사이버 침입 탐지 시뮬레이션을 위한 SSFNet 기반 DNS 구현

*한종현° **이은영 **주미리 *박승규
*아주대학교 정보통신전문대학원 **국가보안기술연구소
hanbell@ajou.ac.kr°, eylee@etri.re.kr, mrjoo@etri.re.kr, sparky@ajou.ac.kr

Implementation of DNS for Network Intrusion simulations based on SSFNet

*Jong-hyun Han° **Eun-young Lee **Mi-Ri Joo *Seung-kyu Park
*Graduated School of Information and Communication, Ajou University
**National Security Research Institute

요 약

규모가 방대한 네트워크 상에서 네트워크의 침입과 방어의 효과와 유용성을 알아보기 위해, 실존하는 네트워크 상에서 직접 침입과 방어를 테스트하는 것은 많은 노력과 비용이 든다. 이와 같은 문제점을 극복하기 위해 인터넷 침입의 표현에 필요한 DNS Service를 포함한 네트워크 침입 시뮬레이션을 하기 위한 SSFNet 확장 연구가 진행되었다. 본 연구는 SSFNet에 새로이 추가된 DNS Service 모듈을 이전 연구에서 만들어진 모듈들과 함께 대규모 네트워크 환경에서 네트워크 침입 시뮬레이션을 테스트 하였다. 본 시뮬레이션에서는 1770개 노드로 구성된 네트워크에서 Http 서비스와 DNS 서비스를 제공하는 호스트들을 설정하고, 해당 서비스가 원활히 진행되는지를 살펴보았다

1. 서 론

네트워크상에서 네트워크 침입이 서버나 클라이언트에 미치는 영향을 실제 네트워크 상에서 연구하기에는 네트워크의 방대함과 또한 많은 비용 및 시간이 소모되기 때문에 어려움이 있다. 또, 네트워크 침입과 방어로 인한 현상들의 효용과 유용성에 대한 정확한 자료도 산출하기 힘들다. 시뮬레이션은 이러한 문제를 해결하는 훌륭한 대안이며, 다양한 시뮬레이션 도구가 개발되었다. 이러한 네트워크 시뮬레이션을 수행하기 위한 도구로는 NS-2[1]와 SSFNet[3]등이 있으며, NS-2는 멀티 캐스트를 지원하고, 지역적이기는 하나 무선환경에서의 시뮬레이션을 할 수 있지만 대규모 인터넷을 모델링하여 시뮬레이션 하기에는 적합하지 못하다.

SSFNet은 시뮬레이션 커널인 SSF의 소스는 공개되지 않았으나 그 중에서 네트워크의 시뮬레이션을 지원하는 SSFNet은 라우터, 링크, 네트워크 인터페이스 카드 등 대부분의 인터넷 서브 시스템들을 시뮬레이션 하는 데 필요한 다양한 객체들이 Java로 구현되어 있어 시뮬레이션 특성에 맞추어 그들의 특성을 변경할 수 있다는 장점을 가지고 있다. 또한 SSFNet은 이를 기반으로 상위단계의 대규모 네트워크까지도 표현하도록 허용하고 있으며, 네트워크상의 실존하는 특정 행동을 따라 구현이 가능하다.

본 논문에서는 향후 네트워크 침입과 방어에 대한 시뮬레이션에서는 실제 네트워크 상황과 비슷한 정보를 얻기 위해 대규모 네트워크 환경이 필요하다고 판단하여, SSFNet을 시뮬레이션 툴로 사용하였으나, SSFNet은 네트워크 침입과 방어에 관련된 시뮬레이션 중 Name Service Server에 질의를 보내어 Domain Name을 얻는 모듈이 구현되어있지 않다. 이에 본 논문에서는 slammer or sapphire worm의 DNS서버 침입을 구현하기 위하여 우선적으로 필요한 DNS Server 모듈을 구현하였다. 그리고 DNS 서버의 동작 시뮬레이션으로 slammer or sapphire worm 의 propagation을 테스트할 것이다. 이러한 연구의 결과 얻어진 시뮬레이션 기반은 네트워크 침입과 방어의 특성 시뮬레이션을 통한 다양한 침입과 방어 알고리즘개발, 네트워크 침입과 방어에 따른 네트워크 특성을 연구하기 위한 도구로 유용하게 사용될 것이다.

2. 관련 연구

NS[1] 는 discrete event simulator이고 Tcl과 C++로 작성되어있다. 그리고 유무선 네트워크에서 TCP, routing, multicast protocols을 제공한다. NS는 네트워크 애니메이터인 Nam과 함께 개발 되었고, Tcl 스크립트를 사용하여 네트워크 토폴로지(노드, 링크)과 트래픽 등을 표현한다. Tcl 오브젝트는 node, connection agents, traffic sources and sinks 등과 운영체제까지도 표현할 수 있다. 이것이 NS를 쉽게 사용할 수 있도록 해준다.

SSFNet[2]는 또 다른 시뮬레이션 프레임워크로서 SSF(Scalable Simulation Framework)를 기반으로 만들어졌다. 자바로 구현되어있고, 네트워크 시뮬레이션을 위해 만들어졌다. DML (Domain Modelling Language)은 SSFNet를 위해 네크워크를 표현하는데 사용되며, 어렵지 않고 쉽게 거대 네트워크를 만들어 낼 수 있다.

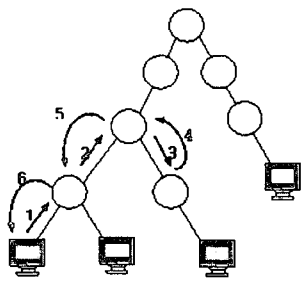
Domain Name System (or Service or Server)[4]는 도메인 네임을 IP주소로 변환시켜주는 인터넷 서비스이다. 도메인 네임은 문자로 표현되기 때문에 쉽게 기억할 수 있다. 그러나 인터넷에서는 IP주소 기반으로 만들어져

있다. 항상 도메인 네임을 사용하는 사용자들에게 해당 도메인의 IP주소 변환은 반드시 필요하다. DNS system은 자신의 네트워크에만 주소변환이 가능하기 때문에 주소변환 요청시 다른 DNS server에 변환을 요청할 수 있도록 트리구조를 갖고 있다.

3. DNS 설계와 구현

3.1. DNS Server Model

SSFNet에는 Domain Name의 개념이 없다. 따라서 DNS서비스를 구현하기 위해서는 이와 유사한 역할을 할 수 있도록 NHI주소(Network Host Interface address)를 사용한다. 즉 구현된 DNS 서버는 NHI주소를 IP 주소로 변환시키는 역할을 한다. 세부적인 동작을 보면, http 클라이언트는 http서버와 통신하기 위해 http서버의 IP주소를 얻으려고 한다. 클라이언트는 호스트의 디폴트 DNS 서버에게 DNS message 패킷전송을 시도한다. Message를 받은 DNS 서버는 message 의 NHI주소를 resolution을 위한 네트워크 주소 변환 테이블에 질의 내용과 부합되는 내용이 있는지 조사하고, 부합되는 정보가 있다면 클라이언트에게 응답을 보내준다. 만약 그렇지 않다면 DNS 서버트리에서 자식 또는 부모 DNS 서버에게 질의 내용을 포함하는 message를 보낸다. DNS서버 트리에서 부모 또는 자식 DNS의 선택은 시뮬레이션 시간에 영향을 미치지 않는 선택 알고리즘과 구조체를 사용한다. 이 경우 DNS 서버로부터 message를 받은 DNS서버는 재귀적인 방법으로 message를 처리한다. 마지막으로 SSFNet 에서 호스트에 많은 패킷이 몰려 정체되더라도 시뮬레이션은 진행되기 때문에 이 문제를 해결하기 위하여 DNS서버의 요청 처리 능력을 제한시킨다.



<그림 1> DNS 메시지 교환

<그림 1>은 클라이언트와 DNS 서버, DNS 서버와 DNS 서버 사이의 질의와 응답 message가 교환 되는 모습을 보여주고 있다. http 클라이언트는 자신의 디폴트 DNS 서버에게 http서버의 Domain Name 질의를 담은 message를 보내고 처음 message를 받은 DNS 서버는 자신의 주소변환 테이블에 부합되는 내용이 없으므로 부모 DNS 서버에게 질의 message를 보낸다. 두 번째 DNS 서버에서도 자신의 주소변환 테이블을 확인하고, 부합되는 내용이 없으므로 부모 또는 자식 DNS서버에 질의 message를 보내야 하는데, 자식 DNS 서버의 주소변환 테이블에 질의와 부합되는 내용이 있을 것을 알고 자식 DNS 서버로 질의를 보낸다. 그리고 서버의 IP주소를 얻어 질의가 경유했던 경로를 거꾸로 거슬러 클라이언트에게 전달된다.

3.2. Behavior of each Module

1) DNSmessage

DNS message는 클라이언트와 DNS 서버, DNS서버와 DNS 서버사이의 전송해야할 데이터를 정의하고 있다. Message에는 Question과 Answer 두 가지 타입이 존재한다

2) HttpClient

TCP 연결을 성립시키고, resolution하고자 하는 HttpServer의 NHI주소를 DNS 질의 message에 담아 DNS 서버에게 요청하고, DNS 서버로부터 오는 응답 message의 값으로 HttpServer와 통신을 한다. 인터넷 통신을 사용하기 위해 모든 클라이언트는 Domain Name으로 서비스를 요청한다고 가정하고, 클라이언트가 유지하고 있는 서버 리스트의 랜덤하게 접근하는 서버에 대하여 모두 적용된다.

3) DNSserver

자신에게 들어오는 패킷이 질의 message 인지를 확인한다. 정상적인 질의 message라면, Domain Name을 resolution을 시작한다.

```
if(DNS질의패킷이 들어오면)
{
  if(DNS테이블 탐색)
  { // 테이블에 변환정보가 있다면
    out = new DNSmessage(ANSWER, IP_ans);
    write(out);
    return success;
  }
  else
  { // 테이블에 정보가 없다면 서브트리에 정보가 있는지 체크
    if(서브트리 탐색)
    { // 서브트리에 변환정보가 있다면
      dns_query(childnhi);
      out = new DNSmessage(ANSWER,IP_ans);
      write(out);
      return success;
    }
    else
    { // if not root, query to parent DNS
      if(부모노드가 루트인지 체크) return failure;
      else
      {
        dns_query(parent);
        out = new DNSmessage(ANSWER,IP_ans);
        write(out);
        return success;
      }
    }
  }
}
else return failure
```

<그림 2> DNS 알고리즘

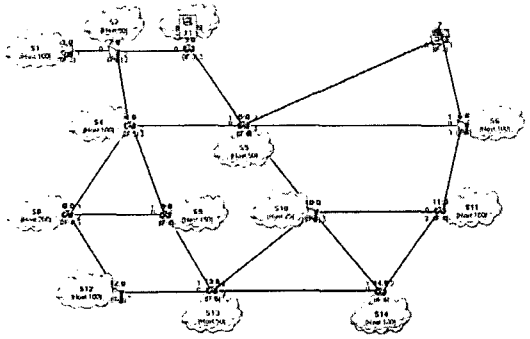
<그림 2>은 DNS서버가 resolution하는 알고리즘을 보여준다. 우선 DNS 서버는 자신의 주소변환 테이블을 탐색한다. 주소변환 테이블에서 얻고자 하는 서버의 NHI주소가 검색되면, DNS는 NHI주소에 해당하는 IP 주소를 질의 message를 보낸 클라이언트에게 응답 message를 보내고 서비스를 마친다. 만약 주소변환 테이블에서 검색되지 않았다면, DNS 서버 트리에서 자식 노드를 루트로 하는 서브트리에서 NHI주소를 IP주소로 resolution 할 수 있는지 가능여부를 확인한다. 서브트리에서 NHI주소를 IP주소로 resolution 할 수 있는지 확인이 되면 해당 서브트리로 질의 message를 보낸다. 그러나, 서브트리에서 resolution 가능성이 없다면, 부모노드의 DNS에 질

의 message를 보내게 된다.

DNS Server는 많은 패킷이 몰려 처리한도를 넘어서게 되면 그 기능을 수행하지 못하고 다운되게 된다. 따라서 DNS서버의 요청 처리 능력 초당 100000개의 패킷을 처리하도록 제한시킨다. 이것은 가변적인 값으로 조정이 가능하다

4. 시뮬레이션

시뮬레이션 하는 네트워크는 14개의 서브넷, 서브넷들을 연결해주는 13개의 라우터, 라우터들 중 3개는 방화벽이 설치 그리고 또 다른 라우터위에 IDS 한 개가 올라가 있는 약 1770개의 호스트로 구성되어있다.



<그림 3> 시뮬레이션 네트워크

그리고 구현된 DNS서버는 1770개의 호스트 중에서 2개의 호스트에 DNS서버를 설정하였다. 각각의 DNS의 설정은 테이블 1과 같다.

Nhi address of DNS server	parent DNS	Address resolution	Client
5:1	5:1	*	10:*
13:1	5:1	1:1	*

[표 1] DNS 서버 설정

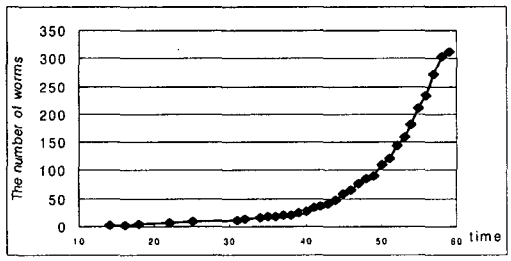
호스트 5:1의 DNS서버는 모든 호스트의 주소를 resolution할 수 있고 서브넷 10:*의 호스트들이 호스트 5:1의 DNS를 디폴트 DNS로 하고 있다. 또, 호스트 13:1의 DNS서버는 호스트 1:1의 주소를 resolution할 수 있고, 서브넷 10:*의 호스트를 제외한 나머지 호스트들이 호스트13:1의 DNS를 디폴트 DNS로 지정하고 있다. 그리고 parent DNS는 트리를 구성하기 위한 정보이다.

```
1: [search_DNS_table] name : 9:45
2: (at 13:1 DNS) 9:* 9:45(0)
3: (at 13:1 DNS) * 9:45(0)
4: [search_child_dns] dnsnhi : 13:1(0)
5: [search_childDNS_table] name : 9:45
6: [search_child_table] can't resolution at 13:1
7: [search_child_table] search at other child DNS
8: query to parent
9: [dns_query] secDNSnhi : 5:1(0)
10: parent DNS connect() OK
11:
12: [search_DNS_table] name : 9:45
13: (at 5:1DNS) 9:* 9:45(0)
14: (at 5:1DNS) * 9:45(0)
15: ipaddress 2094    Domain_name 9:45(0)
16:
17: httpClient connection OK  :2094
```

<그림 5> DNS 시뮬레이션 결과

그림 5는 시뮬레이션의 결과로 나온 로그를 보여준다. 라인 1에서 NHI주소 9:45을 IP로 변화하려는 요청이 있어 테이블을 검색하고 있다. 그러나 테이블에 직접적으로 매칭되는 내용이 없어 라인 2과3에서 *로 표현된 정보를 해석하고 있다.

부모 DNS인 5:1에서 요청한 9:45주소를 변환하였고 라인 15에서 결과message를 보여주고 있다. 라인 17에서 응답을 받은 클라이언트는 message의 값을 가지고 서버와 통신을 한다.



<그림 6> 사파이어웜 시뮬레이션

그림 6은 사파이어 웜의 수를 보여준다. X축은 시뮬레이션이 진행되어진 시간이고, Y축은 시뮬레이션이 진행됨에 따라 퍼진 웜의 수이다. DNS의 동작을 보기위하 사파이어 웜과 함께 시뮬레이션을 하였다. 그리하여 사파이어웜이 IP변환 요청을 만들어내고 DNS는 사파이어웜이 감염된 호스트에게 응답을 주어 사파이어웜은 퍼질 수 있다. 그림 6에서 웜의 확산이 보여주듯이 DNS가 잘 동작함을 알 수 있다.

5. 결론 및 향후 과제

본 논문에서는 DNS Server에 필요한 모듈의 구현과 향후 연구할 네트워크 침입과 방어에 관련된 설정과 그에 따른 반응 구현을 위해 SSFNet을 확장하여 대규모 인터넷에서 DNS의 시뮬레이션이 가능한 기반을 구축하고, 그의 정상적 구동을 확인하였다. 앞으로 본 연구 결과와 네트워크 침입과 방어 시뮬레이션에 더 필요한 모듈들을 응용하여, 대규모 네트워크 환경에서 네트워크 침입과 방어에 관련된 시뮬레이션을 수행 할 것이다.

6. 참고문헌

[1] "NS-2 The Network Simulator", http://www.isi.edu/nsnam/ns/

[2] http://snl.cs.hongik.ac.kr/~kimyj/ns/nsman1/

[3] "SSF Simulator implementation', http://www.ssfnet.org/ssfImplementations.html

[4] Stevens, "TCP/IP Illustrated, Vol 1", Addison Wesley, 1994

[5] Jae-hyuk Lee, Kyu-chan Cho, Seung-kyu Park, Kyung-hee Choi "SSFNet Extension for Web Proxy Server", Korea Information Science Society , vol. 29, issue 1, 2002

[6] Jae-Hyuk Lee, Chul-Won Lee, Eul-Gyu Im, Seung-Kyu Park, Kyung-hee "SSFNet Extension for Telnet Service", Korean Institute of Communication and Sciences , vol. 26, 2002

[7] Paul V. Mockapetris, Kevin J. Dunlap "Development of the domain name system", ACM SIGCOMM Computer Communication Review, Volume 25 Issue 1,1995