KIPS Trans. Comp. and Comm. Sys.
Vol.11, No.4 pp.119~126
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2022.11.4.119

Secure and Efficient Package Management Techniques in Closed Networks

Gun-Hee Ahn† · Sang-Hyuk An† · Dong-Kyun Lim† · Su-Hwan Jeong† · Jaewoo Kim†† · Youngjoo Shin†††

ABSTRACT

In this paper, we present important factors and methodologies that we have to follow for secure and efficient package management systems in a closed network. By analyzing previous works, we present several security considerations for the existing package management systems. Based on the consideration, we propose guidelines regarding the use of package management systems in the closed network. More specifically, we propose the development of new package management tools, utilization of physical storage media, utilization of local backup repositories, package updates, and downgrade batches for secure and efficient package management.

Keywords : Closed Network, Package Management System

폐쇄망에서의 안전하고 효율적인 소프트웨어 패키지 관리 방안

안 건 희† · 안 상 혁† · 임 동 균† · 정 수 환† · 김 재 우†† · 신 영 주†††

요 약

본 연구는 폐쇄망에서 효율적이고 안전하게 패키지 관리 시스템을 사용하기 위해서 고려해야 할 할 중요 요소들과 그 방법론 들을 제시하는 것을 목적으로 한다. 관련 선행 연구의 분석을 통해 기존 패키지 관리에서 보안성을 위해 고려해야 할 사항들을 살펴보고, 이를 바탕으로 폐쇄망이라는 특수한 상황에서 고려해야 할 세부 방법들을 제안한다. 구체적으로, 새로운 패키지 관리 도구의 개발, 물리적 저장매체 활용, 로컬 백업 저장소 활용, 패키지 업데이트 및 다운그레이드 일괄 처리의 방법을 제안한다.

키워드 : 폐쇄망, 패키지 관리 시스템

1. 서 론

리눅스 운영체제는 패키지 관리 시스템 (Package management system)을 통해 각종 소프트웨어를 패키지화하여 편리하고 효율적으로 설치 및 사용할 수 있도록 지원하고 있다. 패키지 관리 시스템은 원격 위치에 별도의 패키지 저장소를 두고 인터넷을 통해 필요한 패키지를 다운로드받아 리눅스 시스템에 설치하는 방식을 취하고 있다. 모든 패키지를 리눅스 배포판에 담을 필요가 없으므로 매우 효율적인 리눅스의 설치가 가능하지만, 인터넷이 연결된 환경에서만 패키지 관리 시스템의 사용이 가능하다는 문제가 존재한다. 또한, 인터넷을 통한 패키지 다운로드 과정에서 해커들의 각종 공격에 노출될 수 있는 보안 문제도 가지고 있다. 이러한 보안 문제들을 이유로 기업이나 군사 시설에는 폐쇄망을 사용하여 네트워크망을 분리, 고립시켜 외부로부터 유입되는 보안 위협을 제거하는 방법을 선택하는 경우가 빈번하다.

본 논문에서는 리눅스의 패키지 관리 시스템에 대한 분석과 현존하는 보안 취약점들에 대해 살펴본다. 즉, 각 패키지 관리 시스템별로 원격 패키지 저장소에서 패키지를 다운로드 받아 설치할 때 패키지 무결성 등 검증 과정이 제대로 진행되는지를 비롯하여 각종 네트워크 공격에 취약할 수 있는 보안 위협 요소들을 분석해 본다.

이러한 분석을 바탕으로, 패키지 관리 시스템의 보안을 강화할 수 있는 여러 가지 방안들을 제시한다. 이와 함께, 인터넷이 연결되지 않은 폐쇄망 환경에 대해서도 안전하고 효율

※ 이 논문은 2021년도 한화시스템(주)의 제원을 지원받아 수행된 연구임.
† 비 회 원 : 고려대학교 사이버국방학과 학사과정
†† 비 회 원 : 한화시스템 수석연구원
††† 정 회 원 : 고려대학교 정보보호대학원 조교수
Manuscript Received : August 23, 2021
Accepted : October 20, 2021
* Corresponding Author : Youngjoo Shin(syoungjoo@korea.ac.kr)

```
vared@ubuntu:~/Test$ binwalk zlib-1.2.7-19.el7_9.x86_64.rpm

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
-
0             0x0             RPM v3 bin i386 "zlib-1.2.7-19.el7_9"
12740         0x31C4          Copyright string: "Copyright/License/"
13756         0x35BC          Unix path: /usr/share/doc/zlib-1.2.7/
14340         0x3804          xz compressed data
```

Fig. 1. Content of a RPM File

적으로 패키지 관리 시스템을 사용할 수 있는 방법들을 제시한다. 구체적으로, 폐쇄망 환경에서 패키지 관리 시스템을 이용할 때 고려해야 할 4가지의 방법을 제안한다. 즉, 새로운 패키지 관리 도구의 개발, 물리적 저장매체 활용, 로컬 백업 저장소 활용, 패키지 업데이트 및 다운그레이드 일괄 처리의 방법 등을 상세하게 제시한다.

폐쇄망은 인터넷과 단절된 지극히 제한적인 환경이라는 점에서 많은 한계가 존재하지만 이러한 환경 때문에 추가로 개선 방안에 관한 연구가 많이 필요하다. 따라서 향후 특정 환경을 제시한 다음 이에 맞게 패키지 관리 서비스를 자체 제작해보거나, 패키지 내부를 검증하는 과정에서 악성코드를 탐지하는 등 새로운 연구 방향도 같이 제시한다.

본 논문의 주요 내용은 다음과 같이 크게 3가지이다.

- 첫째, 현재 많은 리눅스 배포판에서 사용되고 있는 RPM과 DEB 기반 패키지 관리 시스템들의 보안 메커니즘을 분석한다.
- 둘째, 패키지 관리 시스템의 취약한 활용으로 인해 발생할 수 있는 보안 취약점을 분석한다.
- 셋째, 폐쇄망에서 패키지 관리 시스템들을 안전하게 사용해 폐쇄망을 안전하게 유지할 수 있는 방법들을 제시한다.

본 논문의 구성은 다음과 같다. 먼저 II 장에서 논문의 배경지식과 관련 연구에 관해 기술한다. 그리고 III 장에서는 폐쇄망에서의 패키지 관리 시스템의 안전하고 효율적인 관리 방안을 제안한다. 마지막으로 IV 장에서 결론으로 끝을 맺는다.

2. 배경 지식 및 관련 연구

리눅스 운영체제에서는 각 배포판에 따라 서로 다른 패키지 형식을 지원한다. 크게 Debian 계열과 RedHat 계열로 나눌 수 있으며 RedHat 계열에서는 .rpm 파일 패키지 형식을 지원한다. 이러한 패키지 형식이 등장한 배경은 기존에 사용하던 tar.gz 파일들을 수동으로 설치하는 데 불편함이 존재함에 기인한다.

각각의 패키지 파일을 설치하는 도구도 다른 조합으로 구성되어 있으며. 실제 패키지를 설치하는 Low-Level Tool과 High-Level Tool로 구분할 수 있다. Low-Level Tool은 실질적인 패키지의 설치와 업데이트 삭제와 같은 기능을 수행하고, High-Level Tool은 패키지 간 의존성 문제 해결, 패키지 검색과 같은 부가적인 기능을 담당하게 된다. RedHat 계열에서는 RedHat Package Manager (RPM)를 Low-Level로 사용하고, Yellow Dog Updater (YUM)를 High-Level로 사용하게 된다.

2.1 RPM (Redhat Package Manager) 패키지 구조

RPM[1-4] 파일의 내용은 binwalk 와 같은 도구를 통해 쉽게 확인할 수 있다(Fig. 1 참고). RPM 파일의 구조는 Fig. 2와 같이 Lead, Signature, Header 그리고 Payload의 4가지 구역으로 나눌 수 있다. 각 구역의 주요 특징은 다음과 같다.

- Lead : 해당 파일 포맷의 버전, 아키텍처, 유효한 OS와 같은 기본적인 정보들을 가지고 있다.
- Signature : 서명 정보를 가지고 있는 영역으로, 해당 서명 값은 헤더와 페이로드 정보만을 구성하여 생성되는 값이다. Lead와 Signature의 헤더 부분은 포함되지 않으며 검사의 대상 역시 아니다. 이 점이 취약한 부분이 될 수 있으나 Signature 헤더가 변형되는 경우 무결성이 훼손되고, Lead 영역이 변형되는 경우 패키지가 유효한 상태로 인식되지 않기 때문에 취약점으로 인식되지는 않는다. RPM에서는 서명이 필수적인 작업 과정

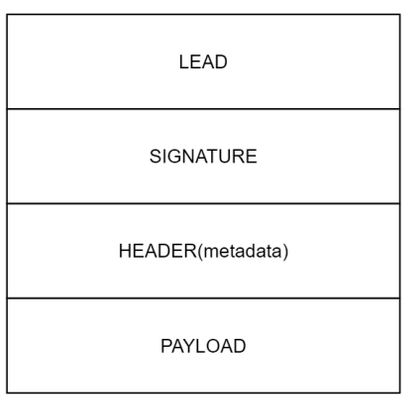

Fig. 2. Structure of a RPM File

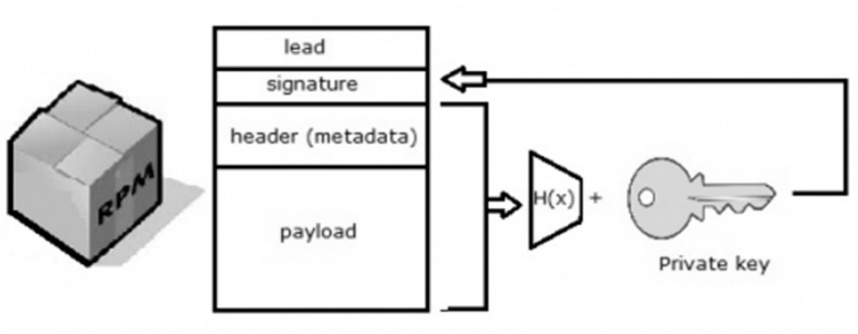

Fig. 3. Signature Structure of RPM

은 아니므로 해당 영역은 선택적으로 생성될 수 있는 영역이다.

- Header : RPM 파일이 가지고 있는 패키지에 대한 정보를 저장해놓은 영역이다. 구조 자체는 Signature 영역과 유사하나, 태그의 개수가 세부적이고 다양하게 존재한다는 특징을 가진다는 점에서 패키지의 세부 정보를 저장하는 영역이다.
- Payload : 실질적인 패키지 구성 파일들이 저장된 영역이다. tar.gz 파일의 구성요소들이 저장되어 있으며, GNU Zip을 사용하여 압축된 상태이다. 압축 해제된 이후에는 cpio 형식과 CRC Checksum 값으로 구성된다.

RPM과 관련된 보안 기능으로는 서명을 검증하는 옵션과, 특정 패키지에 서명하는 옵션이 존재한다. 이때 서명은 패키지 내부에 저장되게 된다. 서명 시 사용하는 개인키는 패키지 제작자의 GnuPG 개인키이며, Fig. 3의 과정처럼 메타데이터와 패키지 영역을 해시 함수의 입력값으로 준 다음 이에 서명하는 방식이다[5]. 해시 함수의 알고리즘은 주로 MD5, SHA1을 사용하며 서명 방식으로는 RSA, DSA, ElGamal의 조합을 선택하여 사용할 수 있다.

RPM 패키지 자체에 대한 보안을 제외한 RPM 패키지를 관리하는 영역, 즉 RPM 패키지를 업로드하고 다운로드하는 저장소에 대한 보안 기능은 별도로 존재하지 않았으며 저장소 별로 특정 기업의 정책에 따라 관리하는 매뉴얼이 존재하는 경우는 있었다. 다운로드하여 설치하는 과정에서도 서명을 통해 패키지의 무결성을 검증하는 과정은 있으나, 패키지 자체의 보안 요소를 점검하는 과정은 없다.

2.2 DEB 패키지 구조

Debian 계열의 패키지는 Debian Package (dpkg)[6]를 Low-level tool로 사용하고 Advanced Packaging Tool (APT)를 High-level tool로 사용한다.

debian 계열의 패키지는 확장자가 .deb인 파일의 형태로 저장되어 사용하는데, 이 deb 파일의 기본적인 구조는 Fig. 4

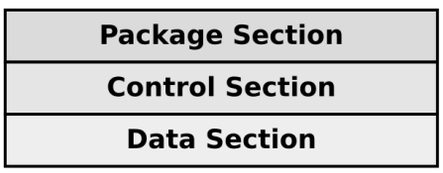

Fig. 4. Structure of Debian Package (deb) File

와 같이 Package Section, Control Section, Data Section으로 나뉘어있다. 각 Section 별 특징은 다음과 같다.

- Package Section : Package Section이 시작되기 전에는 간단한 .deb파일의 시그니쳐가 있고, Package Section에는 해당 .deb 파일의 정보들과 패키지 포맷의 버전 정보가 담겨있다.
- Control Section : Control Section에는 현 패키지에 대한 상세 설정이나 의존성 등의 정보를 담은 파일들이 .tar.gz 혹은 .tar.xz 형태로 아카이빙되어 저장되어 있는데, 해당 섹션에는 control, md5sums, conffiles, preinst, postinst, prerm, postrm, config, shlibs 등등의 다양한 파일들이 아카이빙되어있다. control 파일은 해당 패키지의 이름, 버전, 설치 용량, 의존성 등등의 종합적인 패키지의 정보들을 모아놓은 파일이다. md5sums 파일은 해당 패키지 내에 존재하는 파일들의 md5 hash들을 저장해, 패키지의 무결성을 증명하는 데 쓰인다. 또, conffiles는 패키지 내에서 설정 파일로 구성되어야 하는 파일들을 열거하는데, 이 파일 내에 존재하는 파일들은 따로 지정하지 않는 이상 패키지 업데이트 중에 변경되지 않는다. preinst, postinst, prerm, postrm 파일들은 각각 패키지 설치 전, 후 및 패키지 삭제 전, 후로 실행되는 필수가 아닌 선택적 쉘 스크립트 파일이다. 또, config 파일은 단순히 debconf의 설정 메커니즘을 따르는 설정 파일이며, shlibs 파일은 공유 라이브러리 의존성에 대한 정보들을 열거한 파일이다.
- Data Section : Data Section에는 실제 패키지 내에

존재하는 파일들을 .tar.gz 혹은 .tar.xz의 형태로 아카이빙한 파일이다. 해당 섹션을 압축해제하게 되면 패키지에 존재하는 모든 파일이 설치된다.

Control Section과 Data Section은 모두 각 Section의 실제 아카이빙 데이터가 쓰이기 전에 해당 아카이빙 데이터에 대한 정보들(타임 스탬프, 파일 권한, 소유자, 파일 크기, 파일 이름 등)들이 적혀있다. deb 패키지 역시도 RPM 패키지와 같이 서명을 진행할 수 있는데, RPM과 비슷하게 GnuPG 개인키를 사용하여 서명을 진행한 후, 파일 내에 signature를 삽입하여 서명 인증을 진행한다.

2.3 주요 패키지 관리 시스템에 대한 공격 기법

선행 연구에 의하면 RPM과 DEB 방식의 패키지 관리 시스템들에 대한 다음과 같은 공격방식이 제시된 바 있다[7,8].

- Slow Retrieval 공격: 이 공격은 연결이 오픈된 상태에서 데이터를 전송하지 않거나 매우 느리게 전송하는 방식으로 DoS를 유도하는 공격이다. 기존의 APT, YUM은 이러한 정보에 대한 로그를 남기지 않기 때문에 가용하다.
- Endless Data 공격: 첫 번째 방식과 유사하게, 무한정의 데이터 스트림을 보내는 방식이다. 이 방식으로 마찬가지로 DoS 공격이 가능하다. 더 나아가서, 시스템 크래시까지도 가능하다.
- Replay Old Metadata 공격: 이는 취약한 패키지가 있던 당시의 루트 메타데이터를 replay 하는 공격을 말한다. 이는 메타데이터에 서명하더라도, 날짜 정보를 검사하지 않기 때문에, replay 하는 방식으로 검증을 우회할 수 있는 공격이다.
- Extraneous Dependencies 공격: 이 공격은 패키지에 대한 거짓 의존성 정보를 작성하는 공격이다. 즉, 올바르지 않은 외부 패키지에 대해서 의존적이게 만드는 것이다.
- Depends on Everything 공격: 이는 특정 패키지가 엄청 많은 수의 의존성을 가지고 있다고 거짓 정보를 작성하는 공격을 말한다. 따라서 해당 패키지를 다운로드 시, DoS나 시스템 크래시가 발생할 수 있다.
- Unsatisfiable Dependencies 공격: 이는 다섯 번째 공격방식과 유사하다. 많은 수가 아니라 불가능한 의존성을 가지고 있다고 거짓 정보를 작성하는 것을 말한다.
- Provides Everything 공격: 이 공격은 특정 패키지가 많은 양의 가상 의존성을 제공할 수 있는 것처럼 메타데이터를 조작하는 공격을 말한다. 즉, 특정 가상 의존성을 요구하는 패키지가 다운로드 될 시, 이 조작된 패키지가 다운로드 되도록 하는 공격이다.
- Use Revoked Keys 공격: 이 공격은 만료된 키를 이용하여 사용자들이 패키지를 다운로드 받도록 하는 공격이다. 이는 만료된 키를 제거하는 과정은 수동으로 진행되기 때문에 오래 걸리는 점을 이용한 공격이다.

3. 폐쇄망에서의 패키지 관리 방법 제안

폐쇄망이라는 특수한 환경에서도 패키지 관리를 위한 방법들이 마련되어 있어야 한다. 여기서 폐쇄망은 외부 환경과 완벽하게 분리된 상태의 네트워크망을 의미한다. 이러한 환경에서 안전하게 패키지를 관리하고, 설치하기 위해서는 기존 패키지 관리 방법과는 다른 방법이 필요하다.

앞서 살펴본 기존 연구를 보면 APT나 YUM의 가용한 취약점들뿐만 아니라 보안성 만족을 위하여 고려해야 할 6가지 사항도 언급하고 있다. 이를 정리하면 다음과 같다.

첫째, 통신 과정에서 파일 크기, 날짜 정보등을 검증해야 한다. 이는 앞서 언급한 정보들을 이용하여 DoS나 크래시를 일으키는 공격을 방지하기 위함이다. 이를 검사함으로써 slow retrieval 공격과 endless data 공격을 방지할 수 있다. 둘째, 서명값의 시간을 검증해야 한다. 이는 오래된 버전의 서명된 정보를 재사용하는 공격을 방지하기 위함이다. 이는 과거에 서명 값을 이용하는 Replay Old Metadata를 방지하기 위함이다. 셋째, HTTPS를 사용한다. HTTP를 사용하여 중간자(Man-in-the-middle) 공격을 당하는 것을 방지하기 위함이다. 넷째, 미러 사이트를 만들어 배포 권한을 위임할 때 주의해야 한다. 이는 공격자가 가짜 미러 사이트를 통해서 저장소를 위장하는 것을 방지하기 위함이다. 구체적으로, depends on everything 공격이나 extraneous dependencies 공격 등 공격자가 저장소를 위장하면서 발생하는 공격기법을 방지할 수 있다. 다섯째, 메타데이터와 패키지 모두 서명을 한다. 기존에 한쪽에만 서명해서 발생하는 가용한 공격을 방지하기 위함이다. 여섯째, 메타데이터가 올바른지 검사한다. 이는 서명 검증뿐만 아니라 메타데이터 자체가 저장소에서 제공한 기존의 것과 같은지를 검사하라는 뜻이다. 이는 앞서 언급한 provides everything 공격, depends on everything 공격, extraneous depends 공격, 그리고 unsatisfiable dependencies 공격을 방지하기 위함이다.

본 연구는 폐쇄망이라는 특수한 환경을 고려해야 하므로 위 6가지 권고사항을 모두 따를 필요는 없을 것이다. 그렇다고 무시해서는 안 되므로 다음 내용에서 고려사항을 제시할 것이다[9].

3.1 새로운 패키지 관리 도구 개발

먼저 기존의 RPM이나 YUM등의 패키지 설치 매니저에서

해당 패키지를 검증하는 과정은 필수적이지 않다. 즉 패키지 설치 과정에서 한 서명 검증 기능을 필수적으로 요구하며, 패키지 자체가 악의적인 동작을 하도록 만들어진 패키지인 경우, 이를 식별하는 과정이 필요하다. 이는 심각한 보안 취약점으로 이어질 수 있으므로 보안이 일반적인 상황보다 중요시되는 폐쇄망이라는 환경에서 특수하게 고려되어야 한다.

Justin Cappos 등[10]의 연구 논문에 의하면 자체적으로 위에서 언급된 취약점들을 보완한 Stork라는 패키지 매니저를 개발하였다.

해당 패키지 매니저에서는 신뢰 가능한 패키지 형식을 새로 설계하고, 설치 과정에서 발생할 수 있는 취약점들을 보완했다. 이처럼 필요한 환경에 적합하게 기능들을 추가하고, 취약한 부분을 보완할 방법이 고려된 신규 패키지 매니저를 도입하는 것이 효과적이다.

3.2 물리적 저장매체 활용

앞서 언급한 폐쇄망은 외부 네트워크와 연결되어 있으면 폐쇄망이라는 환경적 특성을 유지한다고 볼 수 없다. 일반적인 패키지 매니저는 필요로 하는 패키지가 저장된 외부 저장소와 통신하여 패키지를 다운로드하고, 패키지 제작사에서는 저장소에 대해서 패키지를 업데이트하는 방식을 채택한다. 폐쇄망에서는 외부 저장소와 연결되어 있을 수 없기에, 자체적인 패키지 서버를 구축하여야 한다. 패키지 서버에 필요한 패키지를 업로드 하기 위해서는 물리적 저장매체를 활용하여 외부에서 패키지를 가져와야 한다. 단순히 USB, CD와 같은 저장매체를 활용해서 가져오는 것만으로는 보안 측면인 요소를 모두 고려한다고 보기는 어려우므로 절차적인 요소들이 필요하다.

본 논문에서 제시하는 절차를 순서도로 정리해놓은 것이 Fig. 5와 같다. 먼저 클라이언트의 다운로드 및 업데이트 요청을 받게 된다. 이때 클라이언트는 해당 요청에 대한 합당한 사유가 있어야 하므로, 해당 패키지가 필요한 것인지 그 필요성에 대한 검토가 이루어져야 한다. 폐쇄망 내부에서 해당 과정이 이루어진 다음 폐쇄망 외부로 요청을 전달한다. 폐쇄망 외부에서는 요청받은 패키지에 대해서 취약점이 존재하는지 확인하고, 악의적인 기능이 들어있는지 확인하기 위해 백신 프로그램의 검사를 통과하는지 확인한다. 이후 해당 패키지에 대한 특별 사항이 없는 경우 패키지를 다운로드한다. 폐쇄망 외부에서 폐쇄망 내부로 전달하기 위해 물리 저장장치에 패키지들을 저장한다. 저장된 패키지는 전달 과정에서 Read-Only 상태를 유지해야 하며 변형될 수 없도록 그 무결성을 유지해야 한다. 해당 과정이 필수적이기 때문에 일반적인 메모리 장비를 사용하는 것보다는 Read-Only 상태를 지원하는 특수한 장비를 사용하는 것을 권고한다.

이 과정에서 인터넷 환경을 이용하지 않고 물리적인 환경을 필수적으로 요구하므로 클라이언트의 요청을 개별적으로 처리

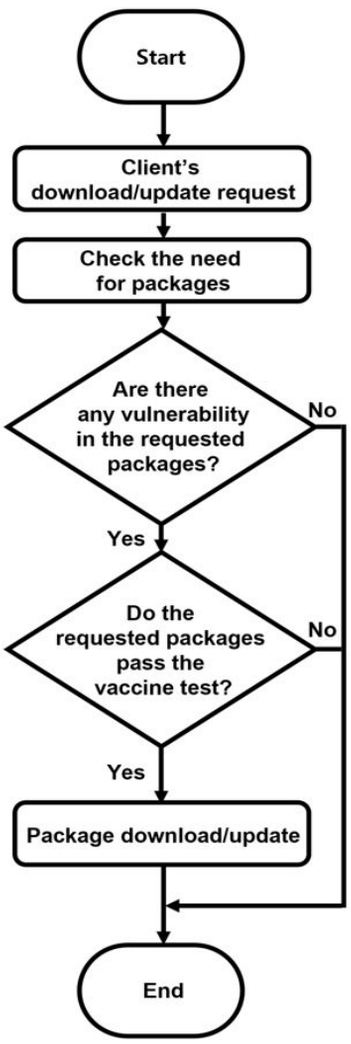

Fig. 5. Package Download Process

하는 것보다는 일정 기간마다 일괄적으로 처리하는 것이 효과적이나, 새로운 취약점이 발견되는 특수한 상황 속에서는 긴급하게 패키지를 삭제하거나, 업데이트하는 과정이 필요하다.

3.3 로컬 백업 저장소 활용

보안의 위협으로부터 시스템을 안전하게 보호할 수 있는 방법은 이론상 존재하지 않지만, 보안의 위협이 발생하고 난 뒤에 대처하는 부분에서는 수많은 방안이 존재한다. 그중에 가장 효율적인 방법 중 한 가지는 꾸준한 백업을 진행하는 것이다. 이 방법은 폐쇄망 내의 패키지를 관리할 때도 예외 없이 대처 방안으로서 채택되어야 할 만큼 여전히 좋은 방안으로 인정받고 있다.

폐쇄망의 클라이언트 로컬에서 사용하는 패키지들을 일정한 기간을 두고 createrepo 등의 서비스를 이용해 로컬 백업 저장소를 구성해놓을 경우, 따로 폐쇄망 서버에 접근하지 않고도 일정 기간의 패키지들을 모두 백업할 수 있다.

Table 1. Permissions

	Server Batch Permission	Server Management Permission	Package Management Permission
Batch Request	O	X	X
Package Management (on the server)	X	X	O
Other server management	X	O	X

위에 설명이 되어있는 것처럼 안정적인 방식으로 패키지들을 관리한다고 하더라도 패키지를 관리하다 보면, 어쩔 수 없이 패키지의 과거 버전이 필요하게 될 경우가 종종 발생할 수 있다. 대부분은 과거 버전을 단순히 패키지 서버를 통해 다운그레이드를 진행하면 그만이지만, 안전한 방법론이 해커에 의해 쉽게 우회될 수 있다.

현재 우리 사회에서 가장 영향력이 있다고 이야기할 수 있는 구글, 삼성, 애플 등등의 대형 IT 기업들에는 자체적인 보안 팀이 존재하며, 최고의 인재들과 함께 항상 자사 서비스에 대한 보안에 큰 힘을 쏟고 있다. 하지만 해커들은 매 순간 수많은 서비스에서 꾸준히 취약점을 발굴해낸다. 즉, 아무리 보안상으로 최선의 노력을 기울이더라도, 해커들은 매번 보안을 무력화시키는 방법을 발견해왔고, 앞으로도 그럴 가능성이 농후하다.

이 사항은 모든 서비스에 똑같이 해당하며, 당연히 폐쇄망 내의 패키지 서버와 클라이언트에서도 예외는 아니다. 아무리 위에서 제안한 방법처럼 보안상 안전한 방법론을 사용해 폐쇄망 내에서 패키지를 설치 및 활용했다 한들, 해커들로 인해 보안상의 안정성이 훼손당할 가능성은 충분히 존재한다. 그러므로 결국에는 폐쇄망 내에 악성 패키지가 설치될 가능성, 혹은 취약한 패키지가 설치될 가능성을 완전히 배제할 수 없다.

만일 위의 방법론에서 제시한 순서도를 모두 거쳐 서버와 클라이언트에 설치된 패키지가 취약한, 혹은 악성 패키지였다고 가정해보자. 그럴 경우, 해당 서버는 물론이고, 패키지를 현재 사용하고 있는 클라이언트는 최대한 빠르게 해당 패키지를 취약하지 않았던, 또는 악성 패키지가 아니었던 버전으로 되돌릴 수 있어야 한다. 하지만 현재 폐쇄망 내의 패키지가 취약하거나 악성 패키지인 경우, 패키지 서버에 대한 보안상의 신뢰도를 잃은 상태와 마찬가지이기에 서버로부터 이전 및 다음 버전의 패키지를 설치하는 것은 매우 위험한 행위다.

따라서 패키지 업그레이드 혹은 다운그레이드를 시도는 해야 하지만, 폐쇄망 내의 패키지 서버에는 접근하지 않아야 한다는 모순이 발생하게 되는데, 이때 이전에 구성해두었던 로컬 백업 저장소를 활용하는 것이다.

이전에 꾸준히 패키지들을 로컬 백업 저장소에 백업을 해놓았다면, 현재 사용 중인 위험 패키지들을 모두 삭제하고 이전 버전의 패키지로 다운그레이드를 진행할 수 있게 되고, 클라이언트는 현재 악성 패키지에 의해 위협당하던 보안성을 되찾을 수 있게 된다.

3.4 패키지 업데이트 및 다운그레이드 일괄 처리

패키지로 인해 클라이언트가 위협을 받는 경우 중의 가장 최악의 경우는 이미 클라이언트의 대부분이 악성 패키지를 설치 및 사용 중인 경우다. 해당 경우에는 일괄적으로 모든 클라이언트의 패키지를 업데이트 혹은 다운그레이드 처리를 해줘야 하는데, 이를 일일이 클라이언트 모두에서 작업할 수는 없는 노릇이다. 따라서, 해당 경우에는 재빠르게 일괄 처리를 할 방안이 필요하다.

바로 위에서 설명했듯이 위협적인 패키지가 클라이언트들에게 배포되었다는 사실을 알게 되었을 때, 클라이언트 단에서 일일이 패키지 업데이트 및 다운그레이드 처리를 하는 것은 불가능하므로, 이 작업을 할 수 있는 매체가 필요한데, 그것이 가능한 매체가 바로 패키지 서버다.

서버 단에서는 해당하는 패키지를 모든 클라이언트에게 일괄적으로 업데이트 및 다운그레이드 요청을 보내서 클라이언트에게 강제적으로 위협적인 패키지들을 제거 및 업데이트, 다운그레이드를 진행하는 것이 기술적으로 가능하다. 따라서 클라이언트의 위험한 패키지에 대해서 서버 단에서 직접 패키지 일괄 처리를 진행하여 클라이언트의 위협을 제거할 수 있다. 하지만, 역으로 생각해보면 이 기술 또한 클라이언트를 향한 보안 위협의 요소로 자리할 수 있다.

현재 패키지 서버 또한 클라이언트와 다르지 않게 매우 큰 위협을 받는 상황이라면, 즉 패키지 서버가 이미 취약성에 의해 원격으로 해킹당하였을 경우라면 패키지 서버에서 클라이언트를 향해 함부로 업데이트, 다운그레이드 요청을 보내는 것은 신뢰할만한 요청이 아니다. 이 경우 해커에 의한 서버의 패키지 일괄 처리 요청으로 인해 클라이언트들이 위험 패키지를 무더기로 설치할 가능성도 존재한다. 따라서 Table 1에 나타난 바와 같이 해당 요청을 사용할 수 있는 권한과 일반 패키지 서버 관리 권한을 분리하여 패키지 일괄 처리 기능을 활용하여야 한다.

권한을 분리하여 패키지의 일괄 처리 기능을 사용하면, 권한을 조절할 수 있는 폐쇄망 내부의 관리자 외에는 일괄 처리 기능을 요청할 수 없고, 안전하게 패키지에 대한 일괄 작업을 거쳐서 클라이언트들의 위험 패키지에 대한 관리를 진행할 수 있다.

4. 결 론

본 논문에서는 최근 중요한 논란이 되었던 패키지 관리 서비스에 대한 분석과 현존하는 취약점에 대해서 알아보았다. 기본적으로 가장 큰 문제점은 패키지 관리 서비스에서 패키지를 설치할 때 별다른 검증 과정과 패키지를 검사하는 과정이 존재하지 않았다는 것이다. 추가로 각종 네트워크 공격에도 취약한 부분을 찾아볼 수 있었다.

제일 많이 사용되는 인터넷이 연결된 환경에 그치지 않고, 보안 기능이 더욱 강화되어야 하는 폐쇄망 환경에 대해서 패키지를 관리하는데 현존하는 취약점과, 이를 개선할 방법을 제시하였다. 본 연구의 결과물로 폐쇄망 환경에서 패키지 관리 서비스를 이용할 때 고려해야 할 사항을 기존의 연구에 덧붙여 3가지의 방법을 제안하였다. 폐쇄망의 지극히 제한적인 환경에서 이루어진 네트워크망이라는 점에서 많은 한계가 존재하지만 이러한 환경 때문에 추가로 연구할 요소들이 많다고 볼 수 있다. 따라서 향후 특정 환경을 제시한 다음 이에 맞게 패키지 관리 서비스를 자체 제작해보거나, 패키지 내부를 검증하는 과정에서 악성코드를 탐지하는 기법에 관해 연구하며 제도적인 차원에서의 권고사항보다 기술적인 연구로 나아갈 계획이다.

References

[1] Format of the RPM File - RPM File Format [Internet], http://ftp.rpm.org/max-rpm/s1-rpm-file-format-rpm-file-format.html.

[2] RPM Package File Structure [Internet], https://docs.fedoraproject.org/ro/Fedora_Draft_Documentation/0.1/html/RPM_Guide/ch-package-structure.html.

[3] RPM Package Manager [Internet], http://www.rpm.org/.

[4] A. Athalye, R. Hristov, T. Nguyen, and Q. Nguyen, "Package Manager Security," Tech. Rep. [Internet], https://pdfs.semanticscholar.org/d398/d240e916079e418b77ebb4b3730d7e959b15. pdf. 2020.

[5] Securing RPM Signing Keys [Internet], https://access.redhat.com/blogs/766093/posts/3373211.

[6] Man dpkg-sig [Internet], http://pwet.fr/man/linux/commandes/dpkg_sig.

[7] J. Cappos, J. Samuel, S. Baker, and J. H. Hartman, "A look in the mirror: Attacks on package managers," *Proceedings of the 15th ACM conference on Computer and communications security*, Oct. 2008.

[8] J. Cappos, J. Samuel, S. Baker, and J. H. Hartman, "Package management security," University of Arizona Technical Report, 2008.

[9] Y. H. Jo and E. K. Lee, "Design of information security management for industrial control system," *Proceedings of the Korean Society of Computer Information Conference*, Korean Society of Computer Information, Jan. 2016.

[10] J. Cappos et al., "Stork: Package management for distributed VM environments," *Proceedings of the 21st Large Installation System Administration Conference (LISA '07)*, Nov. 2007.

안 건 희

https://orcid.org/0000-0003-2906-1651
e-mail : ipwn@korea.ac.kr
2020년 ~ 현 재 고려대학교 사이버국방학과 학사과정
관심분야 : IoT hacking, Kernel hacking

안 상 혁

https://orcid.org/0000-0003-2303-3633
e-mail : starson1@korea.ac.kr
2019년 ~ 현 재 고려대학교 사이버국방학과 학사과정
관심분야 : Digital Forensics, Reverse Engineering

임 동 균

https://orcid.org/0000-0003-1914-5640
e-mail : sts08015@korea.ac.kr
2020년 ~ 현 재 고려대학교 사이버국방학과 학사과정
관심분야 : Operating systems

정 수 환

https://orcid.org/0000-0003-4131-8050
e-mail : pk2861@korea.ac.kr
2019년 ~ 현 재 고려대학교 사이버국방학과 학사과정
관심분야 : System Hacking, Operating systems, Cryptography

김 재 우

https://orcid.org/0000-0002-2012-812X
e-mail : jaewooz@gmail.com
1996년 한양대학교 전자통신공학과(학사)
2010년 한양대학교 전기전자컴퓨터공학과 (석사)
2010년 ~ 현 재 한화시스템 수석연구원
관심분야 : Tactical Radio Comm, SOC Implementation or FPGA System Design H/W and S/W Partitioning and Co-verification. 5G NR Systems.

신 영 주

https://orcid.org/0000-0003-4831-7392
e-mail : syoungjoo@korea.ac.kr
2006년 고려대학교 컴퓨터학과(학사)
2008년 KAIST 전산학과(석사)
2014년 KAIST 전산학과(박사)
2008년 ~ 2017년 국가보안기술연구소 선임연구원
2017년 ~ 2020년 광운대학교 컴퓨터정보공학부 조교수
2020년 ~ 현 재 고려대학교 정보보호대학원 조교수
관심분야 : System security, Network security, Cloud computing security