Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.4, Aug. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.4.661

웹쉘 기술을 통한 프록시 기반의 확장 가능한 서버 관리 프레임워크*

김 다 은,[1†] 배 상 욱,[2] 김 성 민,[3] 정 은 영[4‡]
[1,3]성신여자대학교 (학생, 교수), [2,4]알파카네트웍스 (연구원, 대표이사)

Proxy-Based Scalable Server Access Management Framework Using Reverse Webshell Protocol*

Daeun Kim,[1†] Sangwook Bae,[2] Seongmin Kim,[3] Eunyoung Jeong[4‡]
[1,3]Sungshin Women's University (Undergraduate student, Professor),
[2,4]Alpaca Networks (Researcher, CEO)

요 약

서버리스 컴퓨팅 패러다임의 등장과 함께 클라우드 기술이 발전함에 따라, 서버 관리를 위한 백엔드 인프라의 구조가 온프레미스부터 최신 컨테이너 기반 서버리스 컴퓨팅까지 다변화되고 있다. 그럼에도 불구하고, 서버 관리를 위한 접속 방식은 전통적인 SSH 프로토콜에 여전히 의존하고 있으며, 보안성 및 확장성 측면에서의 한계점으로 인해, 사용자의 서버 인프라 관리 편의성 및 업무 생산성을 저하시킨다. 이러한 문제를 해결하기 위해 본 논문에서는 웹쉘을 프록시 기반의 서버 관리 프레임워크에 적용하여 실용성과 보안성을 갖춘 서버 관리체계 설계에 활용하고자 한다. 흔히 해커들이 서버의 취약점을 이용하여 임의 명령을 실행하기 위해 웹쉘을 사용하지만, 본 논문에서는 서버 관리 측면에서의 웹쉘 기술의 활용성에 대해 고찰하고, 웹쉘 기술을 사용한 접속 프록시 기반 서버 관리 프레임워크를 새롭게 제안한다. 또한, 본 논문에서는 실제 구현을 통하여 제안한 프레임워크가 추가적인 오버헤드 없이 표준적인 서버 접속 프로토콜로 사용되어온 SSH의 단점을 보완하고, 현대의 다양화된 컴퓨팅 환경에서도 대규모의 인프라를 효율적으로 운영할 수 있음을 보인다.

ABSTRACT

With the emergence of serverless computing paradigm and the innovations of cloud technology, the structure of backend server infrastructure has evolved from on-premises to container-based serverless computing. However, an access control on the server still heavily relies on the traditional SSH protocol, which poses limitations in terms of security and scalability. This hampers user convenience and productivity in managing server infrastructure. A web shell is an interface that allows easy access to servers and execution of commands from any device with a web browser. While hackers often use it to exploit vulnerabilities in servers, we pay attention to the high portability of web shell technology for server management. This study proposes a novel proxy-based server management framework utilizing web shell technology. Our evaluation demonstrates that the proposed framework addresses the drawbacks of SSH without additional overhead, and efficiently operates large-scale infrastructures in diverse computing environments.

Keywords: SSH, Webshell, Server management, Access proxy

Received(07. 05. 2023), Modified(07. 31. 2023), Accepted(08. 01. 2023)

* 본 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(NRF-2021R1G1A100632611), 과학기술정보통신부 및 정보통신기획평가원의 ICT혁신인재4.0 사업(IITP-2022-RS-2022-00156310) 및 정보통신산업진흥원 공개 SW 기술확산 지원사업(S0301-23-1005)의 지원을 받아 수행된 연구임.

* 본 논문은 알파카네트웍스의 인턴십을 통해 수행된 연구결과임.

† 주저자, 20190886@sungshin.ac.kr

‡ 교신저자, ey.jeong@alpacanetworks.com(Corresponding author)

I. 서 론

SSH(Secure Shell)는 현재까지 약 30년의 시간 동안 가장 표준적인 서버 접속 프로토콜로 사용되어 왔다. 암호화된 통신 채널을 통해 종단 간 기밀성과 진실성을 보장하며, OpenSSH, PuTTY와 같은 클라이언트가 활용된다. 그러나, SSH는 비밀번호 또는 공개키 방식의 로그인을 사용함으로써 브루트 포스와 중간자 공격에 취약점을 노출한다. 이러한 위협을 완화하고자 그동안 많은 연구에서 루트 로그인 차단, 원격지 접속 차단, 포트 변경 등의 추가적인 보안 조치[1,8,9]들이 소개되었지만, 이는 사용자의 편의성과 업무 생산성을 저하시킨다.

SSH에서 사용자 계정 및 비밀번호 검증은 PAM(Pluggable Authentication Module)이 담당한다. PAM은 통합 UNIX 인증 프레임워크로 시스템의 응용 프로그램에 대한 사용자의 인증 및 권한을 제어하는 모듈이다. PAM에서 비밀번호를 대체하여 FIDO(Fast Identity Online)[2]와 같은 인증 방법을 추가할 수 있지만, 각각의 모듈에 맞는 SPI(Service Programming Interface)를 구현하여 PAM에 등록하여야 하는 번거로움이 있다. 즉, PAM을 사용한다면 편의성 및 안전성이 강화된 기술이 개발되더라도 이를 즉시 적용하기 어려운 문제가 있다[3].

현재의 컴퓨팅 환경은 과거와 달리 클라우드 컴퓨팅, IoT, 원격 근무 등으로 인해 다양화되고 인프라의 규모가 커졌으며, 이에 따라 증대되는 보안 위협으로 인해 서버 관리의 애로사항이 커지고 있다. SSH나 PAM이 설계된 당시에는 규모나 보안 위협 측면에서 당시의 설계가 적합하였으나, 현재의 환경에서는 이를 사용한 서버 접속 인증과 관리 방법이 효율적이지 않을 수 있다. 기본적으로 SSH는 사용자가 하나의 서버를 접속 및 관리하는 과정에서 사용되는 프로토콜로, 대규모의 서버들을 관리하기 위해서는 서버 수만큼의 계정 관리와 연결 관리를 필요로 하게 된다. 이에 따라 근본적으로 이를 대체할 새로운 서버 접속 방법과 인증 방식의 필요성이 대두되고 있다[4].

이와 같은 기존 SSH 기반 서버 관리 솔루션의 한계를 개선하고자 최근 접속 프록시(access proxy) 기반 인프라 관리 시스템이 등장하였다[11]. 이러한 프록시를 통한 관리 시스템에서는 각 서버에서 데몬 형태로 구동되는 에이전트를 두어 프록시와 통신하고, 프록시가 인증서 및 키 관리, 접근 제어 등의 서버 관리 솔루션 기능을 통합 관리하는 형태의 디자인을 갖는다. 이를 통해 접속을 요청하는 클라이언트에게는 접속 프록시만을 노출하면서도, 멀티 프로토콜에 대한 지원 및 인증 서비스를 제공한다.

본 논문에서는 이러한 현재 주목받고 있는 접속 프록시 기반 인프라 관리 시스템의 장점을 충분히 활용하여 현재의 컴퓨팅 환경에 적합한, 실용성과 확장성을 갖춘 웹쉘 인터페이스 기반 서버 관리 프레임워크를 새롭게 제안하고자 한다. 제안하는 프레임워크는 리버스 쉘 방식으로 SSH 및 PAM을 대체하고 통합 콘솔을 통해 서버에 접속, 관리하도록 함으로써 유지관리의 편의성을 제공한다. 또한 웹 서버와 클라이언트 간의 인증 방법에도 웹 인증뿐만 아니라 IDP(Identity Provider), MFA(Multi Factor Authentication) 등을 유연하게 도입할 수 있게 한다. 이를 통하여 안전하면서도 더욱 편리하게 대규모의 서버들을 관리할 수 있는 실용적인 프레임워크가 설계 가능함을 보이고자 한다.

본 논문에서는 제안하는 웹쉘 기술을 응용한 프록시 기반 인프라 관리 시스템을 구현하고 그 성능 평가를 통하여 기존 SSH 기반의 서버 관리 시스템 대비 오버헤드가 크지 않음을 보이고 이를 통해 성능 및 보안성이 고려된 시스템임을 보이고자 한다.

II. 배경지식 및 관련 연구

2.1 SSH를 통한 서버 접속 및 관리 프레임워크

SSH는 서버 접속과 관리에 표준으로 활용되고 있는 프로토콜이다[18]. 서버 접속 과정에서 사용자 인증을 위해 비밀번호 방식과 공개키 방식을 사용한다. 비밀번호는 현재까지도 많이 활용되고 있는 접속 방법이지만 여러 공격에서 취약함이 드러나면서 클라우드 서버 등에서는 공개키 인증이 주로 이루어지고 있다. SSH 클라이언트에서 개인키와 공개키 쌍을 생성하고 서버에 공개키를 등록하면 추후 개인키를 사용해 서버에 인증할 수 있는 구조이다.

SSH 공개키 인증은 브루트 포스 공격 등에 충분한 안전성을 제공하지만, 대규모의 조직에서 사용자 아이덴티티를 관리하기에는 어려움이 따른다. 개별 사용자와 서버마다 키를 관리해야 하기 때문이다. 이를 보완하고자 OpenSSH에서 독립적인

CA(Certificate Authority)를 활용한 인증 방식이 대두되었으며, 이는 클라이언트와 서버의 양방향 인증을 지원하고 인증서의 유효기간을 통해 인증 기간을 설정할 수 있는 등의 이점을 제공하고 있다[11,15].

2.2 웹쉘

본 논문에서 다루고 있는 웹쉘이란 웹페이지의 약어인 웹과 서버에게 명령을 내려 실행하기 위한 인터페이스 역할을 하는 쉘의 합성어로 웹페이지에서 서버에 명령어를 실행하기 위해 만들어진 프로그램이다. 웹쉘은 대개 사이버 공격을 위한 목적을 주로 해킹 공격에 많이 사용되어왔다[10]. 보통의 공격 과정은 공격자가 웹 서버의 취약점을 통해 악성 프로그램을 업로드하며 웹쉘 공격이 시작된다. 웹쉘 공격을 이용하면 공격자는 보안 시스템을 피해서 별도 인증 없이 시스템에 접속할 수 있다. 더 나아가 권한 상승 취약점을 이용하여 권한 상승 시 공격자는 임의의 명령어를 실행할 수 있으며 웹 서버에 대한 완벽한 통제가 가능하다.

2.3 리버스 쉘

해커들이 웹쉘 공격을 하기 위하여 웹 서버의 쉘을 접속할 때 가장 많이 사용하는 방법은 리버스 쉘이다. Fig. 1.과 같이 리버스 쉘은 대상 서버에서 외부 공격자에 연결을 요청하는 방식이며, 여기에 서버의 쉘을 연동하여 공격자가 명령을 수행할 수 있게 해준다.

해커들은 리버스 쉘을 이용하여 SSH를 우회하고 서버에 대한 접근 권한을 확보, 임의 명령을 실행하는 방법을 활용하고 있다. 대상 서버에서 연결 요청을 시작하기 때문에 대상 서버와 공격자 사이에 존재할 수 있는 방화벽 및 기타 보안 장비들을 회피하는데 용이하다[5].

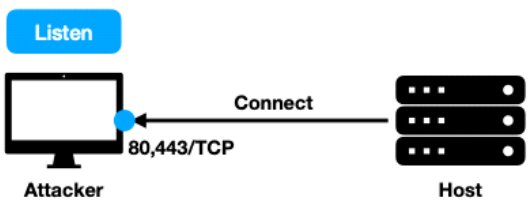

Fig. 1. Operational logic of reserve shell

III. 서버 관리 측면의 웹쉘의 활용 가능성

앞장에 살펴본 바와 같이, 웹쉘은 언제 어디에서나 브라우저를 활용하여 손쉽게 서버에 접근 가능하다는 점에서 서버 접속 및 관리 목적으로의 활용 가능성이 존재한다. 특히, 최근의 서버 인프라 환경을 고려하였을 때, 이러한 특징은 서버 관리에 있어서 더욱 큰 장점이 될 수 있다. 이 장에서는 서버 관리 체계로써 웹쉘의 활용 가능성과 활용을 위한 요구사항들에 대해서 분석하고자 한다.

3.1 웹쉘의 활용 가능성 분석

최근 웹 자바스크립트 기술의 발달로 웹에서 서버 터미널을 구현하고 활용하는 시도가 늘고 있다. 특히 AWS(Amazon Web Services) 등 클라우드 서비스 제공자는 기본적으로 웹 기반으로 클라우드 서버에 접속할 수 있는 콘솔을 제공한다[11, 12]. 하지만 이들은 SSH를 단순히 웹으로 옮겨 놓은 형태로, 각 서버의 키 및 비밀번호를 개별 관리해야 하고 기존 프로토콜을 그대로 사용하는 점 등에서 SSH와 차이가 없다.

본 연구에서는 서버 관리체계로서 웹쉘의 활용 가능성을 알아보기 위하여 리버스 쉘을 이용한 웹쉘의 장점을 분석하였다. 첫째, 리버스 쉘을 이용한 웹쉘은 SSH를 완전히 배제, 새로운 틀에서 서버 접속 프로토콜, 인증, 관리 등의 모든 체계를 구현할 수 있다는 점에서 확장성을 보인다. 또한 웹 기반 프레임워크로 확장이 가능함에 따라, 최신 웹 인증 기술, 자바스크립트 기술 등을 활용할 수 있다는 가능성까지 가지고 있다.

둘째, 리버스 쉘은 서버에 어떤 포트도 바인딩하지 않는다는 측면에서 시스템의 공격 벡터를 줄이는 효과도 있다. SSH, 텔넷과 같은 바인드 쉘은 특정 포트를 바인딩해 외부 연결 요청을 기다리는데, 브루트 포스 스캔 등에 공격받을 여지가 있다. 하지만 리버스 쉘은 이 점에서도 장점을 가진다.

3.2 서버 관리체계로의 사용을 위한 요구사항

웹쉘을 활용한 서버 관리체계 구현을 위해서는 실시간 양방향 통신과 서버 제어 행위에 대한 프로토콜 명세가 필수적이다. 이 절에서는 이를 위한 REST(Representational State Transfer)

API(Application Programming Interface)와 웹소켓의 활용 가능성을 살펴본다.

먼저 서버 관리에는 REST API를 통해 모든 관리 행위를 나타낼 수 있다. REST는 자원을 이름으로 구분하여 해당 자원의 상태를 주고받는 것을 의미한다. HTTP URI를 통해 자원을 명시하고 GET, POST, PUT, DELETE와 같은 HTTP 메서드를 통해 해당 자원을 조회, 생성, 수정, 삭제 등과 같은 행동을 하며 HTTP 메시지 페이로드를 통하여 행위의 내용을 나타낸다. 서버에서 명령을 실행하고자 할 때에는 HTTP 메시지에 명령의 내용과 대상을 담아 POST 요청을 하는 방식이다.

쉘 등의 구현을 위한 실시간 양방향 통신에는 웹소켓을 활용할 수 있다. HTTP는 서버가 클라이언트의 상태를 보존하지 않는 비연결성 프로토콜이기에 양방향 실시간 통신을 지원하지 않는다. 반면 웹소켓은 자바스크립트로 활용할 수 있으며, HTTP 환경에서 브라우저와 서버 사이에 독립된 TCP 연결을 통해 실시간 양방향 통신이 지원된다.

IV. 웹쉘을 활용한 접속 프록시 기반 서버 관리 프레임워크

이 장에서는 앞서 분석한 리버스 쉘 및 웹쉘의 개념을 이용하여 접속 프록시 기반의 서버 관리 프레임워크의 구조를 설명한다. 제안하는 프레임워크는 역방향 HTTP, 웹소켓 연결 기반의 서버 접속 방법 및 서버 관리를 지원하는 접속 프록시를 통해 여러 서버와 통신하게 된다.

4.1 접속 프록시 기반 서버 관리 시스템 개요

제안하는 시스템은 3가지의 구성 요소를 가진다. 먼저 1) **접속 프록시**는 개별 서버들과 클라이언트(사용자) 사이에 위치하여, 프록시가 인증서 및 키 관리, 접근 제어 등의 서버 관리 솔루션 기능을 통합 관리한다. **2) 에이전트**는 개별 서버 내에 설치되어 동작하며, 접속 프록시와의 통신을 통해 각 서버를 제어한다. 마지막으로 3) **웹 기반 클라이언트**를 통하여 사용자는 각 서버를 제어하게 되며, 인증된 사용자에게 웹 브라우저를 통한 쉘 기능 및 서버 제어 기능을 제공한다. 이를 통해, 개별 서버들을 관리하던 기존 방식을 Fig. 2.와 같이 **접속 프록시**를 통한 실용성과 확장성을 갖춘 통합 관리 방식으로 전환하

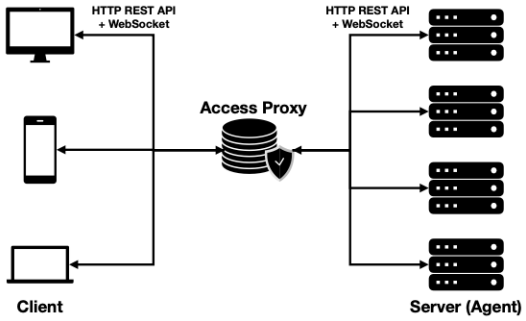

Fig. 2. Overview of access proxy based server management framework

며, 각 서버에 대한 인증과 명령 실행의 권한을 통합 콘솔 역할을 하는 접속 프록시로 이관시킨다.

각 서버에서는 데몬 형태로 구동되는 에이전트가 설치된다. 에이전트는 프록시와 연결되어 통신을 수행하며 연결에는 TLS(Transport Layer Security)를 적용하여 SSH와 동등한 수준의 보안성을 확보한다. 또한 개별 서버 안에서 동작하는 에이전트는 접속 프록시 외 다른 포트 및 원격 접속을 차단한다. 접속 프록시 내 통합 콘솔은 웹 표준 기술로 구현하여 웹 브라우저로 이용할 수 있도록 하고, 공개 API를 제공하여 확장성과 유연성, 이식성을 갖는다.

4.2 웹소켓 채널 기반 서버 제어

본 프레임워크에서는 상시적인 서버 제어가 가능하도록 접속 프록시와 에이전트 간 웹소켓 명령 채널을 사용한다. 모든 제어는 이 채널을 통해 이루어져 서버는 어떤 포트도 리스닝하지 않고 SSH를 비활성화할 수 있다. 통합 콘솔은 사용자의 요청이 있거나 스케줄에 따른 명령이 필요할 때 이 연결을 통해 명령을 전달하며, 에이전트는 이를 수행하고 그 결과를 API로 보고한다.

사용자의 서버 접속은 REST API로 지원한다. 사용자가 서버의 쉘에 접속하고자 할 때는 Fig. 3.과 같이 POST로 새 세션을 요청할 수 있으며, 이때 통합 콘솔은 클라이언트와 에이전트에게 동시에 세션에 접속할 수 있는 웹소켓 URL을 전달한다. 에이전트는 이 웹소켓 URL로 통합 콘솔에 새로운 연결을 맺으며, /bin/bash 등 사용자가 지정한 쉘을 실행하여 표준 입출력을 웹소켓 입출력에 연동시킨다. 이때 연결에 사용되는 웹소켓은 에이전트에서 통

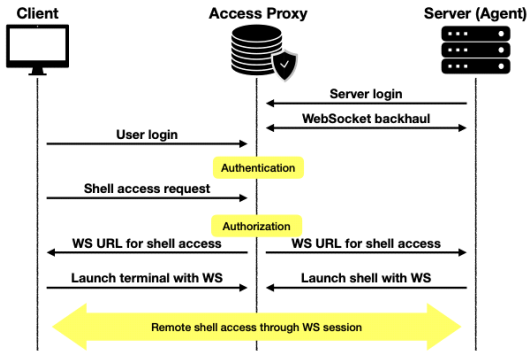

Fig. 3. Procedure of establishing the connection between client and server through the access proxy

합 콘솔 방향 접속이기 때문에, 별도의 포트를 사용하지 않고 인바운드 방화벽에도 영향받지 않는다.

4.3 사용자 계정 및 인증

사용자 계정은 통합 콘솔의 IAM(Identity Access Management) 데이터베이스의 사용자 정보로 일괄 관리한다. 접속하고자 하는 서버에 계정이 없는 경우 API 요청 즉시 데이터베이스의 username, uid, shell, home directory 등 정보로 서버에 계정이 생성되며, 이후 쉘 접속이 가능하게 된다.

통합 콘솔은 이 과정에서 사용자가 해당 서버에 실제 접속할 수 있는지 인증과 권한을 검증한다. 이 인증과정을 개별 서버가 아닌 통합 콘솔에서 대신하면서 다양한 현대화된 웹 기반 아이덴티티 인증 방식을 사용할 수 있다. 예를 들면 비밀번호에 MFA가 적용된 방식을 포함하여 Auth0[6]와 같은 IDP를 활용할 수 있는 것이다. 서버에 대한 접속 권한은 API를 통해 어떤 사용자 또는 그룹이 해당 서버에 접속할 권한을 가지는지 정의된다.

4.4 웹 기반 클라이언트

인증된 사용자는 브라우저와 웹 프로토콜을 지원하는 도구를 통해 통합 콘솔에 접속하여 원하는 서버의 웹쉘을 사용할 수 있다. 클라이언트는 서버 관리를 위해 등록된 서버들 목록 및 개별 서버의 정보들을 보여줄 수 있는 서버 관련 페이지와 사용자 계정 및 그룹 관리를 할 수 있는 IAM 관련 페이지들로 구성된다. 통합 콘솔과 REST API를 통해 데이터를 주고받으며, 이를 활용할 수 있는 웹 프레임워크를 사용하여 구현할 수 있다.

클라이언트는 사용자가 서버의 웹쉘 페이지에 진입하면 통합 콘솔에 웹쉘 연결을 위한 웹소켓 URL을 요청한다. 이후 클라이언트는 자바스크립트 Xterm.js[7]와 같은 터미널 인터페이스 생성 도구를 사용하여 웹쉘을 생성한 뒤 전달받은 웹소켓 URL에 연결하여 서버 터미널에 접속을 제공한다. 추후 자바스크립트의 확장성 및 이식성을 활용하여 웹쉘에서 파일을 클릭하여 웹 코드 에디터를 실행하는 등 다양한 기능을 추가할 수 있다.

V. 구현 및 성능 평가

이 장에서는 다음의 두 가지 성능 관점에서의 질문들을 통하여 제안하는 접속 프록시 기반의 서버 관리 시스템의 적용 가능성 및 효율성을 보이고자 한다.

- ◆ Q1. 접속 프록시를 사용하는 구조로 인하여 최초 접속 과정에서의 인증 및 세션 수립과정으로 인한 추가적인 지연이 발생하는가?
- ◆ Q2. 접속 프록시를 사용하는 구조로 인하여 서버 사용 과정에서 발생하는 추가적인 오버헤드가 있는가?

또한, 이 장에서는 위의 두 가지 질문들의 평가와 함께, SSH를 통한 서버 관리 기술과의 정성적 비교를 통하여, 제안하는 시스템의 특징을 평가하고자 한다.

5.1 구현 및 평가 환경

평가를 위하여 연구에서 제안하는 시스템을 구현하였다. 접속 프록시의 통합 콘솔은 Django[16]를 사용한 프레임워크로 Python으로 구현하였다. 통합 콘솔 내에는 IAM 및 서버 정보 등의 처리를 위해 데이터베이스로는 PostgreSQL[17], 각 모듈 간 통신을 위한 메시지 브로커로는 Redis를 사용하여 구현하였다. 에이전트는 Python를 사용하여 구현하였으며, 웹 클라이언트는 React 기반 PWA(Progressive Web App)로 구현하였다.

결과적으로 제안하는 시스템 구현을 위해 총 18,320 SLoC을 작성하였으며, 통합 콘솔, 에이전트는 각각 16,760, 1,560 SLoC를 가진다.[1] 웹 클

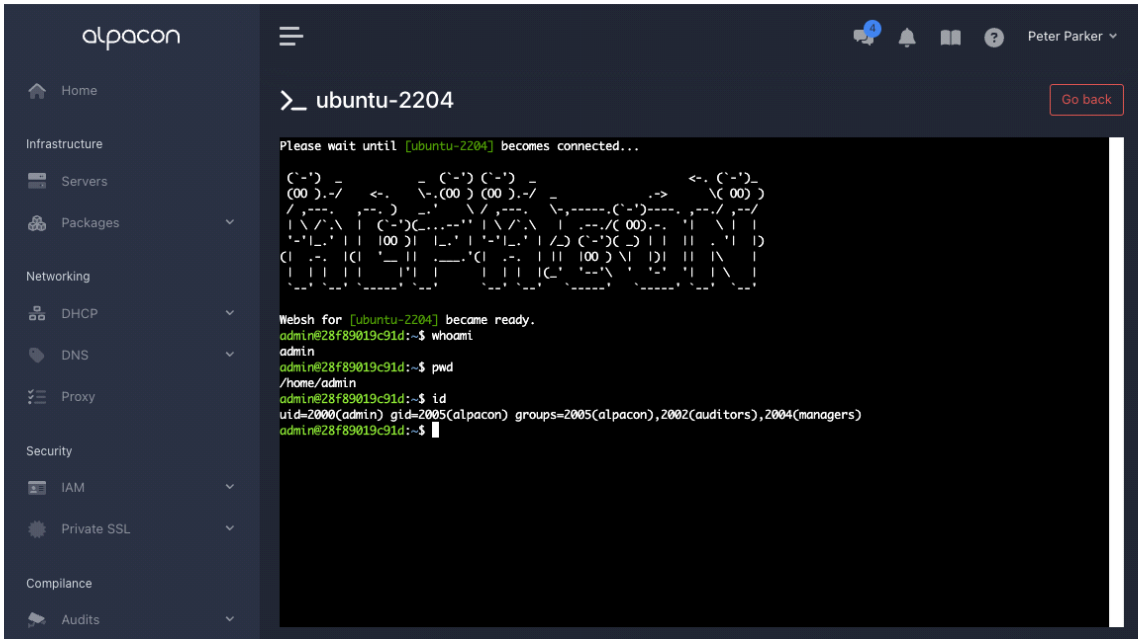

Fig. 4. Snapshot of web client implementation (PWA)

라이언트의 코드 크기는 별도로 측정하지 않았다.

연구에서는 접속 프록시 역할을 하는 1대의 서버, 에이전트가 설치되는 2대의 서버, 그리고 사용자 역할을 하는 웹 클라이언트를 위한 1대의 PC를 사용하였다. 평가에 사용한 서버들은 Intel Core m3-8100Y @ 1.1GHz CPU, 8G RAM으로 구성되어 있다. 접속 프록시와 관리되는 서버들은 Ubuntu 20.04를 사용한다. 웹 클라이언트는 macOS 운영체제가 설치된 PC에 설치된 Chrome (Version 114) 브라우저를 사용하였다. Fig. 4.는 제안하는 시스템 구현의 결과로써 사용자 웹 클라이언트의 화면을 보여준다.

5.2 최초 접속 시간

본 연구에서 제안하는 접속 프록시 기반 서버 관리 프레임워크에서는 초기 접속 과정에서 SSH를 통한 접속 과정과 비교하여 2단계의 추가적인 과정이 발생한다. Fig. 3.에서 소개한 바와 같이, 사용자가 접속 프록시의 인증을 통과한 이후, 1) 서버 내 쉘 접속을 요청하는 과정과 2) 웹쉘 기능 제공을 위한 웹소켓 생성 및 세션 연동 과정이 요구된다.

이러한 추가적인 과정에 의한 지연시간을 측정하기 위해 연구에서는 크롬 개발자 모드를 사용하여 각 과정에 소요되는 시간을 10회 측정하였다. 그 결과, 사용자가 프록시에 인증하는 과정에 평균 478ms가 소요되었으며, 첫 번째 단계인 쉘 접속 요청과정에는 65ms가 소요되었다. 이후, 두 번째 단계인 웹소켓 생성을 통한 웹쉘 환경 제공에 110ms가 소요됨을 확인할 수 있었다. 따라서 총 653ms가 최초 접속에 필요하게 되며, 사용자 프록시 인증 이후부터는 각 서버에 접속 시 175ms가 소요됨을 확인할 수 있다.

성능 비교를 위하여 같은 네트워크 환경에서 인증서 기반의 사용자 인증을 사용하는 SSH를 사용하여 SSH에서 초기 접속 과정에 소요되는 시간을 10회 측정한 결과 평균 367ms가 소요됨을 확인하였다.

주목할 점은 사용자가 프록시에 인증하는 과정은 최초에 사용자가 서버 관리 프레임워크로의 접속 과정에만 발생하며, 그 뒤로는 발생하지 않는다. 또한, 여러 개의 서버에 접속하는 환경을 고려하였을 때, SSH의 경우 반복적으로 사용자 인증 과정을 반복해야 하는 반면, 제안하는 접속 프록시 기반 서버 관리 시스템에서는 인증과정은 웹 클라이언트를 통하여 접속 프록시에 접속하는 과정에 수행되며, 이후 웹쉘을 통해 접속할 때는 요구되지 않는다.

1) LoC는 순수 코드 크기로 sloccount로 측정한 수치

Table 1. Comparison of SSH and Access Proxy-based Server Management Frameworks

	SSH	Proxy-based approach
Key management & user authentication	Key rotation and management is required as servers and users change since SSH keys do not have expiration date, keys are valid until retrieved	Key management is not required, since all authentication between server and user is established through access proxy
Least privilege	No native support for least privilege	Provides detailed user group and permission configuration
Auditing log & monitoring	Integrated management of distributed logs is challenging since separate logging system is required for each server	Easy to provide integrated audit/log for all servers within management
Session management	Tracking and disabling sessions per device is hard as SSH is stateless application.	Easily extensible for session recording and managing
Multi-factor authentication	Requires installation and configuration of additional modules such as PAM	Easy to add various authentication methods between access proxy and web-based client

5.3 상시 사용상태에서의 지연시간

다음으로, 연구에서는 초기 접속 이후 서버 쉘 사용 과정에서 발생하는 추가 지연시간을 측정하였다. 접속 프록시를 통한 통신의 경우, 1 hop delay가 추가로 발생하며, 이를 측정하기 위해 Fig. 5.와 같이 웹쉘 세션용 웹소켓 연결의 메시지별 타이밍을 분석하였다. 쉘 특성상 매 타이핑한 글자마다 동일한 메시지가 서버로부터 에코되며, 이 타이밍 차이가 지연시간이 된다. 30회 측정 후 분석 결과 지연시간은 평균 27ms로 프록시 연결에 TLS가 사용됨에도 충분히 빠른 속도를 보여 사용성에 문제가 없었다.

Data	Length	Time
↑ c	1	00:29:42.300
↓ c	1	00:29:42.328
↑ a	1	00:29:42.400
↓ a	1	00:29:42.415
↑ t	1	00:29:42.523
↓ t	1	00:29:42.551

Fig. 5. Latency results of the WebSocket message transmission timing analysis

5.4 SSH를 통한 서버 관리 방법과의 정성적 비교

이 장에서는 접속 프록시 기반의 서버 관리 프레임워크가 가지는 장점들을 기존 SSH를 통한 서버 접속 및 관리 기술과 비교한다.

먼저 접속 프록시는 통합 IAM을 통해 SSH의 키 관리 문제의 효율성과 보안성을 개선한다. 서버와 사용자별 키를 관리해야 하는 SSH와 달리 통합 IAM은 일원화된 사용자 계정 및 서버 인증정보를 통해 키 관리 어려움을 해소한다. 또한 서버와 사용자 규모의 확장에도 사용자 그룹을 통한 권한 설정으로 민첩하게 대응함으로써 관리 부실에 따른 보안사고를 미연에 방지할 수 있다.

프록시 구조는 또한 얻을 수 있는 여러 장점이 존재한다. 먼저 접속 프록시가 사용자와 서버 간 통신을 중계함에 따라 사용자의 활동 내역과 실행 명령 등이 접속 프록시에 기록된다. 현대적인 기업의 조직 구조에서는 권한에 따른 통제가 중요한데, 서버 접근 기록을 통해 세션 이력 관리, 자동화된 모니터링, 권한 감사 등 추가기능을 구현하기 용이하다.

한편 접속 프록시는 SSH와 비교하여 동등한 수준의 프로토콜 보안성을 보장한다. 접속 프록시가 연결에 사용하는 Secure WebSocket, HTTPS는 TLS 1.2 이상에 기반하여 사용자-프록시, 프록시-서버 간의 종단간 암호화를 지원하며 MiTM을 차단한다. SSH 프로토콜 또한 TLS에 기반하여 종단간 암호화를 지원한다. 본 논문에서는 접속 프록시는 신뢰할 수 있다고 가정하며, 이에 따라 사용자와 서버 간의 종단간 보안성은 SSH와 동등하다고 할 수 있다.

VI. 논의 및 고찰

한편, 본 제안으로 서버 관리체계가 통합 콘솔로 일원화되면 통합 콘솔이 단일 장애점이 될 수 있다는 우려가 있을 것으로 생각한다. 하지만 복잡화된 현대 서버 인프라의 보안을 개별적으로 어렵게 관리하는 것보다 관리지점을 일원화하는 것이 실용적인 접근이라 생각한다. 또한, 앞서 기술한 바와 같이 TLS, MFA가 적용된 IDP, AI 기반 명령어 모니터링 등을 사용하여 여러 단계의 보안 메커니즘과 감사 체계를 구현함으로써 이러한 우려를 방지할 수 있으며, 본 기술의 보안성 분석과 확장 가능성에 대해 향후 연구를 지속하고자 한다.

접근 제어를 관장하는 접속 프록시와 에이전트 기반의 서버 관리 시스템의 경우, 접속 프록시가 각 서버에서 구동되는 데몬 에이전트를 신뢰할 수 있을 것인가 또한 중요한 보안 이슈이다. 구체적으로, 에이전트의 기밀성 및 무결성을 보장하여 악의적인 공격자가 에이전트 프로그램의 실행 흐름을 탈취하거나, 기밀을 유출하는 등의 이상 행위가 발생하는지 파악할 수 있는 감사 메커니즘이 필요하며, 키 및 인증서 등 기밀 데이터들에 대한 보안성 또한 접속 프록시 및 에이전트 프로그램이 보장하여야 한다.

이에 대한 해결책으로 최근 클라우드 컴퓨팅에서 주목받고 있는 기술인 기밀 컴퓨팅(confidential computing)을 활용할 수 있다[13]. 하드웨어 기반 신뢰 실행환경이 보장하는 암호화된 격리 실행환경에서 에이전트들을 구동시키고, 원격 검증(remote attestation)을 통해 각 에이전트 프로그램의 기밀성 및 무결성을 검증한다. 이후 TLS 및 DHKE (Diffie-Hellman Key Exchange) 와 같은 암호화 프로토콜을 원격 검증 프로토콜과 결합하면 제로 트러스트 기반의 암호화된 채널 구축이 가능하다. 개발한 접속 프록시 및 데몬 에이전트들을 Intel SGX[14]와 같은 상용 기밀 컴퓨팅 기술을 활용하여 개발하는 후속 연구를 진행할 계획이다.

VII. 결 론

본 논문에서는 클라우드 기술 혁신에 따라 발전된 서버 백엔드 인프라 기술과 전통적인 서버 접속 방법인 SSH 프로토콜 간 기술적 간극을 줄이고 SSH 프로토콜의 한계점을 개선하기 위해 웹쉘을 활용한 프록시 기반 서버 접속 및 관리 프레임워크를 제안하였다. 일반적으로 공격의 목적으로 활용되는 웹쉘의 탄력적인 사용성 및 실용성에 주목하여 서버 관리체계를 다변화된 현재 컴퓨팅 환경에 맞게 설계하는 데 활용하였다. 이를 통해 서버 관리의 생산성을 향상시키고 표준 웹 프로토콜로 확장성 및 이식성을 확보할 수 있음을 보였다. 또한, 웹 기반 터미널은 텍스트 기반 UI의 한계에서 벗어나 웹 기술을 결합한 새로운 형태의 UX를 제공할 수 있을 것으로 기대한다. 본 논문에서 구현한 프레임워크의 소스코드는 GitHub에 오픈소스로 공개될 예정이다.

References

[1] Protect against brute-force/dictionary SSH attacks, https://www.cmu.edu/iso/aware/be-aware/brute-force_ssh_attack.html, Accessed: Jul. 2023. [Online]

[2] D.W. Chadwick, R. Laborde, A. Oglaza, R. Venant, S. Wazan, and M. Nijjar, "Improved identity management with verifiable credentials and FIDO," IEEE Communications Standards Magazine, vol. 4, no. 3, pp.14-20, Dec. 2019.

[3] Problems with PAM, https://www.dtucker.net/pam/, Accessed: Jul. 2023. [Online]

[4] T.T. Stöcklin, "Evaluating SSH for modern deployments," Thesis, Noroff University College, May 2022.

[5] A. Hannousse and S. Yahiouche, "Handling webshell attacks: A systematic mapping and survey," Computers & Security, vol. 108, no. 102366, Jun. 2021.

[6] Auth0, https://auth0.com/, Accessed: Jul. 2023. [Online]

[7] Xtermjs, https://xtermjs.org/, Accessed: Jul. 2023. [Online]

[8] Dan Wendlandt, David G. Andersen and Adrian Perrig, "Perspectives: improving SSH-style host authentication with multi-path probing," Proceedings of the USENIX

2008 Annual Technical Conference, pp. 321-334, Jun. 2008.
〔9〕 Ogundoyin, Sunday Oyinlola, and Ismaila Adeniyi Kamil. "PAASH: A privacy-preserving authentication and fine-grained access control of outsourced data for secure smart health in smart cities," Journal of Parallel and Distributed Computing, vol. 155, pp. 101-119, Sep. 2021.
〔10〕 H. Zhang, H. Guan, H. Yan, W. Li, Y. Yu, H. Zhou and X. Zeng, "Webshell traffic detection with character-level features based on deep learning," IEEE Access, vol. 6, pp. 75268-75277, Nov. 2018.
〔11〕 Teleport, https://goteleport.com/, Accessed: Jul. 2023. 〔Online〕
〔12〕 AWS, https://aws.amazon.com/, Accessed: Jul. 2023. 〔Online〕
〔13〕 Fahmida Y. Rashid, "The rise of confidential computing: big tech companies are adopting a new security model to protect data while it's in use - 〔news〕," IEEE Spectrum, vol. 57, no. 6, pp. 8-9, Jun. 2020.
〔14〕 Costan, Victor and Srinivas Devadas, "Intel SGX explained," Cryptology ePrint Archive 2016-086, Jan. 2016.
〔15〕 Vault, https://www.vaultproject.io/, Accessed: Jul. 2023. 〔Online〕
〔16〕 Django, https://docs.djangoproject.com/, Accessed: Jul. 2023. 〔Online〕
〔17〕 Postgresql, https://www.postgresql.org, Accessed: Jul. 2023. 〔Online〕
〔18〕 SSH rfc, https://datatracker.ietf.org/doc/html/rfc4253, Accessed: Jul. 2023. 〔Online〕

〈저 자 소 개〉

김 다 은 (Daeun Kim) 학생회원
2019년 3월~현재: 성신여자대학교 융합보안공학과 학사
〈관심분야〉 정보보호, 시스템 보안, 네트워크 보안

배 상 욱 (Sangwook Bae) 정회원
2013년 2월: 한국과학기술원 전기 및 전자공학부 졸업
2015년 8월: 한국과학기술원 전기 및 전자공학부 석사
2022년 8월: 한국과학기술원 전기 및 전자공학부 박사
2022년 9월~현재: 알파카네트웍스(주) 플랫폼개발팀
〈관심분야〉 이동통신 보안, 신뢰 실행 환경, 시스템 보안

김 성 민 (Seongmin Kim) 종신회원
2012년 2월: 한국과학기술원 전기 및 전자공학과 졸업
2014년 2월: 한국과학기술원 전기 및 전자공학과 석사
2019년 2월: 한국과학기술원 정보보호대학원 박사
2019년 9월~2020년 8월: 삼성전자 삼성리서치 Staff Engineer
2020년 9월~현재: 성신여자대학교 융합보안공학과 조교수
〈관심분야〉 신뢰 실행 환경, 클라우드 컴퓨팅, 시스템 보안

정 은 영 (Eunyoung Jeong) 정회원
2012년 2월: 한국과학기술원 전기 및 전자공학과 졸업
2014년 2월: 한국과학기술원 전기 및 전자공학과 석사
2013년 12월~현재: 국가보안기술연구소 선임연구원
2022년 7월~현재: 알파카네트웍스(주) 대표이사
〈관심분야〉 네트워크 시스템, 네트워크 보안, 시스템 보안