情報保護學會論文誌
第17卷 第2號, 2007. 4

ROAD(RPC Object vulnerability Automatic Detector) 도구의 설계 및 구현

양 진 석,[1†] 김 태 균,[1†] 김 형 천,[1†] 홍 순 좌[1†]
[1]국가보안기술연구소

A Design and Implementataion of ROAD(RPC Object vulnerability Automatic Detector)

Jin-seok Yang,[1†] Tae-ghyoon Kim,[1†] Hyoung-Chun Kim,[1†] Soonjwa Hong[1†]
[1]National Security Research Institute

요 약

소프트웨어 테스팅은 소프트웨어의 버그 및 잘못 구현된 부분 등을 찾아내는 과정을 통해 품질을 평가하는 방법이다. 퍼징(fuzzing)은 소프트웨어 테스팅 기술의 여러 가지 방법 중 하나로써 난수를 발생시켜 테스팅하고자 하는 소프트웨어에 주입하는 방법으로써 보안에 중점을 두어 테스팅하는 방법이다. 퍼징은 단위 시간 당 테스팅 효율성, 비용 절감 등 여러 가지 장점을 이유로 다수 사용되고 있으나 퍼징 수행 시 전문가의 개입이 많은 단점이 존재한다. 예를 들면 해당 소프트웨어가 사용하는 프로토콜 혹은 퍼징 대상이 파일인 경우 파일 포맷에 대한 분석을 수행한 후에야 가능하기 때문에 테스팅 기간이 길어질 수 있으며 퍼징 도구를 이용해도 퍼징 대상의 프로토콜 및 포맷에 대한 분석이 난해한 경우 테스팅 대상에 대한 퍼징을 수행하지 못할 수도 있다.

본 논문에서 설계한 ROAD는 RPC 기반 프로토콜 및 소프트웨어를 자동으로 퍼징할 수 있는 도구이다. RPC는 다수의 취약점이 발견된 구성요소로써 본 논문에서는 이를 자동으로 퍼징할 수 있는 도구의 구현을 목표로 하였다. 기존의 도구 중 RPC 기반 소프트웨어를 퍼징하는 도구가 존재하지만 자동화되어 있지 않을 뿐만 아니라 소프트웨어에 따라 도구를 수정해야만 사용이 가능하다. 본 논문은 이러한 단점을 극복하고자 자동화 도구를 설계 및 구현하여 실제 RPC 기반 프로토콜 및 소프트웨어에 적용하였다. 또한 실험을 통해 도구의 효용성을 검증하였다.

ABSTRACT

Software testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item. [12] A traditional testing focuses on proper functionality, not security testing. Fuzzing is a one of many software testing techniques and security testing. Fuzzing methodology has advantage that low-cost, efficiency and so on. But fuzzing has defects such as intervening experts. Also, if there is no specification, fuzzing is impossible.

ROAD Tool is automated testing tool for RPC(Remote Procedure Call) based protocol and software without specification. Existing tools are semi-automated. Therefore we must modify these tools. In this paper, we design and implement ROAD tool. Also we verify utility in testing results.

Keywords : *Software testing, Fuzzing, Fuzzer, RPC, Windows XP SP2, Vulnerability*

Ⅰ. 서 론

윈도우즈 운영체제는 다수의 사용자가 이용하고 있으므로 운영체제의 취약점이 발견되면 해당 취약점의 파급효과는 치명적이라 할 수 있다.[5] 윈도우즈 운영체제를 개발한 마이크로소프트사도 이에 대한 중요성을 인식하고 있으며 정기적으로 보안패치를 발표하는 등 지속적으로 취약점에 대한 대비책을 마련하고 있다. 윈도우즈 XP SP(서비스팩)2는 보안에 대한 기능을 강화하여 출시된 버전으로써 윈도우즈 방화벽, 웹 브라우저 보안, 메일 보안, 메모리 보호 등 다양한 보안 관련 기능을 업데이트하여 출시하였다.[2] 그럼에도 불구하고 윈도우즈 관련 취약점은 계속해서 발표되고 있으며, 이를 악용하여 공격을 시도하는 사례가 빈번히 발생하고 있다. [그림 1]에서 보는 바와 같이 2005년 운영체제별 침해 사고 중 윈도우즈가 70%를 차지하고 있다.[1]

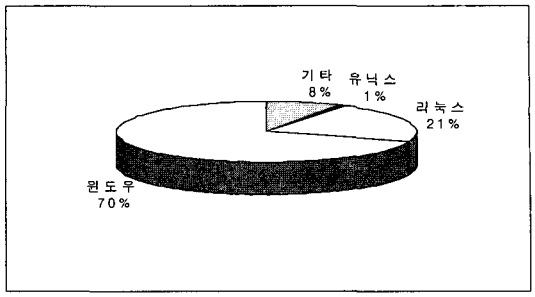
또한 윈도우즈 관련 취약점 현황 자료를 분석한 결과, 특정 구성요소와 관련된 취약점의 발생빈도가 높음을 확인할 수 있다. 바로 RPC 기반 소프트웨어가 그것이다.[1]

윈도우즈에서 제공하는 RPC 기반 소프트웨어는 다른 서비스와 상호 의존성이 매우 크다. 따라서 다른 서비스에 영향을 미치는 파급효과도 매우 클 뿐만 아니라 복잡한 실행 메커니즘을 가지고 있다.

본 논문에서는 상기 기술한 바와 같이 침해 사고가 다수 발생한 윈도우즈 운영체제의 RPC 기반 프로토콜 및 소프트웨어를 취약점 점검 대상으로 하였다. 윈도우즈 운영체제의 RPC 기반 프로토콜 및 소프트웨어에 대한 취약점 점검을 위해 블랙박스 기반의 퍼징(fuzzing) 기법을 도입하였으며 ROAD 도구를 사용하여 퍼징을 수행하였다.

기존의 소프트웨어 보안 테스팅 도구들은 전문가의 개입이 많고 테스팅 대상에 대한 지식 및 노하우가 많아야 한다. 뿐만 아니라 일단 퍼징 대상에 대한 지식을 습득한 후에도 도구 사용법, 무작위 데이터 값의 적절한 조합 등의 복잡한 과정들이 다수 존재한다. 이와 같이 프로토콜 분석 및 구현 과정을 거친 후에 퍼징이 가능하며 테스팅 수행 시 효율이 매우 좋다.

그러나 퍼징 분석 및 구현 과정은 퍼징 대상 및 프로토콜에 대한 많은 지식이 요구되므로 전문가에 의존적인 면이 크다.[8,13] 또한 퍼징 대상에 대한 지식을 습득한 후에도 구현 및 패킷 분석을 동시에 수행해야하므로 실험 기간이 매우 길어지게 된다.



[그림 1] 2005년 운영체제별 침해사고 현황

본 논문에서는 이러한 단점을 해결하기 위해 RPC 기반의 소프트웨어를 자동으로 퍼징할 수 있는 도구를 설계하였다. ROAD는 RPC Object Vulnerability Automatic Detector의 약자로써 RPC 프로토콜을 사용하는 소프트웨어를 자동으로 테스팅할 수 있는 도구이다. ROAD 도구는 기존의 도구에서 수작업으로 수행하던 처리를 자동화하여 사용자 편의성을 극대화하였다. ROAD는 RPC 프로그램의 인터페이스 함수를 정의하는 idl 파일에서 UUID(Universal Unique IDentifier), 인터페이스 버전 등의 정보를 얻어와 자동 테스팅이 가능하도록 한다.

본 논문의 구성은 다음과 같다. 2장은 퍼징 및 관련 도구에 대해 기술하고 3장에서는 ROAD의 설계 및 구현에 대해서 기술한다. 4장은 구현한 ROAD 도구를 실험한 결과에 대해서 기술한다.

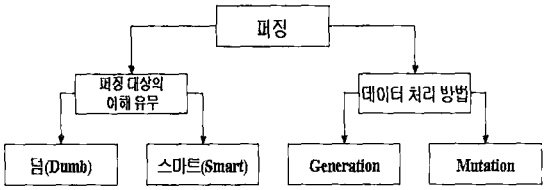
Ⅱ. 관련연구

이번 장에서는 소프트웨어 테스팅을 위한 여러 가지 방법 중 본 논문의 실험에서 사용된 퍼징 기법에 대해서 기술한다. 또한 기존의 퍼징 도구를 분석하였으며 이를 통해 도구의 기능 및 단점을 기술한다.[1]

2.1 퍼징(Fuzzing)

퍼징은 소프트웨어의 알려진 취약성뿐만 아니라 알려지지 않은 취약성을 점검할 수 있는 소프트웨어 테스팅 기술의 여러 가지 방법 중 하나이다. 퍼징의 종류에는 크게 퍼징 대상의 이해 유무에 따라 덤(dumb), 스마트

접수일: 2006년 10월 27일; 채택일: 2006년 12월 14일
† 주저자, jsyang@etri.re.kr
‡ 교신저자, jsyang@etri.re.kr



〔그림 2〕 퍼징의 분류

(smart)로 분류하고 데이터 처리 방법에 따라 generation, mutation으로 나눌 수 있다. 각각에 대해서 자세히 기술하면 다음과 같다.

• 덤(dumb) 퍼징

덤 퍼징은 퍼징을 하고자 하는 대상에 대한 어떠한 기반 지식도 없이 입력을 주어 테스팅을 수행하는 기법을 말한다. 예를 들면, 워드 어플리케이션에 임의의 입력 데이터 혹은 파일들을 무작위로 입력하는 방법을 말한다.

• 스마트(smart) 퍼징

스마트 퍼징은 테스팅을 하고자 하는 대상에 대해 입력 데이터를 처리하기 위한 방법 혹은 올바른 데이터 구조에 대한 정보를 어느 정도 알고 있는 상태에서 테스팅을 수행하는 기법을 말한다. 예를 들면, 워드 문서 포맷과 올바른 처리 과정을 알고 있는 상황에서 오프셋이나 길이 필드에 올바르지 않은 정보를 입력하는 방법을 말한다.

• Generation

Generation은 테스트를 위한 입력 데이터를 새롭게 만들어 처리하는 기법을 말한다. 예를 들면, 도움말 파일 포맷을 알고 있는 상태에서 도움말 보기 어플리케이션을 위해 .chm의 확장자를 갖는 새로운 도움말 파일을 구성하는 방법이 있다.

• Mutation

Mutation은 기존에 올바른 데이터에 대한 샘플들을 얻어와 그 일부를 변형시켜 입력 데이터를 처리하는 기법을 말한다. 예를 들면, 네트워크 outbound 패킷의 몇 바이트를 실시간으로 변경하는 방법이 있다.

2.2 기존의 도구 분석

퍼징 도구는 그 대상에 따라 파일 퍼징, API 퍼징, 프로토콜 퍼징 등으로 나누어지며 〔표 1〕에서 보는 바와 같이 다양한 도구들이 존재한다.[14,16] 이번 소절에서는 이러한 퍼징 도구들의 특징 및 기능에 대해서 알아본다. 또한 잘 알려진 퍼징 도구 중 RPC 퍼징이 가능한

〔표 1〕 잘 알려진 퍼징 도구들

도구 이름	설 명
scratch	SSL, SMB의 퍼징을 위한 프레임워크를 수립
antiparser	여러 가지 데이터 타입을 갖는 파일 포맷이나 네트워크 프로토콜의 api에 대한 퍼징이 목적
peach	COM/ActiveX, SQL, Shared library, 네트워크 애플리케이션을 퍼징할 수 있는 도구
smudge	FTP, SMTP, IMAP, POP3, HTTP, RTSP, MSSQL, DCE RPC, SMB 뿐만 아니라 인터넷 익스플로러 등을 퍼징할 수 있는 도구
fuzzball2	TCP/IP의 옵션을 퍼징하는 도구
ISIC (IP Stack Integrity Checker)	TCP, UDP, ICMP 등의 프로토콜의 안전성 테스팅 도구
IP6sic	ISIC와 비슷한 기능을 가진 도구로써 IPv6 프로토콜 스택을 퍼징하는 도구
efuzz	알려지지 않은 버퍼 오버플로우를 찾아내기 위한 Win32 TCP/ UDP 프로토콜 퍼징 도구
BSS(Blue Tooth Stack Smasher)	L2CAP 계층을 퍼징할 수 있는 도구
radius Fuzzer	RADIUS를 퍼징할 수 있는 도구

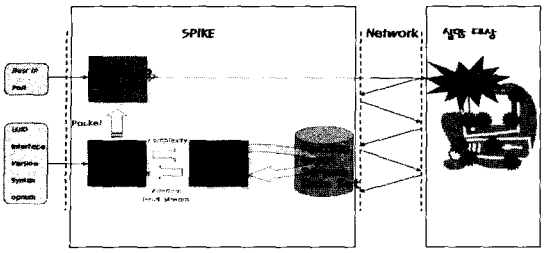
SPIKE에 대해서 더 자세히 알아본다.

상기 기술한 바와 같이 각각의 도구들은 퍼징 대상에 맞게 특화되어 구현되었다. 상기 도구 중 RPC 퍼징이 가능한 도구는 SMUDGE와 SPIKE가 있는데 SMUDGE 도구는 더 이상 진행되고 있지 않아 분석이 불가능하였다.[15] SPIKE 도구는 RPC 프로토콜 및 소프트웨어의 퍼징이 가능한데 이를 위해서 프로토콜 및 소프트웨어의 분석이 필수적이고 분석된 결과로 소스를 수정해야하는 단점이 있다.

본 논문에서는 이러한 단점을 개선하기위해 ROAD 도구를 설계 및 구현하였다. ROAD 도구는 SPIKE 프레임워크를 이용하였으며 이를 위해 SPIKE의 세부 모듈을 분석한다.

2.3 SPIKE [9]

SPIKE는 소프트웨어 테스팅 도구로써 블랙박스 테스팅 기반으로 다양한 프로토콜 및 소프트웨어를 테스



〔그림 3〕 SPIKE 구조

팅 할 수 있도록 구현하였다. SPIKE의 전체 소스 및 데이터 크기는 약 3.5 메가바이트로 225개의 파일로 이루어져 있으며 SPIKE가 테스팅할 수 있는 소프트웨어 및 프로토콜은 CIFS(Common Internet File System), NT LM(NT LAN Manger), POST 메시지, TCP 프로토콜, 웹서버, HTTP/HPPPS 프로토콜, MS/SUN RPC(Remote Procedure Call), DCE(Distributed)/ONC(Open Network Computing) RPC, SMTP 프로토콜, quake 서버, X11 서버 등 약 20여종의 프로토콜 및 애플리케이션을 테스팅할 수 있도록 구현하였다.

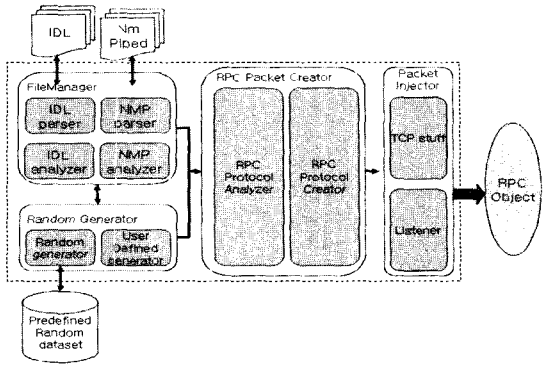
〔그림 3〕에서 보는 바와 같이 SPIKE는 크게 3개의 모듈과 1개의 데이터 셋으로 이루어져 있다. Fuzzer는 여러 가지 프로토콜 및 프로그램을 퍼징할 수 있는 실행 파일로써 약 20여개 프로토콜 및 프로그램을 퍼징할 수 있도록 구현되어 있다. Packet Generator는 무작위 패킷을 생성하는 모듈로써 실행 프로그램에 주입되는 무작위 패킷을 생성하는 역할을 수행한다. TCP/UDP Sender는 통신을 담당하는 모듈로써 실험 대상의 퍼징 포맷에 맞추어 생성된 패킷을 전송하는 역할을 수행한다.

RPC 기반의 애플리케이션 퍼징의 경우 〔그림 3〕에서 보는 바와 같이 퍼징 대상 IP 주소, 포트 번호, UUID, Interface name, version number, syntax, opnum, random number 등을 사용자가 직접 입력하여 실행해야한다. [7]

ROAD 도구는 상기 기술한 바와 같이 RPC 기반 소프트웨어 퍼징 시 사용자가 입력해야하는 부분을 자동화하였다. 다음 소절에서는 이와 같이 자동화된 부분을 포함한 ROAD 도구 전체 모듈을 설계한다.

Ⅲ. 도구 설계

이번 장에서는 RPC 기반의 프로토콜 및 소프트웨어를 자동으로 퍼징할 수 있는 ROAD 도구에 대한 설계



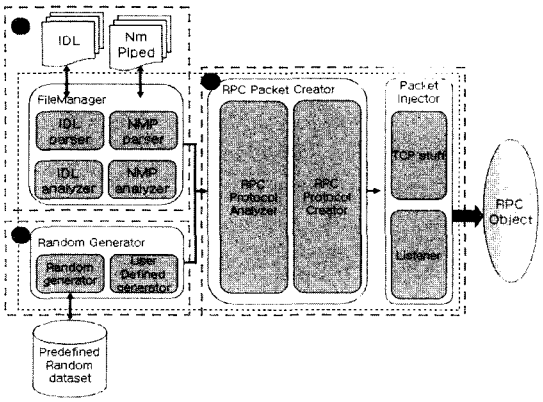
〔그림 4〕 ROAD 구조

를 한다. ROAD는 테스트 전문가의 개입을 최소화하기 위해 idl 파일을 파싱하여 필요한 정보를 추출하여, 이를 기반으로 RPC 기반의 소프트웨어를 자동으로 퍼징할 수 있도록 설계하였다. 또한 확장성을 위해 RPC 기반 통신 시 필요한 Named Pipe에 대한 정보를 데이터베이스화하여, 새로운 RPC 서비스 퍼징 시 해당 Named Pipe를 데이터베이스에 업데이트 하면 퍼징이 가능하도록 하였다.

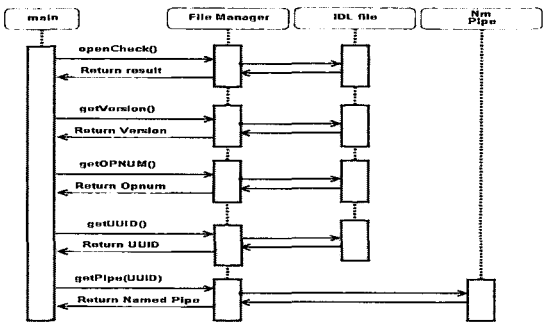
3.1 ROAD 구조

ROAD는 크게 File Manager, Random Generator, RPC Packet Creator, Packet Injector 4개의 모듈과 idl, NMPipe, random data 3개의 데이터로 이루어져 있다.

- File Manager
  - IDL parser/analyzer : test.idl 파일을 파싱하여 UUID, Version, OP(OPeration) Number와 같은 자동으로 실행할 시 필요한 정보를 얻어오는 처리를 수행함
  - NMP(NaMed Pipe) parser/analyzer : Named Pipe 정보를 얻어오기 위해 미리 정의한 NamedPipe.txt 파일을 파싱하여 해당 UUID와 일치하는 Named Pipe의 정보를 얻어오는 처리를 수행함
- Random Generator
  - Random generator : 소프트웨어 테스팅 시 사용할 입력 값을 만들어내는 모듈
  - User Defined generator : 사용자가 지정하여 random하게 입력을 줄 수 있도록 처리하는 모듈
- RPC Packet Creator



〔그림 5〕 ROAD의 전체 순서도



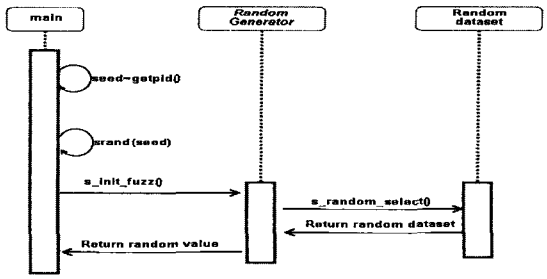
〔그림 6〕 File Manager의 처리 순서

- RPC Protocol Analyzer : RPC 통신을 위해 사용되는 smb 및 dce rpc 프로토콜을 재조합하기 전 검사하는 모듈
- RPC Protocol Creator : RPC 통신을 하기 위한 패킷을 재조합하는 모듈

• Packet Injector
- TCP stuff : Socket 생성, 바인드, 리슨, 어셉트 등 TCP 통신을 수행하기 위한 모듈
- Listener : 퍼징 시 사용하는 데이터 구조를 정의하고 핸들링하는 함수를 구현한 모듈

3.2 ROAD 처리 순서

ROAD는 〔그림 5〕에서 보는 바와 같이 크게 3개의 실행 단계로 나눌 수 있다. 첫 번째는 idl 파일 및 Named pipe의 정보를 얻어오는 단계, 두 번째는 무작위 값을 생성하고 선택하는 단계, 세 번째는 1,2단계에서 얻은 정보로 RPC 패킷을 생성하고 전송하는 단계이다. 각 단계별의 상세한 설명은 다음과 같다.



〔그림 7〕 Random Generator의 처리 순서

3.2.1 File Manager 처리 순서

FileManager는 idl 파일에 대한 정보를 얻어오기 위해 해당 idl 파일을 파싱하고 ROAD 프로그램이 실행될 때 필요한 정보를 얻어오는 역할을 수행한다. File Manager의 실행 순서는 다음과 같다.

1. openCheck() 함수를 호출하여 testing.idl 파일 존재 유무 검사
2. getVersion() 함수를 호출하여 인터페이스 버전 정보를 획득
3. getOPNUM() 함수를 호출하여 인터페이스 OP number를 획득
4. getUUID() 함수를 호출하여 인터페이스 UUID를 획득
5. getPipe() 함수는 getUUID() 함수의 결과 값을 인자로 하여 UUID에 해당되는 Named pipe 정보 획득

3.2.2 Random Generator 처리 순서

Random Generator는 무작위 값을 생성하는 모듈이다. 생성된 무작위 값은 퍼징 대상이 되는 서비스의 인자 혹은 프로토콜 필드 등 퍼징하고자 하는 대상에 입력된다. Random Generator의 실행 순서는 다음과 같다.

1. getpid() 함수를 호출하여 랜덤함수에 사용할 seed를 생성
2. srand() 함수는 seed를 인자로 난수 값을 생성
3. s_init_fuzz() 함수를 호출하여 기존의 취약한 문자열이나 다양한 난수 값을 버퍼에 저장
4. s_random_select() 함수를 이용하여 버퍼에 미리 생성된 난수 값 중 무작위로 선택

상기 난수 함수는 SPIKE 도구에서 제공하는 함수이

며 난수를 발생하는 함수 이외에 또 다른 입력 값을 생성하는 함수는 다음과 같다.

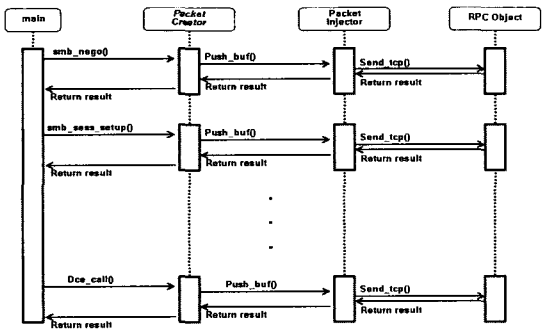
사용자 정의의 입력 값 생성 함수 = f(x ◦ y)
◦ : concatenate 연산자
x = { 0, 1, 2, ... ,a, f }
y : { 0, 1, 2, ... ,a, f }

상기 함수에서 x는 입력되는 한 바이트의 상위 4비트이고 y는 하위 4비트의 입력 값이다. 값을 각각의 값을 생성한 후 concatenate 연산자로 상위 4비트와 하위 4비트를 연결시켜 8비트로 만든 후 입력한다.

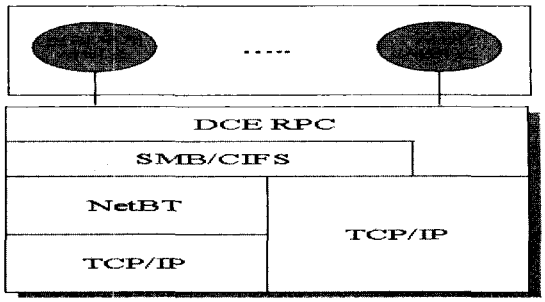
3.2.3 RPC Packet Creator/Packet Injector 처리 순서

RPC Packet Creator는 RPC 통신을 하기 위한 패킷을 생성하는 모듈로써 RPC 기반의 프로그램과의 통신을 위한 패킷을 File Manager와 Random Generator 모듈에서 얻은 값들을 버퍼에 저장하는 역할을 수행한다. Packet Injector는 버퍼에 저장된 무작위 값을 퍼징 대상 객체에 전송하는 역할을 수행한다. RPC Packet Creator 및 Packet Injector의 실행 순서는 다음과 같다.

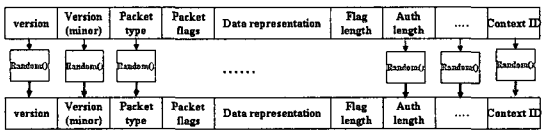
1. Main 함수에서 smb_nego() 함수를 호출하여 RPC 퍼징 대상 시스템과 SMB 프로토콜의 negotiation 시작
2. Packet Creator는 SMB negotiation 패킷을 생성하고 push_buf() 함수를 호출하여 해당 패킷을 버퍼에 저장
3. Packet Injector는 버퍼에 저장된 패킷을 전송하기 위해 Send_tcp() 함수를 호출하여 퍼징 대상 프로그램에 패킷을 전송
4. 퍼징 대상 프로그램은 전송 받은 패킷을 처리하고 응답을 수행
5. RPC 기반 프로그램과 통신을 하기 위한 함수들을 계속해서 호출하고 마지막 호출 함수에 이미 생성된 무작위 값을 입력하여 퍼징을 수행



〔그림 8〕 Packet Creator/Packet Injector의 처리 순서



〔그림 9〕 RPC 기반 소프트웨어의 프로토콜 스택



〔그림 10〕 난수 주입의 예

Ⅳ. 결과 분석

4장은 설계 및 구현한 ROAD 도구의 RPC 기반 프로토콜 및 소프트웨어에 대한 퍼징 수행 결과를 기술하였다.

ROAD 도구는 〔그림 9〕의 RPC 기반 소프트웨어의 프로토콜 스택에서 보는 바와 같이 SMB, DCE RPC, 브라우저 서비스 및 스풀러 서비스, 서버 서비스, 워크스테이션 서비스 등 RPC 기반 소프트웨어를 덤 퍼징하는 기능을 가진다.

〔그림 10〕은 DCE RPC 바인드 패킷에 난수를 주입하는 예를 보여준다. DCE RPC 바인드 패킷은 19개의 필드로 이루어져 있다.

〔표 2〕는 ROAD 도구를 이용한 윈도우 서비스 중 4개

〔표 2〕 DCE RPC 패킷에 대한 실험 결과

퍼징 대상 패킷	바인드된 서비스	총 주입 횟수	발견한 취약점
DCE RPC 바인드 패킷	브라우저	19,000회	3종
	서버	19,000회	발견하지 못함
	워크스테이션	19,000회	발견하지 못함
	스풀러	19,000회	1종

〔표 3〕 SPIKE와 ROAD 실험 데이터 비교

	SPIKE	ROAD
퍼징 대상	DCE RPC패킷	DCE RPC패킷
도구 구현 및 실험 기간	약 10일 이상	약 4일
패킷 주입회수	400,000회 이상	약 76, 000회
발견한 취약점	1종 [10]	4종

서비스의 DCE RPC 바인드 패킷에 실험 결과를 나타낸다. 바인드 패킷 당 각각 19,000회를 주입하여 알려진 취약점 3종 [3,4,6] 과 알려지지 않은 취약점 1종을 발견하였다.

〔표 3〕은 SPIKE와 ROAD의 실험 데이터를 비교한 내용이다. 퍼징 대상은 DCE RPC 바인드 패킷으로 하였다. 도구 구현 및 실험 기간에서 SPIKE의 경우는 소스의 여러 부분을 수정해야하고 해당 서비스가 바인드되는지 확인한 후 퍼징을 실행해야하는 단점이 있는 반면에 ROAD 도구는 해당 서비스의 IDL 파일만 복사하면 실행되므로 구현 및 실험 기간이 단축된다.

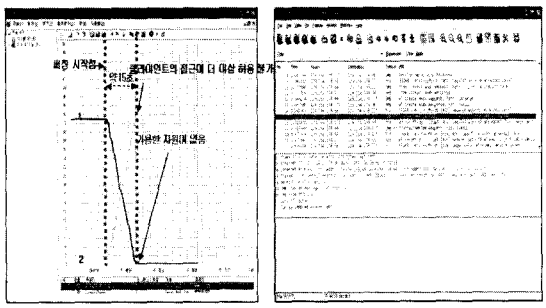
또한 3장 2절에서 정의한 사용자 정의의 입력 함수인 f(x ◦ y)를 이용하여 입력 값을 주입하였다. 이와 같이 실험한 결과 기존의 취약점과 알려지지 않은 새로운 취약점을 포함한 4종의 취약점을 발견하였다. 기존의 SPIKE는 Windows 취약점 1종을 발견하였으며 [10], 이를 발견하기 위해 SPIKE를 수정하여 테스트한 결과 400,000회 이상의 주입 패킷과 10일 이상의 구현 및 실험 기간을 필요로 하였다.

ROAD 도구를 통해 발견한 알려지지 않은 취약점의 파급효과는 실험을 통해 검증하였으며 다음과 같은 파급효과를 나타낸다.

- 파일 공유 서비스 장애
- 작업 그룹 컴퓨터 보기 장애
- 프린터 서비스 장애
- 액티브디렉토리 환경에서 원격관리서비스 장애

알려지지 않은 취약점의 원인은 〔그림 11〕에서 보는 바와 같이 perfmon 도구와 ethereal로 분석한 결과, 잘못된(invalid) 입력 값을 주입한 후 회복 함수가 제대로 호출되지 않아 가용한 자원이 소진되는 현상을 보였다.

알려지지 않은 취약점은 perfmon 모니터링 도구의 available work item(1번 라인) 항목과 current client(2번 라인) 항목으로 모니터링하였다. 〔그림 11〕에서 보는 바와 같이 퍼징 시작 후 가용한 자원을 나타내는 available



〔그림 11〕 perfmon과 ethereal을 이용한 모니터링

work item 항목은 급격히 떨어지고 current client 항목은 더 이상 올라가지 않는 그래프를 나타내고 있다. 정상적인 경우라면 적당한 자원을 할당하며 client의 요청을 계속해서 처리한다. Ethereal로 모니터링한 화면은 퍼징 시작 후 패킷을 모니터링하는 화면이다. 정상적인 경우에는 TranctNmPipe 함수 호출 후 자원을 해제하는 함수가 호출되어야 하는데 자원을 회복하는 함수가 호출되지 않아 DoS 증상을 일으킨다.

알려지지 않은 취약점은 윈도우의 여러 버전에 테스팅하여 Windows 2000과 Windows XP SP2에서 문제가 있음을 확인하였으며 이를 마이크로소프트사에 통보하였다.

V. 결론

본 논문에서는 기존의 퍼징 도구를 사용할 시 전문가의 개입이 많다는 단점을 극복하기 위해 RPC 기반의 소프트웨어를 자동으로 퍼징할 수 있는 ROAD 도구의 설계, 구현 및 테스팅 결과에 대한 내용을 기술하였다.

ROAD는 RPC 기반 소프트웨어 퍼징 시 전문가가 입력해야하는 부분을 자동화하여 기존의 도구와 차별화를 두었다. 즉, UUID, Interface name, version number, syntax, opnum, random number와 같은 정보를 idl 파일에서 얻어와 패킷 생성 시 해당정보들을 입력 값으로 하여 자동으로 테스팅을 가능하게 한다.

본 논문에서는 설계 및 구현한 ROAD 도구의 효용성을 검증하기 위해 윈도우 계열의 RPC 기반 프로토콜 및 서비스를 퍼징하였다. 실험을 통해 알려진 취약점과 알려지지 않은 취약점을 발견하였으며 알려지지 않은 취약점은 마이크로소프트에 통보한 상태이다.

결과 분석을 통해 ROAD 도구는 RPC 기반 프로그램의 퍼징 시 유용하게 사용될 것으로 기대된다.

참고문헌

[1] 국가사이버안전센타, "2006년 국가정보보호백서", 2006년 5월.

[2] 빌게이츠, "마이크로소프트의 보안기술 발전에 대한 보고", 마이크로소프트, 2004년 3월.

[3] 원격 데스크톱 프로토콜의 취약점으로 인한 서비스 거부 문제점, http://www.microsoft.com/korea/technet/security/bulletin/MS05-041.mspx, 마이크로소프트, 2005년 8월.

[4] 인쇄스풀러 서비스의 취약점으로 인한 원격 코드 실행 문제점, http://www.microsoft.com/korea/technet/security/bulletin/MS05-043.mspx, 마이크로소프트, 2005년 8월.

[5] Onestat 닷컴 홈페이지, "Microsoft's Windows dominates the OS market on the web according to OneStat.com", Onestat 닷컴, 2006년 8월.

[6] SMB의 잘못된 핸들 취약점, http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-2374, Common Vulnerabilities and Exposures, May 2006.

[7] Bloomer, Power Programming with RPC, O'Reilly & Associates, Feb. 1992.

[8] Charles Shelton, Philip Koopman and Kobey DeVale, "Robustness Testing of the Microsoft Win32 API," DSN2000, Jun. 25, 2000.

[9] Dave Aitel, "An Introduction to SPIKE, the Fuzzer Creation Kit", immunity inc. white paper, Jan. 2004.

[10] Dave Aitel, "Advanced Windows Exploitation," immunity inc. white paper, May 2003.

[11] Ivan Medvedev, "Security Tools for Software Development", microsoft corp. white paper. Apr. 2005.

[12] IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.

[13] John P. Devale, Philip J. Koopman, David J. Guttendorf, "The Ballista Software Robustness Testing Service," Testing Computer Software Conference, Nov. 1999.

[14] Matthew Franz, Fuzzing Tools, http://www.scadasec.net/secwiki/FuzzingTools, Nov. 2006.

[15] Greg hoglund, SMUDGE: protocol fault injector in python, http://www.rootkit.com/newsread.php?newsid=113, Apr. 2004.

[16] Jack Koziol, "Fuzzers: The ultimate list," http://www.infosecinstitute.com/blog/2005/12/fuzzers-ultimate-list.html, Dec. 2005.

〈著者紹介〉

양 진 석 (Jin-seok Yang)
2003년 2월: 성균관대학교 정보공학과 졸업
2005년 2월: 성균관대학교 컴퓨터공학과 석사
2005년 3월~현재: 국가보안기술연구소
<관심분야> 소프트웨어 테스팅, 네트워크 보안, 침입 감내

김 태 균 (Tae-ghyoon Kim)
1995년 2월 충남대학교 전자공학과 졸업
1997년 2월 충남대학교 전자공학과 석사
1997년 2월~2004년 8월 한국마이크로소프트 연구소
2004년 8월~현재: 국가보안기술연구소
<관심분야> 소프트웨어 테스팅, 정보보호, 통신공학

김 형 천 (Hyoung-chun Kim)
1999년 8월 고려대학교 전산학과 졸업
2001년 8월 고려대학교 전산학과 석사
2001년 3월~현재: 국가보안기술연구소 선임연구원
<관심분야> 시스템 보안, 소프트웨어 테스팅, 데이터마이닝

홍 순 좌 (Soonjwa Hong)
1989년 2월 숭실대학교 전자계산학과 졸업
1991년 2월 숭실대학교 전자계산학과 석사
1991년 3월~2000년 국방과학연구소 선임연구원
2005년 3월~현재: 국가보안기술연구소 팀장/선임연구원
<관심분야> 컴퓨터 보안, 유/무선 통신 보안