**커맨드 인젝션 공격**

**1. 커맨드 인젝션 공격 개요**

- 웹 요청 메시지에 임의의 시스템 명령어를 삽입하고 전송하여 웹 서버에서 해당 명령어를 실행하는 공격

- 사용자가 입력한 값을 입력 값 검증 없이 시스템 명령어의 일부분으로 전달하면 공격자는 이 값을 조작하여 임의의 시스템 명령어를 실행 가능

텍스트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

- 사용자가 IP 주소를 입력하면 웹 서버에서 ping 명령어를 실행하고 그 결과를 알려주는 웹페이지 예시

- ping 명령은 입력된 IP 주소의 시스템이 현재 동작하고 있는지 확인 가능

텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

- 공격자는 IP 주소 뒤에 ;(세미콜론)을 입력하고 다른 시스템 명령어를 추가

- ;(세미콜론)은 복수의 명령어 실행 가능

- cat /etc/passwd 명령어는 리눅스의 사용자 목록 확인 가능

- 공격자는 ping 명령어와 cat 명령어 동시 실행 가능

**2. 커맨드 인젝션 공격 실습**

텍스트, 전자제품, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<보안 레벨 – Low / Command Injection>

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

- 루프백 주소인 127.0.0.1을 입력했을 때, 위와 같은 결과 출력

|  |
| --- |
| <?php  if( isset( $\_POST[ 'Submit' ]  ) ) {      // Check Anti-CSRF token      checkToken( $\_REQUEST[ 'user\_token' ], $\_SESSION[ 'session\_token' ], 'index.php' );       // Get input      $target = $\_REQUEST[ 'ip' ];      $target = stripslashes( $target );       // Split the IP into 4 octects      $octet = explode( ".", $target );       // Check IF each octet is an integer      if( ( is\_numeric( $octet[0] ) ) && ( is\_numeric( $octet[1] ) ) && ( is\_numeric( $octet[2] ) ) && ( is\_numeric( $octet[3] ) ) && ( sizeof( $octet ) == 4 ) ) {          // If all 4 octets are int's put the IP back together.          $target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' . $octet[3];           // Determine OS and execute the ping command.          if( stristr( php\_uname( 's' ), 'Windows NT' ) ) {              // Windows              $cmd = shell\_exec( 'ping  ' . $target );          }          else {              // \*nix              $cmd = shell\_exec( 'ping  -c 4 ' . $target );          }           // Feedback for the end user          echo "<pre>{$cmd}</pre>";      }      else {          // Ops. Let the user name theres a mistake          echo '<pre>ERROR: You have entered an invalid IP.</pre>';      }  }  // Generate Anti-CSRF token  generateSessionToken();   ?> |

<Command Injection Source Code>

- 리눅스는 shell\_exec() 함수 호출 -> 시스템 명령어를 실행하는 함수

- “ping -c 4”을 실행하는 것을 통해 ping을 4번만 발생시킨다는 것을 알 수 있음

- “$target”은 $\_REQUEST[‘ip’]의 결과 값으로 웹 요청 메시지로 받은 파라미터 값이라는 것을 알 수 있음

- 특별한 입력 값 검증이 없어 ;(세미콜론)을 통해 다른 명령어도 실행할 수 있음

- ;(세미콜론) 앞에 명령이 실패해도 뒤에 명령은 실행될 수 있음

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

<;ls를 실행한 결과>

- 디렉토리의 내용 출력됨

- ping 명령어는 에러로 처리되어 출력되지 않음

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

<;cat /etc/passwd를 실행한 결과>

- 사용자의 정보가 담긴 /etc/passwd 파일이 읽힌 것을 볼 수 있음

- 윈도우의 경우 &&를 사용

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

<;id를 실행한 결과>

- id 명령어를 이용하여 명령어를 실행하는 사용자를 알 수 있음

- 현재 사용자는 “www-data”, 어떤 시스템은 root 사용자를 사용하는데, 이는 심각한 위험 초래 가능

**3. 커맨드 인젝션 공격 대응**

- 가장 좋은 방법은 직접적으로 시스템 명령어를 호출하지 않는 것

- exec() 또는 system()과 같이 직접적으로 명령어를 실행하는 함수 사용 X

- 프로그래밍 언어 및 라이브러리에서 자체적으로 제공하는 함수 사용 권장

ex) system(“mkdir $dir\_name”) X, mkdir($dir\_name) O