DOI: 10.3745/KIPSTD.2009.16D.6.921

톨레미를 이용한 DNS 시스템 모델 기반의 효율적 취약성 탐지

신 승 훈[†] · 박 승 규[††] · 정 기 현[†††]

요 약

소프트웨어가 이미 배포되어 사용되고 있는 경우, 상황에 따라 소프트웨어에 내재된 취약성은 심각한 사회적, 경제적 영향을 미칠 수 있다. 그러므로 소프트웨어의 취약성은 개발 단계에서부터 충분한 고려가 필요하다. 소프트웨어의 모델 및 시뮬레이션은 소프트웨어의 개발 단계에서 사용될 수 있는 취약성 검사를 위한 테스트 정책의 하나로 고려될 수 있다. 본 논문에서는 해당 방법의 사용 가능성 확인을 위해 톨레미를 이용하여 DNS 시스템의 행동 양식을 모델링하고 이를 시뮬레이션 하였다. 실험 결과에 따르면 기존에 알려진 DNS 서버의 취약성이 효과적으로 검출되고 있음을 확인할 수 있었고, 이는 모델 및 시뮬레이션이 취약성 테스팅에 사용 가능함을 의미한다.

키워드 : 모델 테스트, 소프트웨어 취약성, 톨레미, DNS 중간자 공격

An Efficient Searching of Vulnerabilities Based on a DNS System Model using Ptolemy

Shin, Seung-Hun[†] · Park, Seung-Kyu[††] · Jung, Ki-Hyun[†††]

ABSTRACT

Vulnerabilities in software can result in many social and economic problems once it has already been deployed and put to use. Thus, the vulnerabilities should be seriously taken into consideration from the beginning step of software development. A modeling and simulation method for software can be adopted as a testing tool for establishing vulnerability inspection strategies. For verification of usability of this strategy, in this paper, we modeled the behavior of a DNS system using Ptolemy and the simulation was performed. The result shows that a well-known vulnerability of DNS server could be effectively found, which confirms that the modeling and simulation can be used for vulnerability testing.

Keywords : Model Test, Software Vulnerability, Ptolemy, DNS MITM Attack

1. 서 론

소프트웨어 테스팅은 소프트웨어 개발 과정에서 매우 중요한 역할을 담당한다. 이에 따라 소프트웨어 개발에 소요되는 비용 중 반 이상이 소프트웨어 테스팅에 사용됨에도 불구하고[8], 소프트웨어가 가지는 오류나 취약성의 완벽한 탐지 및 제거는 거의 불가능한 일이기 때문에 소프트웨어가 배포되어 사용되는 시점에서 충돌 혹은 연산 결과 오류 등의 형태로 사용자 등에 의해 발견된다. 이 중 소프트웨어의 취약성은 IT 환경의 기술적, 인적 및 프로세스에서 기인하며[9], 그 중 주원인은 소프트웨어의 구현 과정에 있다[6]. 즉, 소프트웨어 설계 시의 오류나 고려되지 못한 사항은 소프트웨어가 취약성을 가지는 주원인 중 하나가 될 수 있음을 의미한다. 하지만 설계 시에 잠재되는 취약성은 일반적인 테스팅이나 운용 환경에서는 발견하기 어렵고, 특히 소프트웨어가 배포되어 다양한 버전을 가지고 이종 시스템과 네트워크를 통해 연동되어 사용되는 경우의 취약성 파악은 많은 어려움을 수반하게 된다. 따라서 소프트웨어의 모델을 작성하고 이를 시뮬레이션 하는 방법은 소프트웨어의 구현 이전 단계에서 수행 가능한 취약성 검사의 한 방법이 될 수 있다. 따라서 본 논문에서는 이와 같은 방법의 사용 가능성 확인을 위해 기존 소프트웨어를 해당 소프트웨어의 행동 양식(behavior)을 바탕으로 모델링하고 이를 시뮬레이션 하여 기존에 알려진 해당 소프트웨어의 취약성을 재연한다. 이 과정에서 모델링과 시뮬레이션을 위해서 톨레미(Ptolemy) II

† 준 회 원 : 아주대학교 정보통신공학과 박사과정
†† 정 회 원 : 아주대학교 정보 및 컴퓨터공학부 교수
††† 정 회 원 : 아주대학교 전자공학부 교수
논문접수 : 2009년 9월 16일
수 정 일 : 1차 2009년 11월 4일, 2차 2009년 11월 16일
심사완료 : 2009년 11월 18일

를, 모델링 대상으로는 DNS 시스템을 이용하며, DNS 시스템 공격 모델에는 메시지 스니핑(sniffing) 대신 임의의 데이터 이용을 포함하는 DNS 질의/응답 정보 기반 공격 모델을 사용한다.

2. 관련 연구

DNS는 인터넷 보급률이 증가함에 따라 공격자의 중요한 공격 목표가 되고 있을 뿐만 아니라 DNS 정보의 위조 및 변조 시 그 영향이 사회 및 경제에까지 피해를 주게 되므로 [3], 다수의 서비스 취약성 및 보호 방법에 대한 연구가 진행되어 왔다. [3]에서는 DNS 캐쉬 오염 공격 방법 분석을 통해 DNS가 반복 질의 검사, 패킷 검사, 공격 탐지 기능 수행 및 캐쉬 검증 기능을 수행하도록 하여 DNS가 독립적으로 정보의 오염 여부를 실시간으로 탐지할 수 있는 방법을 제안하고 있으며, [1]에서는 악의적인 HTTP(Hypertext Transfer Protocol)를 통한 DNS 서버 공격을 예로 들어 침입 감내 시스템의 결함 모델을 작성하고, 시스템의 결함 허용성을 증가시켜 시스템에 대한 공격 시 시스템의 가용성을 증가시키는 방법을 제안했다. 하지만 이들 연구에서는 소프트웨어가 가지는 미지의 취약성에 대한 고려는 반영되지 않았다.

본 논문에서 소프트웨어의 행동 방식 모델링 및 시뮬레이션에 이용하는 톨레미 II는 UC Berkeley에서 톨레미 프로젝트를 통해 개발한 모델링, 시뮬레이션 및 병행(concurrent), 실시간, 임베디드 시스템의 디자인을 위한 자바(Java) 기반의 프레임워크이다. 톨레미 II는 현재 7.0.1 버전까지 제공되고 있으며, 모델 내에 정의되어 단위 기능을 수행하는 엔티티 및 엔티티 간의 동작 방식을 연산 모델(model of computation) 혹은 도메인(domain)으로 규정하고, 동작 방식의 형태에 따라 도메인을 다양하게 정의하고 있다. 현재는 연속 시간 모델링(continuous-time modeling, CT), 프로세스 네트워크(process network with asynchronous message passing, PN) 및 무선(wireless) 등 9개의 정규 도메인과 이산 시간(discrete time, DT), 분산 이산 이벤트(distributed discrete events, DDE) 및 3D 그래픽스 등 9개의 실험적 도메인을 제공하고 있다. 이중 PN 도메인은 기본적으로 하드웨어와 임베디드 애플리케이션 등 높은 수준의 병행 수준을 요구하는 도메인을 위해 작성된 것으로, 모델은 순차적 프로세스의 네트워크를 형성하고, 모델 내의 각 엔티티가 별도의 프로세스로 동작하며, 이들 프로세스 사이의 통신은 비동기 방식을 사용한다. 모델에 적용되는 도메인의 결정은 모델 내에 추가되는 도메인 액터 즉, 디렉터(Director)에 의해 결정된다[5].

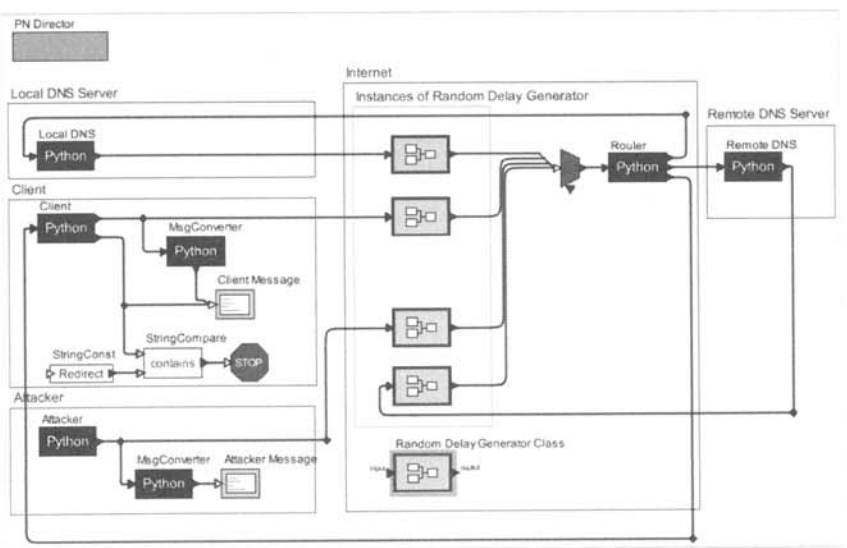
이와 같은 다양한 종류의 도메인은 도메인별 고유 특성이 고려된 액터(actor)를 포함하며, 이외에도 추가 라이브러리에 그래픽, 파이썬(Python) 코드 및 매트랩(Matlab) 표현식 외 다수의 다양한 환경을 고려한 액터들을 포함하고 있다. 이에 추가로 톨레미는 사용자가 직접 자신의 액터를 작성하여 등록할 수 있는 환경을 제공하기 때문에, 톨레미에서 제공되는 도메인 내에서는 소프트웨어나 하드웨어의 행동 양식을 모델 작성자가 대상으로 하는 수준으로 표현 가능하다.

본 논문에서는 네트워크 모델 및 구성 시스템의 모델 작성에 가장 적합한 도메인으로 판단되는 프로세스 네트워크(Process Network, PN) 도메인을 이용하여, 시뮬레이션을 위한 모델을 작성한다.

3. 시스템 모델 및 DNS 공격 모델

3.1 모델 구성

DNS 시스템의 모델링은 기존에 작성된 DNS 시스템 모델[2]을 바탕으로, 중간자 공격을 수행하는 공격자를 추가하여 (그림 1)에 나타난 것과 같은 형태로 구성한다. 모델은 크게 2개의 DNS 서버 액터와 하나의 클라이언트 및 하나의



(그림 1) 시스템 모델

공격자로 구성되고, 각각은 메시지 교환기 역할을 수행하는 인터넷 액터 클러스터에 연결된다. (그림 1)에 나타난 노드들은 실제 시스템의 기능을 간략화 하여 표현한 것으로, 합성 액터 내부에는 기능 수행에 필요한 액터들이 계층구조를 이루도록 추가되었다. 본 논문에서 사용하는 DNS 시스템은 단순화된 형태를 취하고 있으나, 이와 같은 구조가 주는 장점은 기존의 액터를 새로운 행동 양식을 가지는 버전의 액터로 교체가 쉽고, 다수의 복제 액터를 생성해 복잡한 네트워크를 구성하기 용이하며, 다양한 버전을 가진 액터를 동시에 시뮬레이션이 가능해 복잡한 이종 시스템이 연동되는 구조를 표현 가능하다는 것이다. 본 논문에서 사용하는 각 구성 액터의 구조 및 기능은 다음과 같다.

인터넷 액터는 크게 메시지 분배 기능을 수행하는 라우터 액터와 각 메시지에 임의의 지연 시간을 부여하는 임의 시간 지연기(Random Dealy Generator)로 구성된다. 라우터 액터는 인터넷 액터 클러스터에 전달되는 모든 메시지를 파싱하여 메시지 내의 주소를 기준으로 목적 액터를 찾아낸 후, 해당 메시지를 목적 액터를 대상으로 하는 인터페이스로 보내 메시지 교환을 수행하며, 이 때 사용되는 주소는 IP 주소를 단순화 하여 모델 내의 각 시스템들에 부여한 정수 형태의 주소가 사용된다. 한편 임의 시간 지연기와 라우터 사이에는 비결정적 합병(Non-deterministic merge) 액터를 추가하여 라우터에 전달되는 메시지가 인터넷 액터의 인터페이스 배치에 의한 결정론적 순서를 따르지 않도록 하였다.

임의 시간 지연기(Random Delay Generator) 클래스는 가우시안 분포(Gaussian Distribution)에 따라 난수를 만들어 내는 액터를 포함하는 난수 생성 합성 액터를 생성해 구성되었으며, 이를 이용해 이를 지나는 모든 메시지가 0~300mSec의 임의 지연 시간을 갖도록 했다.

클라이언트 액터는 정상적인 DNS 질의를 DNS 서버에 전송하는 기능을 수행하며, DNS 서버에서 응답으로 보내오는 호스트의 주소는 클라이언트의 캐쉬에 TTL(Time To Live)로 정의된 시간동안 저장된다. 이 때 존재하지 않는 호스트에 대한 캐쉬(negative cache)는 수행하지 않는 것을 가정하였다. 클라이언트 액터는 미리 작성된 리스트에 존재하는 호스트 가운데 임의의 호스트를 선택하여 해당 호스트의 주소를 내부 DNS에 질의한다. 클라이언트 액터에서 DNS 서버로 보내지는 질의 메시지 오브젝트는 메시지 변환기(Message Converter)에 의해 파싱이 수행되어 기본적인 정보가 콘솔에 출력되며, DNS 서버로부터의 응답에 따라 발생되는 호스트로의 연결 시도는 클라이언트 액터에 의해 처리되어 콘솔에 출력된다. 이러한 과정은 DNS 서버로부터의 응답 메시지에 공격자가 클라이언트를 유도하기 위해 사용하는 호스트의 주소가 포함되어 있을 때까지 반복 수행된다.

DNS와 공격자 기능을 수행하는 액터는 모든 기능을 파이선 스크립트 액터를 이용해 구성하였으며, DNS 액터는 도메인 네임 리소스 관리를 위한 테이블 및 타 DNS 서버와의 트랜잭션 관리를 위한 테이블을 포함한다. 도메인 네임 테이블에 저장되는 각각의 도메인 네임 리소스는 TTL (Time To Live) 파라미터를 가져 일정 시간동안만 테이블 내에 보관되도록 하였으며, 트랜잭션 관리 테이블은 해당 트랜잭션에 대한 트랜잭션 ID와 UDP 포트 번호 등의 정보를 보관하게 되고, 이를 이용해 타 DNS 서버로부터 전달되는 응답의 진위 여부를 판단한다.

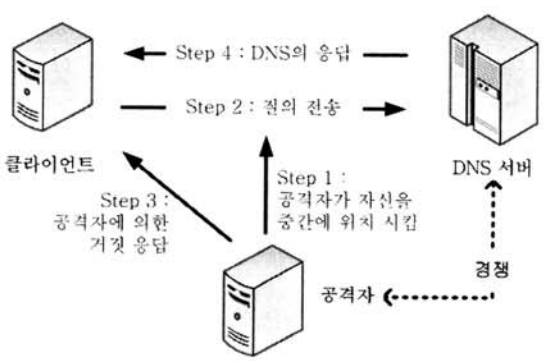
도메인 네임의 질의 과정에 수반되는 루트(root) 네임 서버를 경유한 관할(Authoritative) DNS 조회 과정은 모델의 단순화를 위해 생략하였으며, 따라서 타 도메인 호스트에 대한 질의가 수신되면 바로 상대 DNS 서버에 전달되도록 구성하였다. 한편 공격자 액터는 기존에 알고 있는 DNS 질의/응답 절차와 랜덤 데이터를 바탕으로 트랜잭션 ID와 UPD 포트 번호를 생성해 DNS 응답 메시지를 작성 및 클라이언트 액터에 전송하는 기능을 수행한다. 작성된 모델에서 공격자는 클라이언트의 질의 메시지를 스니핑 하지 않는 것으로 가정하였기 때문에, 세 가지 종류의 트랜잭션 ID 및 UDP 포트 번호 쌍을 생성하는 방법을 이용해 공격에 이용하도록 하였다.

3.2 DNS 중간자 공격 모델

(그림 2)는 DNS 서버에 대한 중간자 (Man In The Middle, MITM) 공격 방식 가운데 DNS 하이재킹(Hijacking)을 나타내고 있다[4].

DNS 하이재킹 공격은 공격자가 자신을 DNS 서버와 클라이언트 사이에 위치시켜 클라이언트의 메시지를 가로챌 준비를 하는 것으로부터 시작한다(step 1). 이 상태에서 클라이언트가 DNS 서버로 특정 호스트의 IP를 묻는 질의를 전송하면(step 2), 공격자는 해당 질의를 가로채 질의 내의 내용을 참조하여 가짜 응답 메시지를 작성한다. 이 때 응답 메시지에 클라이언트를 유인하기 위한 IP를 실어 클라이언트로 전송한다(step 3). 한편 클라이언트의 질의를 수신한 DNS 서버도 정상적인 IP 주소를 가진 응답 메시지를 전송하게 되는데(step 4), 클라이언트에서는 공격자가 보낸 가짜 응답 메시지와 DNS가 보낸 정상 응답 중 먼저 도착하는 쪽의 메시지를 수용하기 때문에, 클라이언트에서 질의를 보낸 순간부터 공격자와 DNS 서버는 경쟁 상태에 들어가는 구조를 가진다.

(그림 2)에 나타난 DNS 중간자 공격은 공격자가 클라이



(그림 2) DNS 중간자 공격

언트의 메시지를 가로채기 때문에 공격이 DNS와 메시지 전송 속도 경쟁 수준의 난이도만을 가지므로 매우 용이한 공격 유형에 속한다. 하지만 본 논문에서는 클라이언트 메시지의 스니핑 없이 간단한 지식만으로도 공격이 성공할 수 있음을 보이기 위해 DNS 질의 및 응답 규칙을 기초로 하고, 기타 파라미터들은 임의로 생성하는 공격 모델을 생성해 이용한다.

클라이언트는 자신의 질의에 대한 응답인지 여부를 확인하기 위해 자신에게 전달된 응답 메시지의 트랜잭션 ID와 UDP 포트 번호를 확인하는데, 특정 윈도우즈 DNS 리졸버(eg. Windows XP SP1, Windows 2000 SP 3~4 등의 DNS resolver)의 경우에는 매우 단순한 형태의 트랜잭션 ID 및 UDP 포트 번호 변경 규칙을 사용하기 때문에 충분이 공격자가 예측 가능하다[11]. 이후 버전의 윈도우즈는 이와 같은 단순성을 회피하기 위해 트랜잭션 ID 및 UDP 포트 생성 규칙을 복잡화하였으나, 현재 이에 대한 분석 결과 및 공격 방식 또한 공개되어 있다[10]. 본 논문의 목적이 공격 알고리즘 생성이 아닌 모델을 통한 취약성 확인 가능성 파악이기 때문에, 본 논문에서는 메시지의 스니핑 없이, 단순한 형태의 트랜잭션 ID와 UDP 포트 번호 생성 규칙을 갖는 시스템을 대상으로 모델을 작성한다.

4. 시뮬레이션

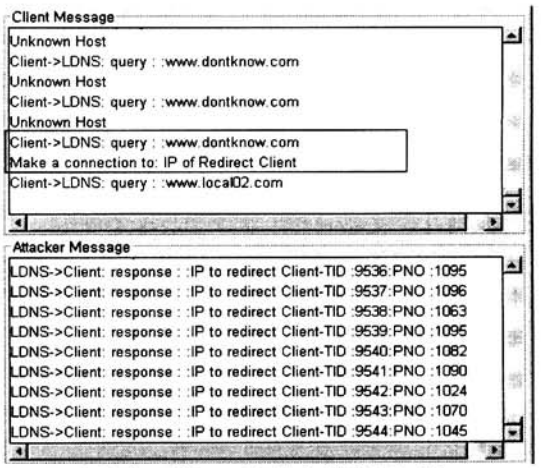
DNS 서버에 대한 중간자 공격 모델의 시뮬레이션을 위한 환경은 다음과 같은 형태로 설정되었다. 우선 클라이언트는 주어진 네 개의 호스트 네임에서 임의의 호스트 네임을 선택하여 질의를 수행하며, 이 때 클라이언트에 의해 선택되는 호스트는 내부 호스트와 외부 호스트 및 실제로 존재하지 않는 잘못된 이름을 갖는 호스트(unknown host)를 포함한다. 공격자는 클라이언트가 속한 도메인에 존재하는 DNS 서버의 메시지를 위조하여 클라이언트에 전달하는 기능을 수행하는데, 이 때 메시지 위조에는 <표 1>에 나타난 세 가지 {트랜잭션 ID, UPD 포트 번호} 생성 방법을 사용한다.

DNS 서비스에 이용되는 UDP 포트 번호의 경우, 표준 운영 환경(Standard operating enviro- nment)에서 대체로 클라이언트들이 동일한 포트를 사용하는 경향이 있음[7]을 이용하여 사전에 알고 있다고 가정하여 고정 포트 번호 사용하였으며, 트랜잭션 ID와 UDP 포트 번호를 모두 임의의 수를 사용하는 경우는 억지(brute force) 공격 기법에 가깝고, 성공 확률 또한 약 1/6백만으로 매우 낮아 실제 공격에 이용 가능성이 낮으므로 이는 시뮬레이션에서 배제하였다. 다음 (그림 3)은 위와 같은 환경을 가진 모델에서 시뮬레이

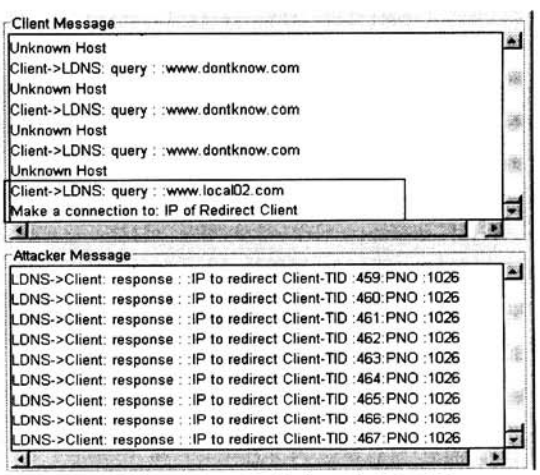
〈표 1〉 공격 메시지 생성 방법

1) {*trID* := *p_trID* +1, *pNo* := *r_PoNo*}
2) {*trID* := *p_trID* +1, *pNo* := *s_PoNo*}
3) {*trID* := *r_TrID*, *pNo* := *s_PoNo*}

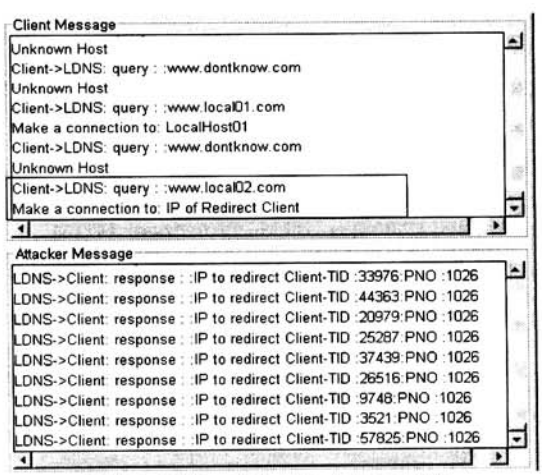
trID : 트랜잭션 ID
pNo : UDP 포트 번호
p_trID : 이전 트랜잭션 ID
r_TrID : 트랜잭션 ID 생성 범위 내 임의 숫자
s_PoNo : 고정 포트 번호
r_PoNo : DNS 서비스 이용에 사용되는 포트 번호 중 임의의 숫자



(a)공격 모델 1 : *trID* 증가, 임의 포트 번호



(b)공격 모델 2 : *trID* 증가, 고정 포트 번호



(c)공격 모델 3 : 임의 *trID*, 고정 포트 번호

(그림 3) 시뮬레이션 결과

션 했을 때 클라이언트의 콘솔에 나타나는 결과를 보인다. 클라이언트의 콘솔에는 클라이언트가 DNS에 요청한 메시지와 DNS로부터 받은 응답을 기초로 시도하는 연결 내용이 나타나는데, 이 과정에서 클라이언트가 공격자의 위조 메시지를 DNS 서버의 응답 메시지로 수용하는 경우 위조 호스트의 주소(IP of Redirect Client)로 연결을 시도하는 내용이 콘솔에 나타나도록 하였으며, 시뮬레이션 결과 수행된 모든 조건에서 클라이언트에 대한 공격이 성공하고 있음을 확인할 수 있었다.

5. 결 론

본 논문에서는 소프트웨어에 내재된 취약성 확인을 위한 방법으로 소프트웨어의 행동 방식(behavior)을 모델로 작성하고, 작성된 모델을 시뮬레이션 하는 방법을 사용하였다. 실험 결과에 따르면 소프트웨어의 고유 기능이 반영된 모델의 시뮬레이션을 통해서도 알려진 취약성 발견이 가능함을 알 수 있었다. 또한 본 논문에서 사용한 공격자의 공격 방식은 임의의 수를 생성하여 이용하고 있을 뿐, 메시지 스니핑 등 현재 알려진 적극적인 공격 방식은 사용하지 않고 있기 때문에, 모델이 포함하는 공격 방식을 정교화 하거나, 기존의 랜덤 테스팅에 사용된 알고리즘을 공격 패턴으로 변환하여 적용하는 경우, 더욱 효과적인 공격 형태 및 이에 대한 대응책을 찾을 수 있을 것으로 예상된다. 또한 모델에 방화벽 등 실제 네트워크에 존재하는 장비들을 추가하는 등 확장하고, 소프트웨어의 행동 방식을 더욱 정교하게 기술하는 경우, 기존에 알려지지 않은 취약성이 발견될 가능성도 있다.

향후 연구로는 소프트웨어 버전별 모델 작성 및 모델의 정교화를 통해 취약성이 발견되는 경우 심각한 피해를 유발하는 인프라 시스템에 사용되는 소프트웨어를 대상으로 모델링을 수행하여, 모델링 및 시뮬레이션을 통한 취약성 테스트 방법의 적용 대상 확장 및 개선 방향 등을 확인할 예정이다.

참 고 문 헌

[1] 박범주, 박기진, 김성수, "침입감내시스템의 생존성 모델," 한국정보처리학회논문지A, Vol.12-A, No.05, pp.395-404, 2005. 10.

[2] 신승훈, 박승규, 최경희, "효과적 취약성 검사를 위한 캐쉬 포이즈닝 모델링 및 시뮬레이션," 한국정보과학회논문지 제출.

[3] 주용완, 이응재, 남광우, "Recursive DNS의 캐쉬 정보 신뢰성 향상 기법," 한국정보처리학회논문지C, Vol.15, No.04, pp.227-238, 2008. 08.

[4] "Attacking the DNS Protocol - Security Paper v2," Security Associates Institute, Nov., 2003.

[5] C. Brooks, E.A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, H. Zheng (eds.), "Heterogeneous Concurrent Modeling and Design in Java," EECS Department, UC Berkeley, UCB/EECS-2008-28, 29, 37, Apr., 2008.

[6] D. Campara, "Software Modernization as a path to Secure Software," OMG Announces Program for Architecture-Driven Modernization Workshop, Oct., 2005.

[7] I. Green, "DNS Spoofing by The Man In The Middle," SANS Institute, Jan., 2005.

[8] M. J. Harrold. "Testing: a roadmap," in Proceedings of the Conference on the Future of Software Engineering, pp.61-72, 2000.

[9] E. Herrera, S. Read-Miller, "Vulnerability Management is Critical to Managing Enterprise Risk," Computer Associates International, Inc. May, 2005.

[10] A. Klein, "Microsoft Windows DNS Stub Resolver Cache Poisoning," Trusteer, Mar.-May, 2007.

[11] R. Larcher, "Predictability of Windows DNS resolver," http://webteca.altervista.org, Mar., 2004.

신 승 훈

e-mail : sihnsh@ajou.ac.kr

2000년 아주대학교 정보 및 컴퓨터공학부 (학사)

2002년 아주대학교 정보통신공학과(석사)

2002년~현 재 아주대학교 정보통신공학과 박사과정

관심분야 : 소프트웨어 테스트 자동화, 멀티미디어 서비스 정책 등

박 승 규

e-mail : sparky@ajou.ac.kr

1974년 서울대학교 용용수학과(공학사)

1976년 한국과학원(KAIST) 전산학과(석사)

1982년 Institut National Polytechnique de Grenoble 전산학과(박사)

1976년~1977년 한국과학기술연구소(KIST) 연구원

1977년~1978년 KIET (현ETRI) 연구원

1978년~1982년 프랑스 그레노블 IMAG 연구원/학생

1982년~1984년 KIET(현ETRI) 실장 / 선임연구원

1984년~1985년 미국 IBM 왓슨연구소 연구원

1985년~1992년 ETRI 연구위원 / 책임연구원

1992년~현 재 아주대학교 정보 및 컴퓨터공학부 교수

관심분야 : 임베디드 테스팅, 자가 컴퓨팅/치료 시스템, 차세대 컴퓨터 구조 등

정 기 현

e-mail : khchung@ajou.ac.kr

1984년 서강대학교 전자공학과(학사)

1988년 미국 Ilinois주립대 EECS(석사)

1990년 미국 Purdue대학 전기전자공학부 (박사)

1991년~1992년 현대반도체 연구소

1993년~현 재 아주대학교 전자공학부 교수

관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등