1. Boink 공격

패킷 조작: Boink 공격에서는 패킷의 시퀀스 번호를 임의로 변경하여 전송한다. 예를 들어, 여러 개의 패킷이 순서대로 전송되어야 할 때, 공격자는 패킷의 순서를 무작위로 변경하여 수신자가 패킷을 올바르게 재조립하지 못하도록 한다.

실제 예시:
공격자는 hping3와 같은 도구를 사용하여 패킷을 생성하고, 특정 IP 주소로 패킷을 전송할 때, 패킷의 ID와 시퀀스 번호를 조작하여 비정상적인 순서로 전송한다.

2. Teardrop 공격의 작동 원리

패킷 조각화: TCP/IP 프로토콜에서 데이터가 너무 클 경우, 패킷은 여러 조각으로 나뉘어 전송된다. Teardrop 공격자는 이 조각화된 패킷을 조작하여, 수신 시스템이 이를 올바르게 재조립하지 못하도록 한다.

순서와 길이 조작: **공격자는 패킷의 조각 순서와 길이를 조작하여**, 수신 시스템이 패킷을 재조립할 때 혼란을 일으킨다. 예를 들어, 조각의 순서를 변경하거나 조각의 길이를 비정상적으로 설정하여, 시스템이 패킷을 올바르게 처리하지 못하도록 만든다.

시스템 충돌 유도: 이러한 조작으로 인해 수신 시스템이 패킷을 재조립하는 과정에서 오류를 발생시키고, 이로 인해 시스템이 충돌하거나 다운된다.

3. 패킷 크기와 죽음의 핑 공격

① ICMP 프로토콜: 죽음의 핑 공격은 ICMP(Internet Control Message Protocol)를 사용하여 이루어진다. ICMP는 네트워크 장치 간의 오류 메시지 및 진단 정보를 전송하는 데 사용되는 프로토콜이다. 일반적으로 ICMP 에코 요청(핑 요청) 패킷은 상대적으로 작은 크기를 가진다.
② 비정상적으로 큰 패킷: **공격자는 ICMP 에코 요청 패킷의 크기를 비정상적으로 크게 설정하여 전송한다. ICMP 패킷의 최대 크기는 이론적으로 65,535바이트이다.** 그러나 일반적인 핑 요청은 보통 32바이트 또는 64바이트 정도입니다. 공격자는 이 크기를 극대화하여 대량의 데이터를 포함한 패킷을 생성한다.
③ 대량 전송: 공격자는 이러한 큰 ICMP 패킷을 대량으로 대상 시스템에 전송한다. 예를 들어, 초당 수천 개의 큰 ICMP 에코 요청 패킷을 보내면, 대상 시스템은 이 패킷들을 처리하기 위해 많은 자원을 소모하게 된다.
④ 시스템 자원 소모: 패킷의 크기가 크면 클수록, 대상 시스템의 CPU와 메모리 자원 소모가 증가한다. 큰 패킷을 수신하고 처리하는 데 필요한 시간과 자원이 더 많이 소모되기 때문에, 시스템의 성능이 저하되고, 결국 정상적인 서비스 요청을 처리할 수

없게 된다.

⑤ 서비스 거부 상태: 이로 인해 대상 시스템은 과부하 상태에 빠지게 되고, 서비스 거부(DoS) 상태에 이르게 됩니다. 사용자는 정상적인 네트워크 서비스에 접근할 수 없게 된다.

방어 방법

패킷 크기 제한: **방화벽이나 네트워크 장비에서 ICMP 패킷의 최대 크기를 제한하여 비정상적으로 큰 패킷을 차단한다.**

속도 제한: ICMP 요청의 수를 제한하여 특정 시간 내에 수신할 수 있는 요청의 수를 조절합니다. 이를 통해 대량의 요청이 들어오는 것을 방지할 수 있다.

네트워크 모니터링: 패킷 크기와 전송 빈도를 모니터링하여 비정상적인 트래픽 패턴을 탐지하고, 공격을 조기에 차단할 수 있다.

4. HTTP/1.1에서 CC 헤더

"Cache-Control" 헤더를 의미한다. 이 헤더는 웹 캐시의 동작을 제어하는 데 사용된다. 주어진 문장의 의미는 다음과 같다:

① 자주 변경되는 데이터: 웹에서 자주 업데이트되는 콘텐츠(예: 뉴스, 실시간 데이터 등)는 캐시를 통해 저장되기보다는 항상 최신 상태로 유지되어야 한다.

② 새로운 HTTP 요청 및 응답 요구: 자주 변경되는 데이터에 대해 사용자가 항상 최신 정보를 받도록, 캐시된 데이터를 사용하지 않고 매번 서버에 새로운 요청을 보내도록 요구할 수 있다.

③ 캐시 기능을 사용하지 않을 수 있음: 이 경우, Cache-Control 헤더를 설정하여 브라우저나 중간 캐시 서버가 해당 데이터를 캐시하지 않도록 지시할 수 있다. 예를 들어, Cache-Control: no-cache 또는 Cache-Control: no-store와 같은 지시어를 사용하여 캐시를 비활성화할 수 있다.

- no-store: 이 지시자를 포함한 요청은 응답을 클라이언트 캐시에 저장하지 못하게 한다. 민감한 정보를 다루는 경우 사용되기도 하지만, 공격자는 이를 악용하여 매 요청마다 서버로부터 새로운 응답을 받도록 강제한다.
- must-revalidate: 이 지시자는 클라이언트가 캐시된 응답을 사용하기 전에 반드시 서버에 재검증을 요청하도록 한다. 이를 통해 클라이언트는 항상 최신 상태의 리소스를 사용하게 되지만, 공격자는 이를 통해 모든 요청이 서버에 도달하게 만들어 서버 자원을 소모시킨다.

공격자는 이러한 Cache-Control 지시자를 조작한 대량의 HTTP 요청을 서버로 전송한다. 이로 인해 클라이언트나 중간 캐싱 서버가 캐시를 활용하지 못하고, 모든 요청이 웹 서버로 직접 전달된다. 결과적으로 웹 서버는 평소보다 훨씬 많은 리소스를 사용하여 요청을 처리해야 하며, 이는 서버 과부하로 이어져 정상적인 서비스 제공을 어렵게 만든다.

5. 슬로 HTTP 헤더 DoS 공격

- 공격자가 HTTP 요청의 헤더를 매우 느리게 전송한다는 의미는, HTTP 요청을 구성하는 헤더 필드들을 정상적인 속도보다 훨씬 느리게 서버에 전송하여, 서버가 요청을 완전히 수신하는 데 오랜 시간이 걸리도록 만드는 것.

- **구체적으로 설명하자면:헤더 구성: HTTP 요청은 여러 개의 헤더 필드로 구성. 예를 들어, GET, Host, User-Agent, Accept 등의 필드가 있음. 공격자는 이러한 헤더 필드를 한 번에 전송하는 것이 아니라, 각 필드를 매우 느리게, 또는 일부 필드만 전송한 후 일정 시간 동안 기다리는 방식으로 전송. 예를 들어, 한 필드를 전송한 후 몇 초 또는 몇 분의 지연을 두고 다음 필드를 전송.**

- 서버의 대기 상태: 서버는 요청이 완료될 때까지 해당 연결을 열어두고 대기. 이로 인해 서버는 리소스를 소모하게 되고, 동시에 여러 개의 느린 요청을 처리하게 되면 서버의 연결 수가 고갈.

- 정상적인 요청과의 구별 어려움: 이러한 방식은 정상적인 사용자 요청과 구별하기 어려워, 서버는 이를 정상적인 트래픽으로 인식하고 계속해서 리소스를 할당. 결과적으로, 공격자는 서버의 자원을 소모시키고, 정상적인 사용자들이 서버에 접근할 수 없도록 만드는 효과

6. 동적 HTTP 리퀘스트 플러딩 공격

- 동일한 URL 반복 요청: 공격자는 특정 웹 페이지에 대해 반복적으로 HTTP GET 요청을 보낸다. 예를 들어, GET /index.jsp HTTP/1.1와 같은 요청을 대량으로 전송하여 서버의 자원을 소모시킨다. 이 경우, 공격자는 동일한 URL을 사용하지만, 요청의 파라미터나 쿼리 문자열을 변경하여 요청을 다르게 보이게 할 수 있다. 예를 들어:
  GET /index.jsp?user=1
  GET /index.jsp?user=2
  GET /index.jsp?session=abc123
  이러한 방식으로 요청을 변경하면 웹 방화벽이 공격을 탐지하기 어려워진다.

- HTTP 헤더 조작: 공격자는 HTTP 요청의 헤더를 조작하여 요청을 다르게 보이게 할 수 있다. 예를 들어, User-Agent 헤더를 변경하여 다양한 브라우저에서 요청하는 것처럼 보이게 할 수 있다. 이는 서버가 요청을 정상적인 사용자로 인식하게 만들고, 방어 시스템을 우회하는 데 도움을 준다.

7. 사용자가 VPN에 연결하면, 모든 인터넷 트래픽은 암호화되어 VPN 서버를 통해 전달된다. **이 과정에서 사용자의 실제 IP 주소는 VPN 서버의 IP 주소로 대체된다. 즉, 외부 웹**

사이트나 서비스는 사용자의 실제 IP 주소가 아닌 VPN 서버의 IP 주소를 보게 된다. VPN 서비스는 여러 지역에 서버를 운영한다. 사용자는 원하는 서버에 연결함으로써, 자신의 위치를 숨기고 다른 지역에서 접속하는 것처럼 보이게 할 수 있다. 예를 들어, **한국에 있는 사용자가 미국에 있는 VPN 서버에 연결하면, 웹사이트는 사용자가 미국에서 접속하는 것처럼 인식한다.**

8. 스위치 재밍(Switch Jamming) 공격

- 전용 도구 사용: 공격자는 특정 네트워크 공격 도구를 사용할 수 있다. 예를 들어, "Yersinia", "Macof"와 같은 도구는 MAC 주소 오버플로우 공격을 수행하는 데 특화되어 있다. 이러한 도구는 사용자가 설정한 수의 **가짜 MAC 주소를 생성하고**, 이를 스위치에 전송하여 MAC 주소 테이블을 가득 채우는 기능을 제공한다.

- 패킷 생성 및 전송: 공격자는 패킷 생성 도구(예: Scapy)를 사용하여 직접 패킷을 생성하고, 이를 네트워크에 전송할 수 있다. 이 경우, 공격자는 다양한 MAC 주소를 설정하여 스위치에 전송함으로써 MAC 주소 테이블을 오버플로우 시킨다.

9. 고가의 스위치는 MAC 테이블의 캐시와 연산자가 쓰는 캐시가 독립적으로 나뉘어 있어 통하지 않는 다는 것의 의미는?

- MAC 테이블 캐시: 스위치의 MAC 주소 테이블은 네트워크에서 각 포트에 연결된 장치의 MAC 주소를 저장하는 데이터 구조이다. 이 테이블은 스위치가 패킷을 올바른 포트로 전달하기 위해 사용된다. **MAC 주소 테이블의 캐시는 스위치가 학습한 MAC 주소와 해당 주소가 연결된 포트를 저장하며, 이 정보는 스위치의 데이터 전송 경로 결정에 사용된다.**

- 운영자 캐시: 운**영자가 사용하는 캐시는 스위치의 관리 및 설정과 관련된 정보를 저장하는 캐시이다.** 예를 들어, 운영자는 스위치의 설정, 상태, 성능 모니터링 데이터 등을 관리하기 위해 별도의 캐시를 사용할 수 있다. 이 캐시는 스위치의 운영 및 관리와 관련된 정보를 효율적으로 처리하기 위해 설계되었다.

- 독립성: 두 캐시가 독립적으로 나뉘어 있다는 것은, 하나의 캐시에서 발생하는 변화나 업데이트가 다른 캐시에 영향을 미치지 않는다는 것을 의미한다. 예를 들어, MAC 주소 테이블이 업데이트되더라도 운영자가 사용하는 캐시는 그와는 무관하게 작동하며, 반대로 운영자가 캐시를 업데이트하더라도 MAC 주소 테이블에는 영향을 주지 않는다. 이러한 독립성은 스위치의 성능과 안정성을 높이는 데 기여한다.

10. SPAN 포트 태핑(Port Tapping) 공격

- 네트워크 스위치에서 발생할 수 있는 보안 위협 중 하나로, 공격자가 스위치의 SPAN(Switched Port Analyzer) 기능을 악용하여 네트워크 트래픽을 가로채는 방식이다. SPAN은 스위치에서 특정 포트의 트래픽을 복제하여 다른 포트로 전송하는 기능으로, 주로 네트워크 모니터링 및 분석을 위해 사용된다.

- SPAN 포트 태핑 공격의 작동 방식
  ① SPAN 설정 변경: **공격자는 스위치의 관리 권한을 얻거나, 물리적으로 접근하여 SPAN 포트 설정을 변경할 수 있다. 이를 통해 공격자는 특정 포트의 트래픽을 복제하여 자신의 장치로 전송하도록 설정할 수 있다.**
  ② 트래픽 가로채기: SPAN 포트가 설정되면, 공격자는 해당 포트에서 전송되는 모든 트래픽을 모니터링할 수 있다. 이로 인해 공격자는 민감한 데이터, 인증 정보, 패스워드 등을 포함한 네트워크 트래픽을 가로챌 수 있다.
  ③ 정보 유출: 공격자가 가로챈 트래픽은 다양한 방식으로 악용될 수 있으며, 이는 데이터 유출, 네트워크 침해, 또는 추가적인 공격(예: 중간자 공격)으로 이어질 수 있다.

11. DNS를 이용한 스니퍼 탐지의 원리

- 스니퍼가 네트워크 상의 다양한 IP 주소를 가진 패킷을 수집할 때, 해당 IP 주소에 대한 추가 정보를 얻기 위해 역방향 DNS 조회(Inverse DNS lookup 또는 Reverse DNS lookup)를 시도할 수 있다. 정상적인 호스트는 자신과 직접 통신하는 IP 주소에 대해서만 DNS 조회를 하는 것이 일반적이다. 따라서 불필요한 IP 주소에 대해 역방향 DNS 조회를 수행하는 호스트는 스니퍼일 가능성이 있다.
- 탐지 과정: 테스트 대상 네트워크로 Ping Sweep을 보낸다. 이는 네트워크 상의 활성 호스트를 파악하기 위함이다. 이후, 네트워크에서 발생하는 역방향 DNS 조회를 감시한다. 네트워크 상의 특정 호스트에서 정상적인 통신 범위를 넘어서는 대량의 역방향 DNS 조회가 발생하는 경우, 해당 호스트에서 스니핑이 이루어지고 있음을 의심할 수 있다.

12. 한 개의 망 안에 두 개의 라우터가 존재

- 한 개의 망안에 두 개의 라우터가 존재할 수 있다. 실제로 많은 네트워크 환경에서는 여러 개의 라우터를 사용하여 트래픽을 관리하고, 네트워크의 범위를 확장하며, 장애 조치를 제공하는 등의 기능을 수행한다.

- 두 개의 라우터가 같은 네트워크에 존재할 경우, 일반적으로 다음과 같은 방식으로 구성된다.
  ① 서브넷 분할: 각 라우터가 서로 다른 서브넷을 관리하여 네트워크를 분할할 수 있다.
  ② 로드 밸런싱: 두 라우터가 동시에 트래픽을 처리하여 부하를 분산시킬 수 있다.

③ 백업 라우터: 하나의 라우터가 실패할 경우 다른 라우터가 자동으로 트래픽을 처리하도록 설정할 수 있다.