검을 수행할 수 있는 방법이 없음

• 사례
- GFI LANGuard Network Security Scanner: 특정 조건 조합 GUI 지원
- CyberCops : NTCASL(CASL 스크립트 저작 도구) GUI 지원

• 평가
- 특정 조건에 있어서는 신속하고 편리한 점검 항목 제작 가능
- 범위를 벗어나는 조건에 있어서는 점검 항목 제작 불가능
- 실제 개발 도구들이 거의 사용이 안 되고 있

는 등 상용화가 실패하였고 실효성이 거의 없음

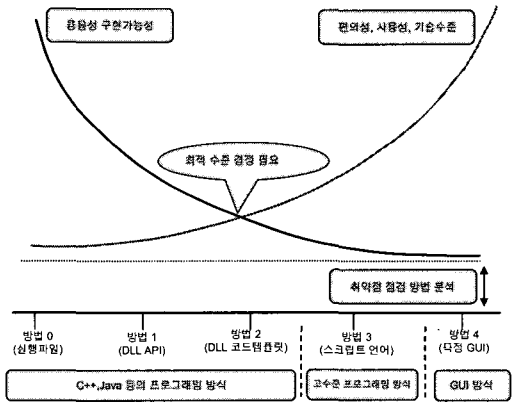


(그림 3) 방법4 - 특정 조건 지정 GUI 사례 : GFI LANGuard NSS의 조건 조합 GUI

4.7 수준별 개발 방법론 비교

방법 0에서 방법 4까지 총 5개의 방법론을 편의성, 난이도, 사용성, 융통성, 그리고 구현가능성이라는 4가지 기준에 따라 비교 분석하면 (그림 4)와 같은 그래프를 얻을 수 있다. 그래프를 통해서 다음과 같은 특징을 분석할 수 있다.

첫째, 모든 방법에 있어서 공통적으로 취약점 점검 방법 분석의 노력은 반드시 필요하다. 둘째, "방법 0"에서 "방법 4"로 갈수록 사용자의 편의성, 사용성은 증가하는 반면에 융통성, 구현 가능성은 낮아진다. 셋째, 사용자의 요구 사항 및 조건을 정확히 분석하여 융통성, 구현 가능성, 편의성, 사용성이 최적화 되는 지점이 존재한다. 마지막으로, "방법 0"에서 "방법 2"까지는 C++, Java 등의 일반 프로그래밍 언어에 의한 제작 방식이고 "방법 3"은 스크립트 언어라는 보다 상위 수준 언어에 의한 제작 방식이며 "방법 4"는 GUI 방식이라는 3가지 방식으로 다시 구분 지을 수 있다.



(그림 4) 취약점 점검 모듈 제작 방법론 비교

또한, 각 방법에 대해 제작환경의 개발에 소요되는 비용과 점검모듈을 제작 시에 얻는 사용자의 효용을 비교 분석하면 <표 2>와 같다.

〈표 2〉 취약점 점검 모듈 제작 방법론의 비용 대비 효용 비교

방법론	제작환경개발자 비용	점검모듈제작자 효용	기술타당성
방법 0 (실행파일)	• 없 음	• 없 음	가 능
방법 1 (DLL API)	• 구동/실행 분리 • API 정의	• 개별 점검실행파일의 제작 불필요 • API 구현/컴파일 필요	가 능
방법 2 (DLL 코드 템플릿)	• 구동/실행 분리 • API 정의 • 코드 생성 기술	• 미리 작성된 코드 기반의 프로그래밍 용이 • 점검 로직 프로그래밍/컴파일 필요	가 능
방법 3 (스크립트 언어)	• 구동/실행 분리 • 스크립트 언어 및 인터프리터 지원	• 점검 항목의 신속한 생성 환경 제공 • 스크립트 이해 요구 • 스크립트 언어 제약시 점검 불가	가 능
방법 4 (GUI 방식)	• 구동/실행 분리 • 하부 구조 구현 • 특정조건항목 도출	• 특정 조건에 대해 빠른 제작 가능 • 범위 밖 점검 불가능 • 실효성이 적음	가 능

5. 최적 방법론 결정 사례

앞에서 언급한 5가지 취약점 점검 모듈 제작 방법들 중에서 사용자의 요구 사항 혹은 제약 사항, 그리고 조건 등에 따라 융통성, 사용성, 편의성, 구현 가능성 등이 최적화되는 방법을 결정하기 위해서는 다음과 같은 절차를 수행하여야 한다.

- 조건 정의
- 방법별 조건 만족 여부 검토
- 최적의 방법 결정

예제를 통해 최적의 방법을 결정하는 방법을 설명하기 위해, 먼저 사용자의 조건이 아래와 같다고 가정한다.

5.1 조건 정의 사례

주어진 조건이 대상, 기술타당성, 효과/비용, 그리고 개발 기간이라고 가정한다. 먼저, 대상에 대해 점검 모듈 제작자는 취약점 분석 및 점검 전문가이고 전문가는 취약점 분석을 통해 구현 가능한 점검 방법을 도출할 수 있으며 점검 프로그래밍이 가능하다.

기술 타당성 면에 있어서 기술적으로 구현이 가능해야 한다. 그리고 이를 판단하기 위한 근거로 연구가 진행된 사례나 도구로 구현된 사례가 존재해야 한다.

효과/비용 면에 있어서 제작환경을 개발하기 위해 드는 노력(즉 비용) 대비 모듈 제작자의 효용을 분석하고 높은 것을 택한다. 이용한 방식은 다음과 같다.

*"효과/비용" = ("효용 항목의 개수 - 이용항목의 개수") * 10 + 10*

예를 들어, 방법 0의 경우 비용과 효용의 개수 모두 없으므로 기본점수 10을 얻게 된다. "효과/비용"이 10이면 중, 그 이상이면 상, 그 이하면 하를 얻는다. <표 2>를 근거로 산정하면 다음과 같은 "효과/비용"을 산출할 수 있다. 방법 0은 중(10), 방법 1은 20(상), 방법 2는 30(상), 방법 3은 20(상), 그리고 방법 3은 0(하)으로 산출된다.

기간 면에 있어서 과제 기간인 1년 내 구현해야 한다. 제작 환경 개발 기간이 6개월이라고 가정하자. 그리고 각 방법별 개발 기간이 다음과 같다고 가정한다. 방법 0은 0개월, 방법 1은 3개월, 방법 2는 6개월, 방법 3은 10개월, 방법 4는 12개월로 가정한다.

5.2 방법별 조건 만족 여부 검토

<표 3>에서 보는 바와 같이 가로로 모든 조건에 걸쳐서 무늬가 있는 방법이 모든 조건을 만족하는 방법이다. 모든 방법이 조건 1-대상과 조건 2-기술타당성을 만족한다. 그러나 "조건 3-효과/비용"면에서 방법 0과 방법 4가 각각 중, 하로 조건을 만족하지 못한다. 그리고 조건 4-기간 6개월을 방법 3은 만족하지 못한다. 따라서 "방법1-DLL API 프로그래밍 방식"과 "방법 2-DLL 코드 템플릿 방식"이 주어진 모든 조건을 만족한다.

〈표 3〉 방법별 조건 만족 여부 검토의 예(기간 6개월 가정)

방 법	조건1-대상	조건2-기술타당성	조건3-효과/비용	조건4-기간
방법0 (실행파일)	해당	해당	중	0개월
방법1 (DLL API)	해당	해당	상	3개월
방법2 (DLL 코드 템플릿)	해당	해당	상	6개월
방법3 (스크립트 언어)	해당	해당	상	10개월
방법4 (GUI 방식)	해당	해당	하	12개월

5.3 최적의 방법 결정

"방법별 조건 만족 여부 검토" 단계에서 방법 1과 방법 2가 선정되었으며 둘 중에 사용자에게 더 많은 편의성을 제공하는 "방법 2-DLL 코드 템플릿 방식"이 최적의 방법이다.

6. 결 론

취약점을 이용한 침입을 차단하기 위해 취약점 점검 도구들의 개발 및 활용으로 시스템의 취약성을 점검하고 이를 제거하는 사례가 많아지고 있다. 그러나 신규 취약점 발생 시 이에 대한 점검 모듈 제작 방법론에 대한 연구가 미흡하기 때문에, 본 논문에서는 신규 취약점 발생 시 이에 대한 점검 프로그램을 제작하는 방법론을 연구하였다.

첫째, 가장 기능상으로 우수하다고 평가한 GFI LANGuard NSS, Nessus 등을 포함한 7개의 취약점 점검 도구에서 신규 취약점 발생 시 대응하는 방법을 조사 및 분석하고 이에 대해 비교 분석을 수행하였다. 대부분의 업체에서 신규 취약점 점검 항목을 스케줄에 따라 갱신함과 함께 사용자에게 신규 취약점에 대한 점검 항목을 작성할 수 있는 제작 환경을 제공하고 있다. DLL 프로그래밍 방식과 스크립트 언어 방식이 많이 사용되고 있었다.

둘째, 이를 기반으로 5가지 취약점 점검 모듈 제작 방법을 정의하고 이를 분석하였다. 점검 모듈의 형태와 제작 방식에 따라 "방법 0-실행파일 프로그래밍 방식", "방법 1-DLL API 프로그래밍 방식", "방법 2-DLL 코드 템플릿 프로그래밍 방식", "방법 3-스크립트 프로그래밍 방식", 그리고 "방법 4-특정 조건 지정 GUI 방식"을 정의하였다. 방법 0에서 방법 4로 갈수록 편의성과 사용성은 증가하지만 융통성, 구현가능성은 감소함을 분석하였다.

마지막으로, 가상의 조건을 정의하고 5가지 취약점 점검 모듈 제작 방법 중 주어진 조건에 부합하는 최적의 방법을 결정하는 절차를 예제로서 설명하였다. 대상, 기술타당성, 효과/비용, 기간(6개월) 등을 조건으로 설정하고 방법별로 조건에 부합하는 지를 검토한 후 최적의 방법을 결정하였다. 예제에서는 "방법 2-DLL 코드 템플릿 프로그래밍 방식"이 최적으로 결정되었다.

본 논문은 신규 취약점 점검 모듈 제작 방식을 결정할 지에 대한 의사 결정 자료로 활용될 수 있다. 점검 모듈 제작자가 프로그래밍 경험이 없고 개발 기간이 길어도 되는 상황에서는 "방법 4-특정 조건 지정 GUI 방식"이 적당하지만 점검 모듈 제작자가 분석 및 프로그래밍 전문가이고 개발 기간이 매우 짧을 때에는 "방법 1" 혹은 "방법 2"가 적당하다. 그리고 점검 모듈 제작자가 분석 및 프로그래밍 중급자 혹은 초급자이고 개발 기간이 비교적 충분할 때에는 "방법 3-스크립트 프로그래밍 방식"이 적당하다.

덧붙여, "방법 4-특정 조건 지정 GUI 방식"에서 더 나아가 신규 취약점 점검에 대한 점검 조건을 일반화하여 패턴을 정형화 할 수 있고 이에 대한 패턴이 구현 가능하다면 방법 4보다 더 제작 편의성이 높은 새로운 방법으로 "일반 조건 지정 GUI 방식"도 생각해 볼 수 있다. 그러나 기술적으로 타당한지에 대한 검토가 더 필요하다.

참 고 문 헌

[1] 백승현, 오윤근, 오형근, 이진석, "악성코드 탐지를 위한 통합형 시스템 프레임워크", *제17회 정보보호와 암호에 대한 학술대회*, 2005. 9.

[2] Stanger J. and Lane, T. P., "Hack Proofing Linux : A Guide to Open Source Security", Syngress Publishing, Inc., 2001.

[3] McClure S., Scambray, J., and Kurtz, G., "Hacking Exposed : Network Security Secrets

and Solutions, 4th ed.", McGraw-Hill, 2003.
[4] Russell, R., et al, "Hack Proofing Your Network 2nd ed", Syngress, Mar. 2003.
[5] Nessus Open Source Vulnerability Scanner Project, http://www.nessus.org
[6] Tenable Network Security, http://www.tenablesecurity.com/
[7] Internet Security Systems, http://www.iss. net
[8] Internet Scanner, http://www.iss.net/ products_services/ enterprise_protection/ vulnerability_assessment/scanner_internet.php
[9] GFI Software Ltd., http://www.gfisoftware.com.
[10] GFI LANGuard N.S.S., http://www.gfisoftware.com/stats/adentry.asp?adv = 142&loc =28.
[11] Network Associates Technology, Inc., "Cyber Cop Scanner for Windows NT and Windows 2000 Getting Started Guide Version 5.5", NAI, 1998-2000.
[12] CyberCop ASaP, http://www.mcafeeasap.com/intl/EN/content/cybercop_asap/default.asp
[13] Shavlik Technologies, http://www.shavlik.com
[14] Microsoft Baseline Security Analyzer, http://www.microsoft.com/technet/security/tools/mbsahome.mspx
[15] NetRecon, http://enterprisesecurity.symantec.com/products/products.cfm?productid=46.

백 승 현

1999년 한동대학교 전산전자공학부(공학사)
2001년 한국과학기술원 전산학과(공학석사)
2001년~2003년 (주)아이디스 (전임연구원)
2003년~현재 국가보안기술연구소(연구원)

오 형 근

1998년 순천향대학교 전산학부(공학사)
2000년 순천향대학교 대학원 전산학과(공학석사)
2000년 2월~8월 한국사이버페이먼트(선임연구원)
2000년 8월~현재 국가보안기술연구소(선임연구원)
2003년 9월~현재 고려대학교 정보보호대학원(박사과정)

이 도 훈

1989년 한양대학교 전산학과(공학사)
1991년 한양대학교 대학원 전산학과(공학석사)
1991년~2000년 국방과학연구소
2000년~현재 국가보안기술 연구소