원격 취약점 증명 도구의 설계 및 구현

김하원*, 김원호*, 권오훈*, 김 종*, 홍성제*, 김수용**, 한광택**, 박중길**

포항공과대학교 컴퓨터공학과*, 국가보안기술연구소**

A Design and Implementation of a Penetration Simulator for Remote Vulnerability

Ha Won Kim*, Won Ho Kim*, O-Hoon Kwon*, Jong Kim*, Sung Je Hong*, Su Yong Kim**, Kwang-Taek Han**, Joong Gil Park**

*Dept. of Computer Science & Engineering, Pohang Univ. of Science and Technology

**National Security Research Institute

요 약

특정 프로그램의 취약점 여부를 판단하기 위한 방법에는 프로그램의 버전이나 작동여부를 점검하는 방법과 해당 프로그램에 실제 침투를 수행하고 그 성공여부를 검증하는 방식이 있다. 후자를 이용한 방식은 침투 코드의 구현, 재사용, 확장이 어렵기 때문에 전자에 비해서 널리 사용되지 않지만, 정확한 결과를 얻을 수 있고 보안의 경각심을 높일 수 있는 장점이 있다. 본 논문에서는 최근에 널리 사용되는 원격 취약점 침투 코드들의 구조를 분석하여 다양한 취약점을 보다 정확하게 검증할 수 있고 새로운 취약점에 대해서도 쉽게 확장할 수 있는 원격 취약점 증명 프레임워크를 제시한 후 이를 바탕으로 원격 취약점 증명 도구를 설계하고 구현한다. 원격 취약성 증명 도구는 원격 취약점 침투 코드들의 모듈화 및 재사용을 위한 프레임워크 부분과 사용자의 입력을 받아서 모의 침투를 수행하고 침투 성공 증거를 제공하는 GUI 부분으로 구성된다.

I. 서론

현대 사회에서 인터넷이 생활의 일부가 되면서 과거에는 상상도 할 수 없을 만큼의 엄청난 양의 정보들이 실시간으로 처리, 보관, 전송되고 있다. 이러한 인터넷 사용자와 활용의 증가는 누구나 쉽게 다른 정보 시스템에 접근할 수 있게 하였으며, 각종 취약점 정보와 침투 기술들은 인터넷에서 쉽게 얻을 수 있게 되어 크고 작은 각종 보안 사고가 많이 발생하게 되었다. 따라서 내부의 중요한 자원을 보호하기 위한 보안이 심각한 문제로 대두되고 있고 이러한 문제의 해결책에 대한 요구가 더욱 증가하고 있다.

정보 시스템의 취약점을 점검하는 방법에는 크게 두 가지의 접근 방법이 있다. 하나는 해당 프로그램의 버전이나 작동여부 등을 점검하는 방식이고, 다른 하나는 실제적으로 정보 시스템에 침투를 수행하여 취약점을 점검하는 방법이다. 전자는 빠른 점검과 손쉬운 사용이 장점이지만, 많은 긍정 오류(false positive)를 유발하고, 침투에 미치는 많은 영향을 고려하지 않아서 점검의 정확성이 떨어진다. 후자는 정확성이 높고 보안의 경각심을 높이는 장점이 있지만 시스템에 해를 끼칠 수가 있으며, 무엇보다 구현이 어렵고 실제 수행에 있어 어려운 점이 많다[2].

취약점을 가진 프로그램은 지속적으로 보고 되고 있고 이를 침투하는 침투코드들도 계속 개발되

※ 본 연구는 한국전자통신연구원 부설 국가보안기술연구소 '취약성 증명 프레임 워크의 확장'의 연구 결과로 국가보안기술연구소의 지원을 받음.

고 있다. 침투를 통한 취약점 점검 방법은 이러한 침투 코드들을 이용해서 모의 해킹을 수행하고 그 결과를 보여줌으로써 취약점을 증명한다. 따라서 이를 구현하기 위해서는 침투의 원리와 구조를 이해하고 침투 코드의 재사용 및 확장이 용이해야 한다. 하지만 실제 작성된 침투 코드들은 개별적으로 작성되어 있어서 차후 이해하기가 힘들고 재사용 및 확장이 어려운 문제점이 있다.

따라서, 이 논문에서는 이전에 제시한 버퍼 오버플로우 취약점 증명 프레임워크[1]를 확장하여 여러 가지 실제 침투 코드에 대해 적용 가능한 원격 취약점 증명 프레임 워크를 제시하고, 이를 바탕으로 설계 및 구현된 취약점 증명 도구인 HackSim에 대해서 기술한다.

본문 1절에서는 원격 취약점 증명을 위한 프레임워크를 제시하고, 2절에서는 이에 따라 설계된 원격 취약점 증명 도구인 HackSim의 전체 시스템 구조를 알아본다. 그리고, 3절에서는 HackSim의 구현에서 주요 이슈에 대해서 기술하고, 4절에서 HackSim을 이용한 원격 취약점 증명 결과에 대해서 설명한 다음 결론을 맺는다.

II. 본문

1. 원격 취약점 증명 프레임워크

최근에 널리 사용되고 있는 여러 침투코드들을 분석해서 만든 재사용과 확장이 가능한 프레임워크 구조는 그림 1과 같다[1].

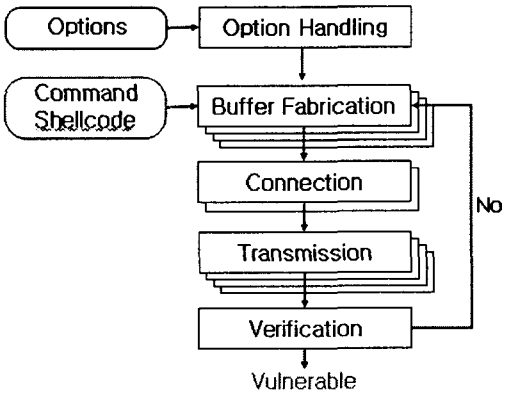


그림 1 원격 취약점 증명 프레임워크

옵션 처리(Option Handling)부분은 GUI 프로그램과의 연동을 지원하기 위해 세부 옵션에 대한 설정을 외부에서 입력받을 수 있게 구현한 부분이다. 옵션을 구현하기 위해서는 각 침투 코드들을 분석해서 구조화된 모듈로 나누고 통합하는 과정이 필요하다.

명령어 전송 쉘코드(Command Shellcode)는 특정 명령어를 실행할 수 있게 작성된 쉘코드 이다. 이것은 동일한 침투 대상 시스템에 대해서는 완전하게 재사용이 가능하다.

버퍼 조작(Buffer Fabrication)부분은 취약점을 지닌 버퍼를 오버플로우 시킬 수 있는 적절한 크기의 버퍼를 구성하고, 쉘코드를 삽입하고 조작된 리턴 어드레스를 기입하는 과정이다. 각각의 데몬에 사용되는 버퍼의 크기 및 구성 형태가 다르므로 버퍼를 생성하는 모듈은 데몬 별로 다르게 이루어져야 한다.

접속(Connection)을 수행하는 부분은 크게 소켓 프로그래밍을 이용한 방식과 원격 접속 절차(RPC)를 이용한 방식으로 나누어져 있다.

전송(Transmission)부분은 침투 대상 프로그램이 여러 번의 입력을 받을 경우, 어떤 버퍼로 조작된 버퍼를 넘겨줄 것인가를 결정하여 실제 버퍼 오버플로우를 일으키도록 버퍼를 전송하는 부분이다.

마지막의 검증(Verification)부분은 침투가 성공하였는지를 확인하는 부분이다. 즉, 조작된 버퍼에 들어있는 쉘코드가 실행되었는지를 확인하는 것이다. 예를 들어, 백도어 포트를 여는 쉘코드의 경우에는 해당 백도어 포트가 열려 있는지를 확인한다.

외부에 화살표로 표시된 부분은 자동화된 침투를 지원하기 위한 부분으로, 조작된 버퍼의 크기나 리턴 주소를 지속적으로 바꾸면서 시도하여 침투의 성공이 이루어지도록 하는 부분이다.

2. 원격 취약점 증명 도구의 구조

이번 절에서는 앞에서 제시한 프레임워크를 기반으로 하여 원격 취약점 증명 도구의 구조를 제시한다.

전체적인 취약점 증명 도구의 구조는 그림2와 같다. 여기에서는 각 부분에서 필요한 기능들을 간략하게 나타냈다. 하위의 프레임워크는 침투에 필요한 옵션을 입력받아 직접 침투를 수행하는 역할을 하고 상위의 GUI 프로그램은 점검자로부터 침투 대상의 범위 및 옵션들을 입력받은 뒤 필요한 침투 명령을 생성하여 프레임워크를 호출하고 그 결과를 정리하여 화면에 보여주는 역할을 한다. 스캐너 부분은 침투를 수행하기 전에 대상 호스트에 대해 사전 점검을 하는 부분인데 호스트

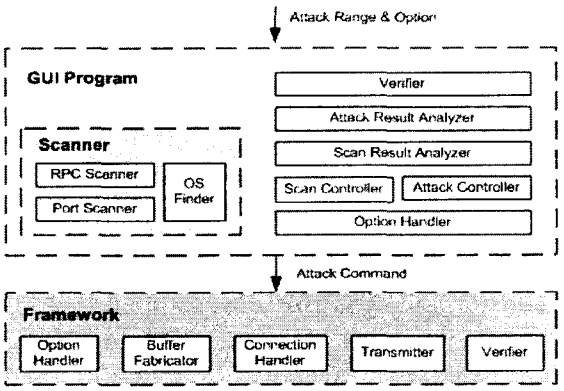


그림 2 원격 취약점 증명 도구의 구조

및 취약한 데몬의 가동여부, OS 버전 등의 정보를 조사하는 역할을 한다. 프레임워크는 GUI 프로그램과 독립적으로 구성해서 프레임워크의 재사용 및 확장을 용이하며 점검 프로그램의 확장 및 유지 보수가 쉽다. 그리고, GUI 프로그램도 각 부분을 모듈화하여 재사용 및 확장이 쉽게 하고 사용자 입력 부분을 제외하고는 점검 과정을 자동화하여 점검을 쉽게 할 수 있다. GUI 프로그램의 상세 구조는 3.2 절에 자세히 기술한다.

3. 원격 취약점 증명 도구의 구현

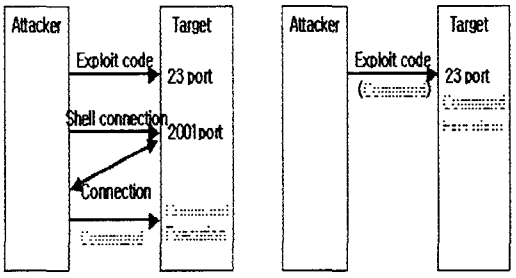
1) 침투 코드의 모듈화 및 통합

본 연구에서 분석에 사용한 침투 코드들은 표1과 같이 다양한 특성을 가졌다. 접속 방법으로는 Socket과 RPC 접속을 사용하고, 침투 방법으로는 버퍼 오버플로우 침투와 힙 오버플로우 침투를 사용한다. 원격 침투에 사용되는 쉘코드로는 bindsocket과 findsocket이 주로 사용되는데 telnetd 데몬에 대한 침투 코드만 bindsocket이 사용되고, 나머지 데몬에 대한 침투 코드는 findsocket이 사용된다[3,4].

데몬	접속 방법	침투 방법	쉘코드
telnetd	Socket	버퍼 오버플로우	bindsocket
cachefsd	RPC	힙 오버플로우	findsocket
snmpXdmid	RPC	버퍼 오버플로우	findsocket
dtspcd	Socket	힙 오버플로우	findsocket

표 1. 실제 원격 취약점 침투 코드의 특성

다양한 쉘코드를 통합하기 위한 방법으로 침투가 성공한 후 원하는 특정 명령어를 자동으로 실행할 수 있는 쉘코드의 작성 및 통합에 대해 연구의 초점을 맞추었다. 이 방법에는 그림 3과 같이 두 가지 방법이 있을 수 있는데 첫 번째 방법은 bindsocket과 findsocket 쉘코드를 사용하여 루트 쉘을 획득한 후, 특정 명령어를 문자열 형태로 호스트에 전송하는 것이다. 두 번째 방법은 루트 쉘을 획득하는 것이 아니라 특정 명령어를 실행할 수 있는 쉘코드를 작성하는 것이다. 첫 번째 방법은 기존의 침투 코드에 명령어 전송 코드만 추가하면 되지만 쉘코드의 재사용이 어렵고 쉘코드의 크기가 크다. 두 번째 방법은 쉘 접속에 필요한 부가적인 코드가 필요 없고, 쉘코드의 크기가 첫 번째 방법에 비해 작다. 본 연구에서는 하나의 명령어 쉘코드(Command Shellcode)로 모든 침투 코드의 쉘코드를 통합하여 침투 코드의 모듈화 및 확장성을 지원한다. 또한, 쉘코드가 통합됨에 따라 침투가 성공하였는지를 검증하는 부분도 쉽게 통합된다.



(a) 문자열 전송 방식 (b) 명령어 쉘코드 방식
그림 3. 특정 명령어 실행 방법

2) 원격 취약점 증명을 위한 GUI

다음과 같은 네 가지 원칙에 기초해서 GUI 프로그램을 구현하였다.

첫째, 플랫폼(Platform) 독립적이어야 한다.

둘째, 다수의 시스템을 대상으로 한 취약점 증명을 지원해야 한다.

셋째, 취약점 증명을 위한 활동을 자동화할 수 있어야 한다.

넷째, 침투의 결과에 대한 정보를 분석하여 이해하기 쉽도록 제공해야 한다.

구현 프로그램의 프레임워크 부분은 C언어로 작성되었고 GUI 부분은 Java로 작성되었다. 구현한 시스템의 전체 구조는 그림 2와 같으며, 그 중 GUI 부분은 그림 4와 같이 동작한다.

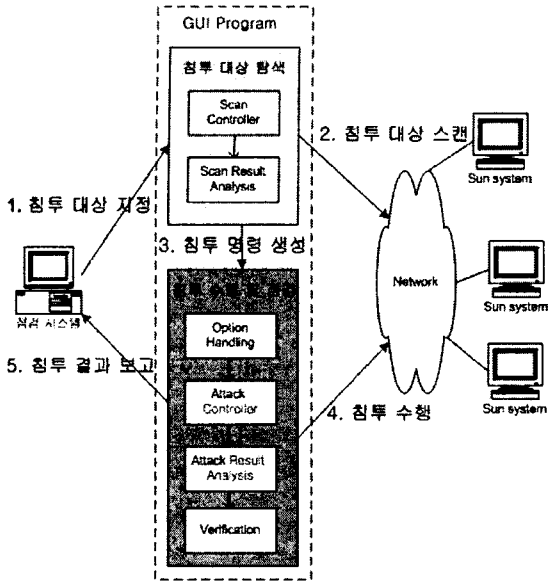


그림 4 취약점 증명 GUI의 작동 방식

침투 대상 호스트 지정부분에서는 취약점 증명을 수행할 호스트를 직접 지정하거나 해당 호스트들이 속한 서브넷을 지정할 수 있다. 여기서는 C-클래스 대역까지 지원한다.

침투 대상 스캔 부분은 프레임워크의 정확성 및 성능 향상을 위해 사용되었다. 실제 침투를 수행하는 경우 침투 프로그램은 brute force 방식을 통해 실행되기 때문에 침투의 성공여부를 알 때까지 많은 시간이 걸리므로 모든 시스템에 대해 침투를 수행하는 것은 매우 비효율적인 일이다. 그래서 여기서는 취약한 프로그램이 수행되고 있는 호스트에 대한 정보를 미리 수집하기 위해서 스캔을 수행한다. 스캔 시에는 우선 대상 호스트에 취약점을 가진 프로그램이 수행되고 있는지를 점검하고 취약한 프로그램이 있을 때는 그 호스트의 운영체제의 버전을 탐지한다. 여기서 운영체제의 버전이 필요한 이유는 버전 별로 리턴 주소의 위치 대역이 다르므로 침투 시 미리 해당 대역을 지정해 줌으로써 침투 수행시간을 줄이기 위해서이다.

침투 명령 생성은 앞서의 스캔 부분에서 얻은 정보를 바탕으로 침투 프로그램의 세부 옵션을 기본 설정값이나 아니면 사용자가 설정한 값을 이용해서 자동으로 세팅하는 부분이다. 사용자가 옵션 값을 입력할 수 있게 함으로써 다양한 침투 코드를 지원할 수 있다.

침투 수행 부분은 주어진 옵션 값을 바탕으로 대상 호스트에 침투를 수행한다. 검증을 용이하게 하기 위해서는 침투를 통해 포트를 오픈하고 사용자가 실제 접속하여 시스템 내의 정보를 수집할 수 있게 하는 게 필요한데 이러한 기능을 지원하는 방법에는 백도어 포트를 오픈하는 방법과 역방향 연결(Reverse Connection)을 이용하는 방법이 있다. 그러나, 백도어 포트 방식은 방화벽에서 백도어 포트가 차단되어 있는 네트워크 내에 있는 호스트를 점검할 수 없고 대상 호스트에 흔적이 남을 수 있기 때문에 여기에서는 역방향 연결 방식을 이용한다. 즉, 역방향 연결 방식에서는 침투가 성공하면 대상 호스트에서 점검 시스템으로 거꾸로 연결을 맺기 때문에 방화벽을 통과할 수 있고 연결을 종료하면 포트가 닫히므로 대상 호스트에 흔적을 남기지 않는 장점이 있다.

침투 결과 보고 부분은 침투 수행부분에서 넘겨받은 결과 값을 바탕으로 전체 수행된 침투에 대해 통계를 내고 각각의 침투 결과를 분석해서 적절한 대응책을 제시한다. 취약점의 침투결과, 위험정도, 패치정보 등을 사용자가 알기 쉽게 알려준다. 그리고 별도의 창을 통해 실제 침투된 호스트에 대한 침투 및 증거 수집 기능을 제공한다.

4. 원격 취약점 증명 결과

그림 5는 구현한 프로그램을 이용해서 원격 취약점 증명을 수행한 결과이다.

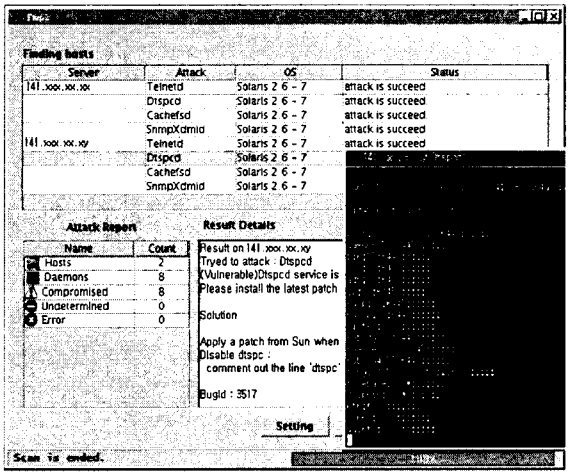


그림 5 원격 취약점 증명 결과

침투를 수행한 대상 호스트들은 SPARC Solaris 7 머신들이고 이미 취약점이 알려져서 최근에도 널리 침투되고 있는 telnetd, dtspcd, cachefsd, snmpXdmid 등 네 가지 원격 데몬에 대해 침투를 수행하였다. 침투 코드를 통합한 프레임워크를 이용하여 쉽게 각각의 침투에 대해서 확

장할 수 있었다. 그리고 옵션 값을 추가할 수 있고 침투 및 증거 수집 과정을 자동화하였다. 침투의 성공/실패 여부는 전면 상태 창에 나타나고 침투가 성공한 경우는 xterm 창을 띄워서 대상 호스트와 Reverse Connection이 맺어진 상태로 대상 호스트의 정보와 /etc/shadow 파일을 화면에 출력함으로써 침투 성공 증거를 제공한다. 그리고 연결이 끊어지지 않은 상태이므로 점검자가 직접 명령을 입력해서 증거 수집 작업을 계속 수행할 수 있다. xterm 창을 닫으면 연결이 끊어지므로 대상 호스트에는 아무런 흔적을 남기지 않는다. 그리고, 보다 정확한 취약점 점검 결과를 알 수 있고 침투를 당했을 때의 위험 정도를 직접 알 수 있으므로 보안에 대한 경각심을 더욱 높여줄 수 있다.

이러한 원격 취약점 증명 도구를 침투 실험에 계속 사용하기 위해서는 새로운 침투 코드에 대한 유지 보수가 중요한데, 현재의 프레임워크는 원격 버퍼 오버플로우 및 힙 오버플로우 침투 유형에 대해서 유지 보수하기 쉽게 구조화되어 있다. 따라서, 앞으로는 새로운 침투 유형에 대한 지속적인 지원이 요구되며, 스팍 솔라리스 플랫폼 뿐만 아니라 다른 플랫폼에 대한 확장도 필요하다.

III. 결론

본 연구에서는 원격 취약점에 대한 실제 침투 코드들을 분석하여 재사용 및 확장이 용이한 원격 취약점 증명 프레임워크를 제시하고 이를 이용하여 실제 원격 취약점 증명에 사용할 수 있는 프로그램을 설계하고 구현하였다. 구현된 프로그램은 스팍 솔라리스 플랫폼의 최근 원격 취약점들에 대해서 동일한 구조로 취약점 증명을 수행할 수 있고 대규모 네트워크에 대한 취약점 증명 및 침투 증거 수집을 수행할 수 있다.

참고문헌

[1] 권오훈, 민병길, 김종, 김수용, 한광택, "버퍼 오버플로우 취약점 증명을 위한 프레임워크", CISC 2002, 2002

[2] Shostack, A., S. Blake., "Toward a Taxonomy of Network Security Assessment Techniques," Proceedings of the 1999 Black Hat Briefings, 1999.

[3] Aleph One, "Smashing The Stack for Fun and Profit," Phrack Magazine, 49(14), 1998.

[4] PLUS(포항공대 유닉스 보안 연구회), "Security PLUS for UNIX", 영진출판사, 2000.