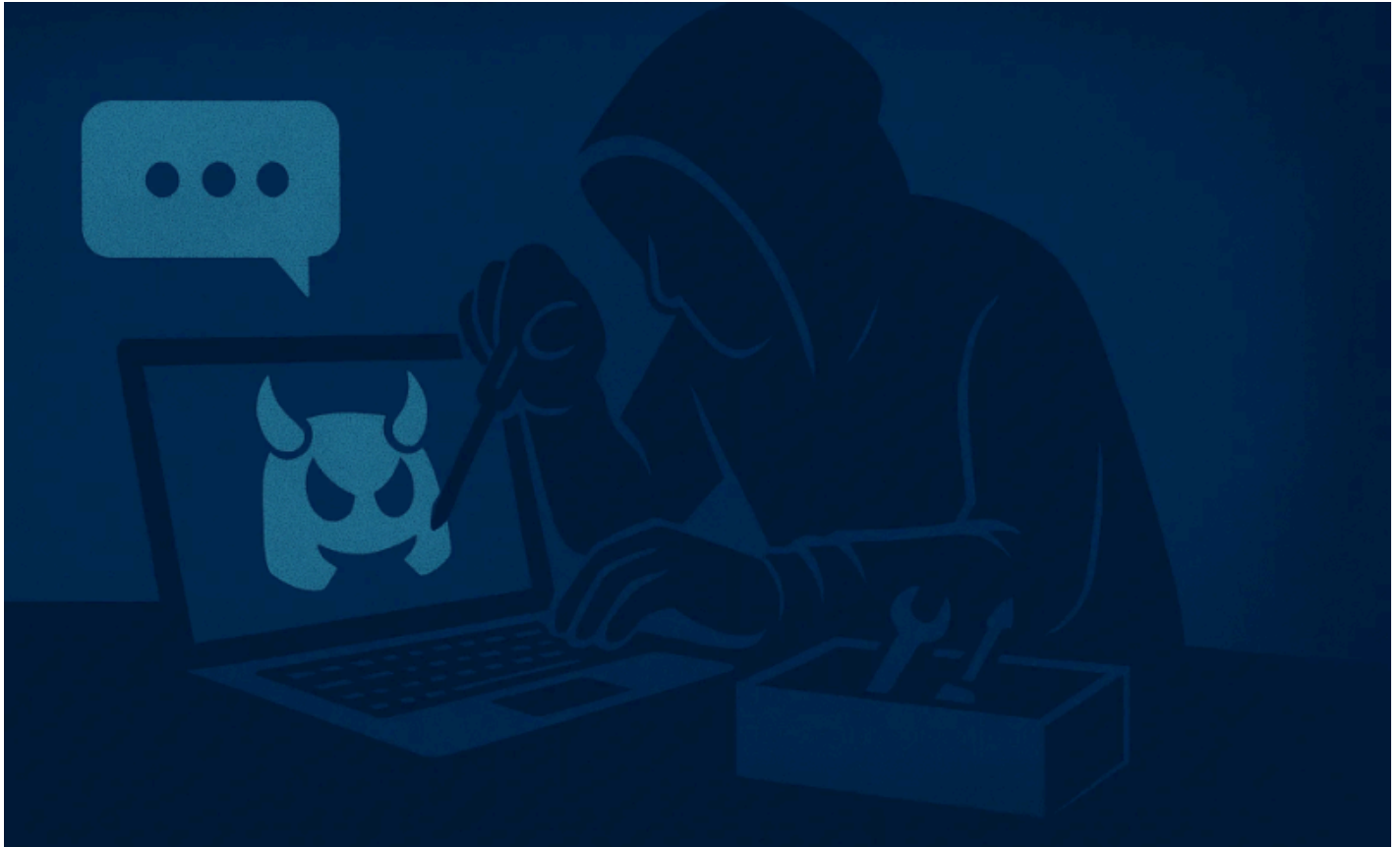


UNC5174 그룹의 Discord Bot 백도어 악성코드

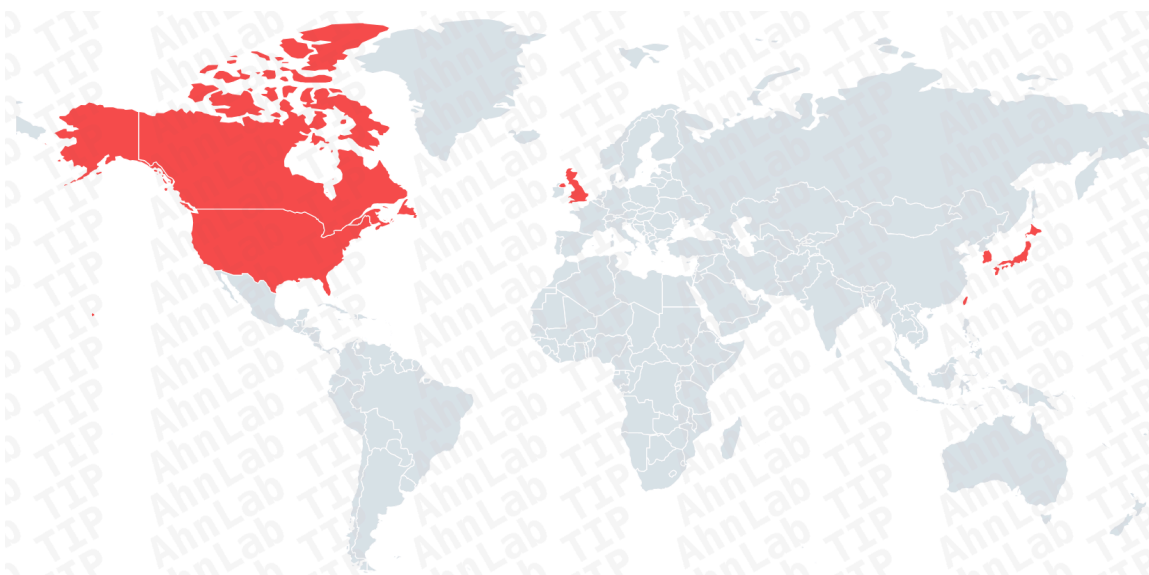
A asec.ahnlab.com/ko/91228

ATCP

November 26, 2025



ASEC(AhnLab Security Intelligence Center)은 최근 Discord API를 이용해 C2(Command and Control) 체계를 구축한 백도어 악성코드가 사용된 공격 사례(UNC5174 그룹[1])를 발견했다. UNC5174 공격그룹은 초기 침투 이후 장기적인 통제권을 유지하기 위해 여러 유형의 백도어를 순차적으로 배치하는 운영 방식을 사용한다. 실제 침해사고 분석 과정에서, 공격자는 먼저 vshell 등 기존에 사용하던 백도어로 대상 시스템에 접근한 뒤, 일정 시간이 지나면 탐지 부담이 더 적은 새로운 형태의 백도어를 추가로 설치하는 전술을 반복적으로 수행한 정황이 확인되었다.

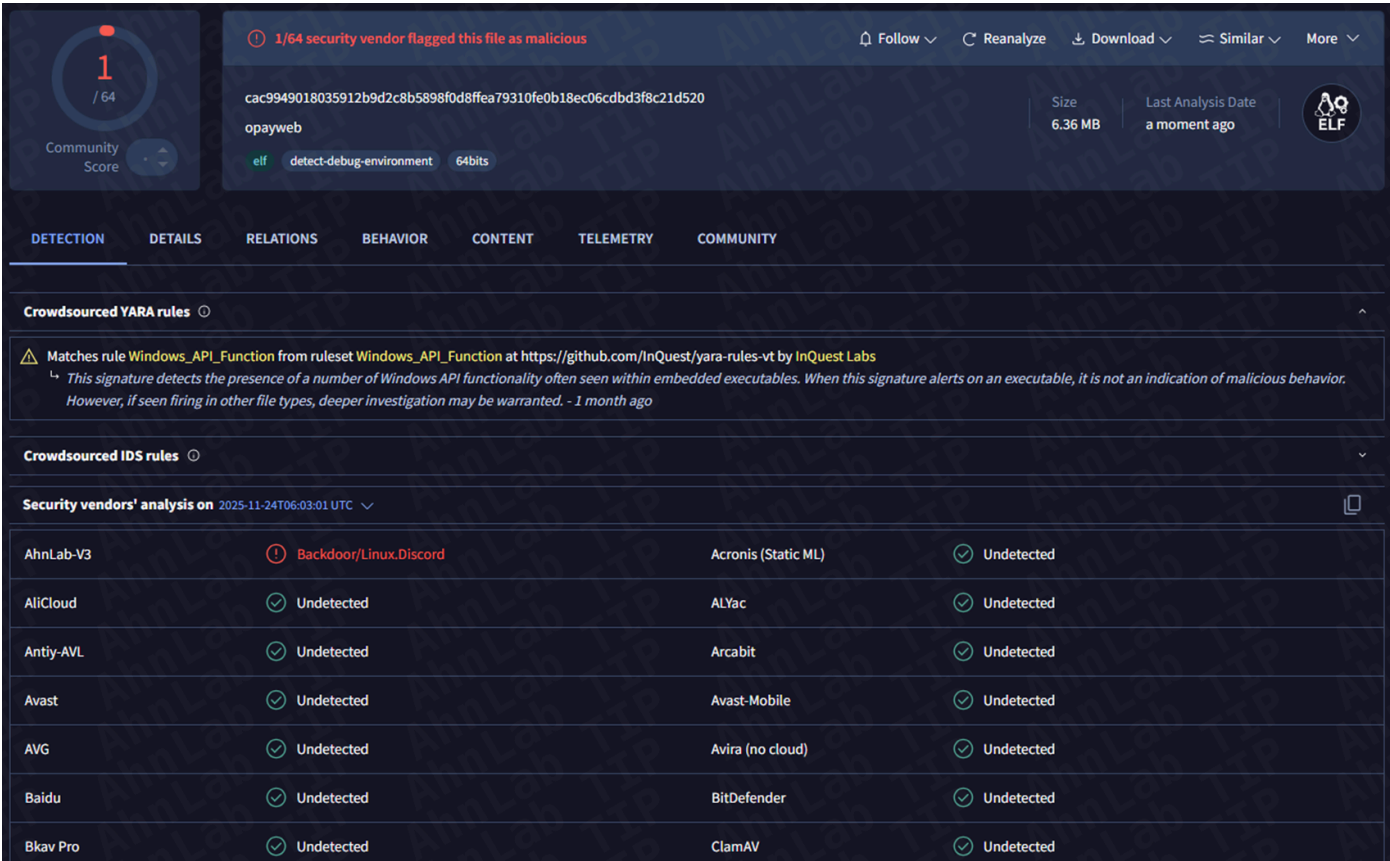


[그림 1] UNC5174 그룹의 공격에 의한 침해 지역 (출처: ATIP)

이 전환 과정에서 Discord Bot 기반의 백도어 악성코드가 추가로 배포된 흔적이 발견되었으며, 해당 악성코드는 공격자의 활동 중 수집된 디스크 아티팩트 및 네트워크 로그에서 식별되었다. 특히, 공격자는 별도의 C2 인프라를 구축하지 않고 Discord API를 명령·제어 채널로 활용함으로써 정상적인 사용자 트래픽과 유사한 패턴을 만들어 내고, 기존 보안 정책을 우회하려는 의도를 보였다. 이러한 운영 방식은 기존 C2 서버가 탐지되거나 차단될 위험이 생겼을 때 즉시 대체 채널을 확보하려는 UNC5174 그룹의 지속성 전략과도 맞닿아 있다.

Discord는 음성, 영상, 텍스트 기반의 커뮤니티 플랫폼으로, 게이머 중심에서 시작했지만 현재는 다양한 관심사와 그룹에서 널리 사용되고 있다. 사용자는 특정 주제로 서버를 만들고, 그 안에서 텍스트 채널, 음성 채널, 스레드 등을 생성해 커뮤니케이션할 수 있다. 또한 Bot을 추가해 자동화 기능을 구현할 수 있으며, Discord API(REST + Gateway)를 활용해 다양한 기능을 개발할 수 있다.

공격자는 이러한 Discord의 기능을 악용하여 Backdoor 유형의 악성코드를 제작했다. Github에 오픈소스로 공개된 discordgo 모듈을 사용한 것으로 확인되며 2025년 11월 24일 현재 이 글을 포스팅 하는 시점에 Virustotal Score는 1/64 점으로, AhnLab 제품군 외에는 탐지되지 않는 파일이다.



[그림 2] VirusTotal Score

Discordgo는 Go 언어(Golang)로 작성된 오픈소스 라이브러리로, Discord API와 상호작용하는 애플리케이션이나 봇을 개발할 때 사용된다. 이 라이브러리는 REST API와 WebSocket Gateway를 모두 지원하여 메시지 전송, 채널 관리, 이벤트 처리 등 다양한 기능을 구현할 수 있다. 또한 음성 채널 연결과 오디오 스트리밍 기능도 제공해 실시간 커뮤니케이션 봇 개발에 적합합니다. Discordgo를 통해 인증 토큰 기반으로 봇을 손쉽게 생성하고, 명령어 처리나 자동화 기능을 구현할 수 있습니다. 오픈소스 특성상 누구나 접근 가능하지만, 공격자는 이러한 점을 악용하여 악성코드를 제작하는데 편리하게 활용했다.

Discordgo Github repo : <https://github.com/bwmarrin/discordgo>

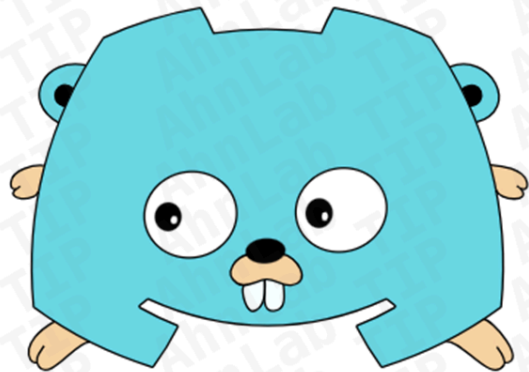
DiscordGo

reference go report A+ CI passing Discord Gophers #discordgo Discord API #go discordgo

DiscordGo is a [Go](#) package that provides low level bindings to the [Discord](#) chat client API. DiscordGo has nearly complete support for all of the Discord API endpoints, websocket interface, and voice interface.

If you would like to help the DiscordGo package please use [this link](#) to add the official DiscordGo test bot **dgo** to your server. This provides indispensable help to this project.

- See [dgVoice](#) package for an example of additional voice helper functions and features for DiscordGo.
- See [dca](#) for an **experimental** stand alone tool that wraps `ffmpeg` to create opus encoded audio appropriate for use with Discord (and DiscordGo).



[그림 3] Discordgo

아래는 오픈소스 Discordgo 라이브러리를 악용하여 제작된 악성코드를 상세하게 분석한다. 우선 Discord Bot을 구현하기 위한 첫 번째 단계로, Token 값과 서버ID 값을 활용하여 공격자의 서버에 연결을 시도한다. 해당 Token값과 서버ID 값은 파일 내부에 암호화 되어있으며 런타임에 base64, AES Decrypt 방식을 이용하여 복호화된다. 이후 복호화된 값을 가지고 공격자의 서버에 연결을 시도한다. 보통의 경우 Discord API를 사용하는 것이 일반적이지만, Discordgo 라이브러리를 활용하면 `discordgo.New()` 함수를 호출함으로써 편리하게 연결을 생성할 수 있다.

```
mov     rdi, cs:off_A5C4E0
mov     rsi, cs:qword_A5C4E8
xor     eax, eax
lea     rbx, aBashbot+4          ; "Bot "
mov     ecx, 4
call    runtime_concatstring2    ; Bot+Token
call    github_com_bwmarrin_discordgo_New
```

[그림 4] discord.New() 호출

Discord는 단순한 채팅 플랫폼을 넘어 다양한 봇(Bot) 기능을 제공한다. 주목할 점은 봇이 서버에서 메시지를 수신하면 특정 이벤트를 발생시키는데, 그중 하나가 `MessageCreate` 이벤트다. 공격자는 바로 이 이벤트를 악용해 백도어 기능을 구현했다.

Discordgo 라이브러리에서는 `AddHandler()` 함수를 통해 이벤트 발생 시 실행할 콜백 함수를 등록할 수 있으며, 공격자는 이 기능을 이용해 다음과 같은 흐름으로 자신의 명령을 Bot이 처리하도록 구성했다.

1. **명령 수신**: 공격자가 Discord 채널에 명령을 입력한다.
2. **이벤트 발생**: Bot이 메시지를 받으면 `MessageCreate` 이벤트가 트리거된다.
3. **콜백 실행**: 등록된 콜백 함수가 실행되면서 명령을 해석하고 수행한다.

```

mov    [rsp+0C0h+var_90], rax
lea    rbx, RTYPE_func_ptr_discordgo_Session_comma___ptr_discordgo_MessageCreate
lea    rcx, func_ptr_cb_MessageCreate
call   github_com_bwmarrin_discordgo_ptr_Session_AddHandler
mov    rax, [rsp+0C0h+var_90]
lea    rbx, RTYPE_func_ptr_discordgo_Session_comma___ptr_discordgo_Ready
lea    rcx, func_ptr_cb_Ready
call   github_com_bwmarrin_discordgo_ptr_Session_AddHandler
mov    rax, [rsp+0C0h+var_90] ; _ptr_discordgo_Session
call   github_com_bwmarrin_discordgo_ptr_Session_Open

```

[그림 5] AddHandler() 호출하여 MessageCreate 이벤트를 처리할 함수 등록

이렇게 제작된 백도어 악성코드는 아래와 같은 기능을 가진다.

- Command 실행
- 파일 업로드
- 파일 다운로드
- 시스템 정보 수집

공격자로부터 명령을 실행하라는 메시지를 수신하면 MessageCreate 이벤트를 통해 이를 수신하고, 등록된 핸들러에서 명령을 처리한다. 이후 `bash -c` 뒤에 payload를 붙여 실행 문자열을 완성해 시스템 명령을 수행한다. 실행 결과는 `/tmp/message.txt` 같은 임시 파일에 저장한 뒤, 그 내용을 다시 Discord로 전송한다. 실행 결과를 공격자에게 전송한 이후 파일을 삭제해 흔적을 최소화하는 방식이다.

```

v4[1] = 2;
v4[0] = &byte_7A7AC5;           // "-c"
v4[3] = a2;
v4[2] = a1;
v2 = os_exec_Command("bashBot ", 4u, v4, 2); // bash -c "{a1} {a2}"
v5 = os_exec_ptr_Cmd_CombinedOutput(v2);

```

[그림 6] bash로 공격자의 명령 실행

```

v14 = feat_runCommand(v11, v10);
v128 = runtime_stringtoslicebyte(v82, v14, v15);
os_WriteFile("/tmp/message.txt", 16, v128._r0, v128._r1, v128._r2, 420);
r0 = os_OpenFile("/tmp/message.txt", 16, 0, 0)._r0;
v123[0] = main_v23_deferwrap1;
v123[1] = r0;
v125 = v123;
v16 = *v127;
v121 =>(*v127 + 16);
v101 =(*v16 + 24);
p_discordgo_MessageSend = runtime_newobject(&RTYPE_discordgo_MessageSend);
p_discordgo_File = runtime_newobject(&RTYPE_discordgo_File);
p_discordgo_File->Name.len = 11;
p_discordgo_File->Name.ptr = "message.txt";
p_discordgo_File->Reader.tab = off_837980;

```

[그림 7] 명령 실행 결과를 message.txt로 저장

아래와 같은 시스템 정보를 Victim으로부터 수집할 수 있다. 해당 정보를 수집하는 과정에서는 또다른 Golang 오픈소스가 활용된다.

```
v24 = fmt.Sprintf(
    "Hostname: %s\n"
    "CPU Cores: %d\n"
    "CPU Model: %s\n"
    "Total Memory: %d MB\n"
    "Used Memory: %d MB\n"
    "Total Disk Space: %d GB\n"
    "Used Disk Space: %d GB\n"
    "OS Platform: %s\n"
    "OS Version: %s\n"
    "OS Architecture: %s\n"
    "Local IP: %s\n"
    "Public IP: %s\n"
    "Current Directory: %s",
```

[그림 8] 시스템에서 탈취할 수 있는 정보

공격자는 대부분 오픈소스 라이브러리를 활용해 구현하기 때문에 직접 작성한 코드가 100줄 미만일 것으로 추정된다. 이렇게 짧고 간단한 방법으로 탐지하기 어려운 Backdoor 악성코드가 충분히 동작한다는 점에서 위험은 더욱 현실적이다. 이처럼, 최근 공격자들은 오픈소스를 기반으로 악성코드를 제작해 정상 파일처럼 위장하는 기법을 적극 활용하고 있다. 특히 Discord를 명령·제어(C2) 채널로 사용하면 일반 트래픽에 섞여 탐지가 훨씬 어려워진다. 이러한 방식은 별도의 C2 인프라를 구축할 필요가 없고, 인증·통신이 모두 Discord 플랫폼을 통해 이루어지기 때문에 보안 솔루션을 우회하기 쉽다.

index	Bot Token	Server ID
1	MTM5MzE4Mzg4NjYwMjkzMjI2NA.G6ooB1.zBdzA0XNv0k219EbJsXgxuR7N_X_H3beHgz_iM	1393184594974474344
2	MTQxMzQyNzA1Nzc2MDY2NTcyMQ.Golp3b.yc5Z1_qjtKJxwsznCRc5k2KCB2_BARLMUGqqNs	1413435292970647596

[표 1] 공격에 사용된 Bot Token과 Server ID

결과적으로 공격자는 간단한 메시지 이벤트를 통해 원격 명령 실행, 데이터 유출 등 다양한 악성 행위를 수행할 수 있다. 오픈소스와 협업 플랫폼을 악용하는 공격이 늘어나는 만큼, 사용자는 이러한 사실을 사전에 인지하고 출처가 불분명한 파일에 대해 각별히 주의해야 한다.

MD5
c193742412e98f1d46953f1ee73841b9

AhnLab TIP를 구독하시면 연관 IOC 및 상세 분석 정보를 추가적으로 확인하실 수 있습니다. 자세한 내용은 아래 배너를 클릭하여 확인해보세요.

