

A Russian-Speaking Campaign Hiding C2 Behind GitHub Redirects

 vmray.com/execution-level-analysis-of-a-russian-speaking-multi-operator-intrusion-campaign-operation-standoff

Fatih Cam

July 20, 2026



Executive Summary

VMRay Labs identified a previously unattributed, actively operated cybercriminal campaign conducted by a Russian-speaking threat group, tracked in this report as Operation STANDOFF. What began as a single malware detection, surfaced through a UniqueSignal finding, was reconstructed, through execution-level analysis and infrastructure pivoting, into a multi-layered operation that unifies commodity-malware distribution, a proxy-botnet, hands-on-keyboard network intrusion, and a large-scale, AI-driven influence and engagement-manipulation capability. All of these components are hosted on shared infrastructure at the Russian provider TimeWeb Ltd. (AS9123) and concealed behind traffic engineered to mimic the GitHub domain.

The operation is composed of several interlocking layers:

- Initial access and monetisation: A pay-per-install (PPI) loader installs a bundle of commodity malware -including Raccoon Stealer, RedLine, Amadey, SmokeLoader, Socelars, Glupteba and the XMRig cryptocurrency miner- while disabling security controls and establishing rootkit-grade persistence. Compromised hosts are simultaneously enrolled into a proxy-botnet that turns victims into traffic-relay nodes.
- Concealed command-and-control: The campaign-specific C2 servers (212.193.30[.]29 and 212.193.30[.]45) are part of a larger cluster of at least 44 servers hosted on the Russian provider TimeWeb. Each server in this cluster is configured to respond to any unsolicited HTTP request with a 301 redirect to [https://github\[.\]com](https://github[.]com). This means that a security analyst or automated scanner probing these IPs will receive a response pointing to GitHub, making the servers appear unrelated to malicious activity and their network traffic blend in with legitimate GitHub communications. One node serves the operator domain [russianhckers\[.\]online](https://russianhckers[.]online).
- That domain hosts "STANDOFF COORD", a custom web console built for teams of operators to coordinate targeted intrusions into corporate Active Directory environments. It stores stolen credentials (NTLM hashes, Kerberos tickets, session cookies, and private keys), tracks compromised hosts organised by their position within the target network (internal, DMZ, or external-facing, and provides shared playbooks and points-based scoreboard to coordinate and measure each operator's activity.

- Influence and mass outreach: A second server (217.198.13[.]211) hosts two more capabilities. The first is a large-scale Telegram account farm. The second is an AI-driven automation platform that operates across three channels: e-mail, Telegram and WhatsApp. The public face of this server is a Russian gaming portal for Standoff 2 and PUBG Mobile, called “Мобильная Арена” (Mobile Arena). It sits at the top of the funnel, drawing in an audience that the rest of the operation then works to exploit.

Key Judgments:

- **VMRay Labs assesses with high confidence that Operation STANDOFF is run by a Russian-speaking group: its operator tooling is written in Russian, scheduled in Moscow time, and self-branded under the “GG Influence” / “ggstandoff” identity.**
- The operation is materially more than a botnet. It couples automated, scaled cybercrime with hands-on-keyboard, targeted intrusion and a coordinated influence capability, all on the same infrastructure and built by a common development team.
- The influence apparatus is assessed with high confidence to be a large-scale, AI-assisted platform that uses networks of fake Telegram accounts and AI-generated personas to artificially boost the visibility of content, push commercial promotions, and drive traffic to gambling and fraud-adjacent services, primarily within the Russian-speaking mobile-gaming ecosystem. The mobile-gaming audience attracted by this content is assessed to represent the primary victim pool for the operation’s malware distribution.
- A significant portion of the infrastructure, including several actor-controlled domains, remained undetected by security vendors at the time of writing of this report and is assessed to be actively maintained.
- Attribution has been carried beyond the group level. VMRay Labs has linked the operation to a specific individual; that identity is withheld from this report since it falls outside its scope.

1. Introduction

Modern malware campaigns are increasingly built around layered execution chains, staged infrastructure, and traffic patterns designed to appear ordinary when observed in isolation. For this reason, campaign reconstruction requires more than static indicators or isolated network artifacts. It requires execution-level visibility into how the malware starts, what processes it creates, which APIs it calls, what files it touches, how it communicates and how its infrastructure responds during runtime.

This report analyzes a Russian-speaking cybercriminal operation that combines two layers on a single, shared infrastructure. The first is a mass-access layer, a pay-per-install (PPI) loader that distributes Racoon Stealer and other commodity malware, together with a proxy-oriented botnet that enrolls compromised hosts as relay nodes. The second is a targeted operations layer: a bespoke, multi-operator command-and-control console through which human operators conduct hands-on-keyboard intrusions of selected victims, with the same infrastructure serving both. The investigation originates from a [UniqueSignal](#) -a cyber threat intelligence platform that turns VMRay’s detailed and noise-free analysis results into actionable intelligence-finding, a behaviour-level detection derived not from a static indicator match but from hypervisor-level observation of the sample’s actual execution: its process behaviour, API traces, memory and file activity, outbound traffic, protocol-level interactions and retrieved server-side content. It was this behavioural visibility, rather than any reputation- or signature-based indicator, that first surfaced the activity and made it possible to connect events that were unremarkable when viewed in isolation. These observations are then correlated with external infrastructure analysis to reconstruct the campaign from initial execution through staging, communication, and operator-side coordination.

A key infrastructure pattern observed in this campaign is the use of proxying and redirection behavior involving the official GitHub domain. The activity does not indicate that GitHub itself is compromised or malicious. Rather, the infrastructure appears to route or proxy normal-looking traffic toward GitHub in order to blend suspicious infrastructure with a trusted domain pattern and complicate straightforward interpretation by scanners, sandboxes, and analysts. It has also been assessed that this method is employed to evade infrastructure-level threat hunting by concealing their command-and-control servers behind a trusted domain.

The malware execution chain shows staged retrieval logic, malformed browser-like User-Agents, hardcoded network resources, fallback communication attempts, and C2-related behavior consistent with botnet maintenance. In parallel, the recovered server-side console reveals a multi-operator coordination platform – with target tracking by network segment, a credential vault oriented to Active Directory, shared playbooks and an operator scoring system. That supports the assessment of an organized, human-operated intrusion capability behind the automated access layer, rather than a single isolated malware host.

This report presents the findings in an evidence-driven manner. It separates confirmed behavior from analytical assessment, avoids treating single indicators as conclusive, and uses execution-level telemetry as the primary basis for linking malware behavior, infrastructure design, and campaign intent.

2. Case Details

VMRay Labs detected several evasive, under-the-counter malware infection incidents by leveraging UniqueSignal threat feed. Upon detecting these findings, we initiated an investigation and, by combining multiple events that appeared unrelated at the indicator level but proved to be connected through execution-level intelligence, identified a campaign and the previously unattributed infrastructure of a Russian-speaking cybercriminal group. The group distributes Racoon Stealer [1] [2] and other commodity malware to gain scaled initial access, while operating a proxy-botnet and a bespoke multi-operator console through which human operators conduct targeted, hands-on-keyboard intrusions of selected victims.

2.1 Initial Sample and Behaviour

The initial sample led to the infection and investigated in this report is “setup_x86_x64_install.exe” with MD5 hash “e77221d7a4b47b9107ba1b61a551ca89”. The file is a 32-bit Windows executable, [analyzed by VMRay](#) at 2026-05-13, 03:05 (UTC).

VMRay Threat Identifiers (70 rules, 479 matches)				
Score	Category	Operation	Count	Classification
50%	YARA	Malicious content matched by YARA rules	17	Spyware, Downloader, Trojan, Rootkit, Miner, PUA
50%	Extracted Configuration	RedLine configuration was extracted	1	Spyware
50%	Extracted Configuration	Socelars configuration was extracted	1	Spyware
50%	Extracted Configuration	XMRig configuration was extracted	1	Miner
40%	Injection	Process Hollowing	1	Injector
40%	Anti Analysis	Makes indirect system call to possibly evade hooking based monitoring	2	-
40%	Execution	Abuses MSHTA to execute code	3	-
40%	Defense Evasion	Modifies Windows Defender configuration	4	-
50%	Defense Evasion	Modifies Windows Defender configuration	1	-
30%	Anti Analysis	Tries to evade debugger	1	-

Malware Configurations	
RedLine	...
Socelars	...
XMRig	...

Figure 1: VMRay Platform VTIs and extracted malware configurations

Statically, the sample is a 16.65 MB NSIS-based installer. Behaviourally, however, it does not act as a single piece of malware. Execution-level telemetry shows that it functions as a multi-payload loader, distributing and launching a bundle of distinct commodity malware families during a single infection. This is the operating model commonly referred to as a Pay-Per-Install (PPI) loader: a single delivery binary that monetises one infection by installing several unrelated payloads at once.

As shown in Figure 1, over the monitored run 152 processes were observed, the binary matched 70 VMRay Threat Identifier (VTI) rules with 479 matches, and VMRay Platform assigned the classifications Downloader, Injector, Miner, PUA, Rootkit, Spyware and Trojan. The same figure also lists the malware configurations extracted from the payloads, including those of RedLine, Socelars and XMRig. The aim of the sample is therefore not narrow data theft alone, but to convert the victim host into a fully monetised, persistently controlled asset. Figure 2 maps the techniques the VMRay Platform observed during this run onto the MITRE ATT&CK framework, grouped by tactic. The Initial Access column is empty, consistent with a post-delivery detonation in which no exploitation or intrusion vector is exercised; the recorded activity begins at the Execution tactic. Defense Evasion is by far the most heavily populated column, reflecting how much of the sample’s effort goes into evading analysis and disabling security tooling.

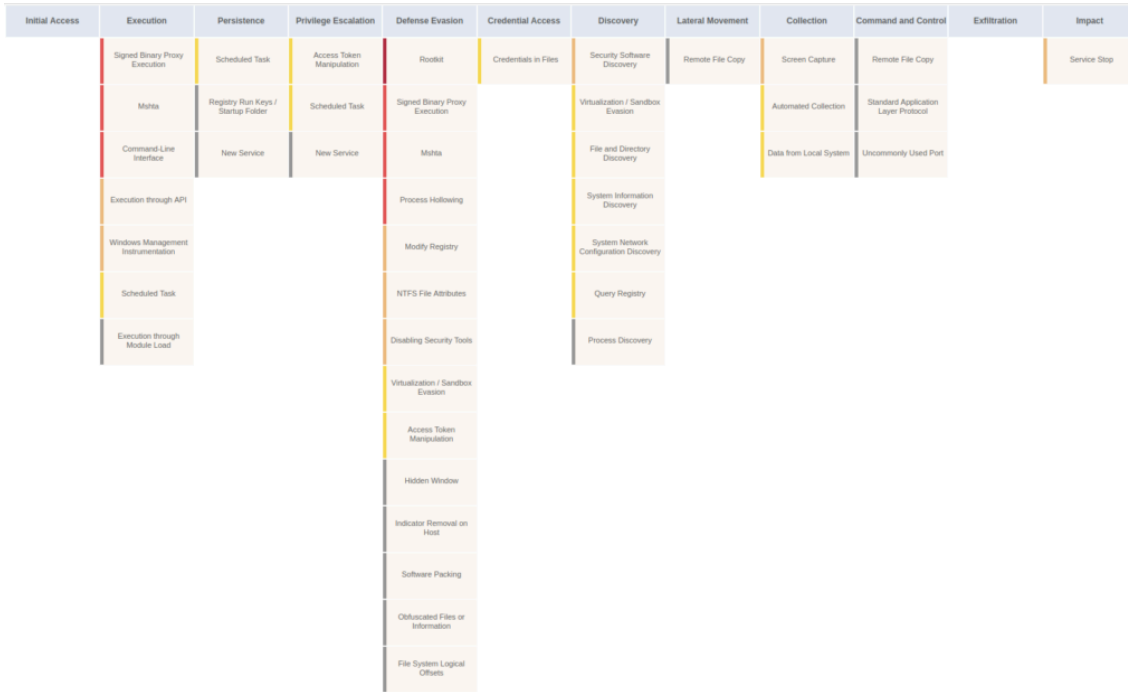


Figure 2: MITRE ATT&CK techniques observed during sample execution, as mapped by the VMRay Platform

Execution and staging: On launch, the installer unpacks and writes a large set of secondary executables (approximately 40-50 dropped PE files) into the staging directory %LOCALAPPDATA%\Temp\7zSCB82E89C\, using opaque, randomised file names of the form MONXXXXXXXXX.exe. The parent process #3 (setup_install.exe) orchestrates execution, repeatedly spawning cmd.exe to run each dropped payload in turn, frequently with hidden windows and ambiguous "/c" arguments to reduce visibility. Several payloads side-load helper libraries (libcurl.dll, libcurlpp.dll, libgcc, libstdc++, libwinpthread) and are protected by a mix of packers and protectors observed across the family – NSIS, ASProtect v1.23 RC1, ASPack v2.12, .NET Reactor, EnigmaProtector, the CryptOne packer, and Go-compiled binaries- indicating that the bundle is assembled from heterogeneous, separately built components rather than a single codebase. Figure 3 shows the resulting process tree, with the parent setup_install.exe fanning out into the dozens of Mon*.exe payloads it launches; the "Modify Memory" and "Modify Control Flow" edges between nodes mark the code-injection relationships that appear as Process Hollowing in Figure 2.

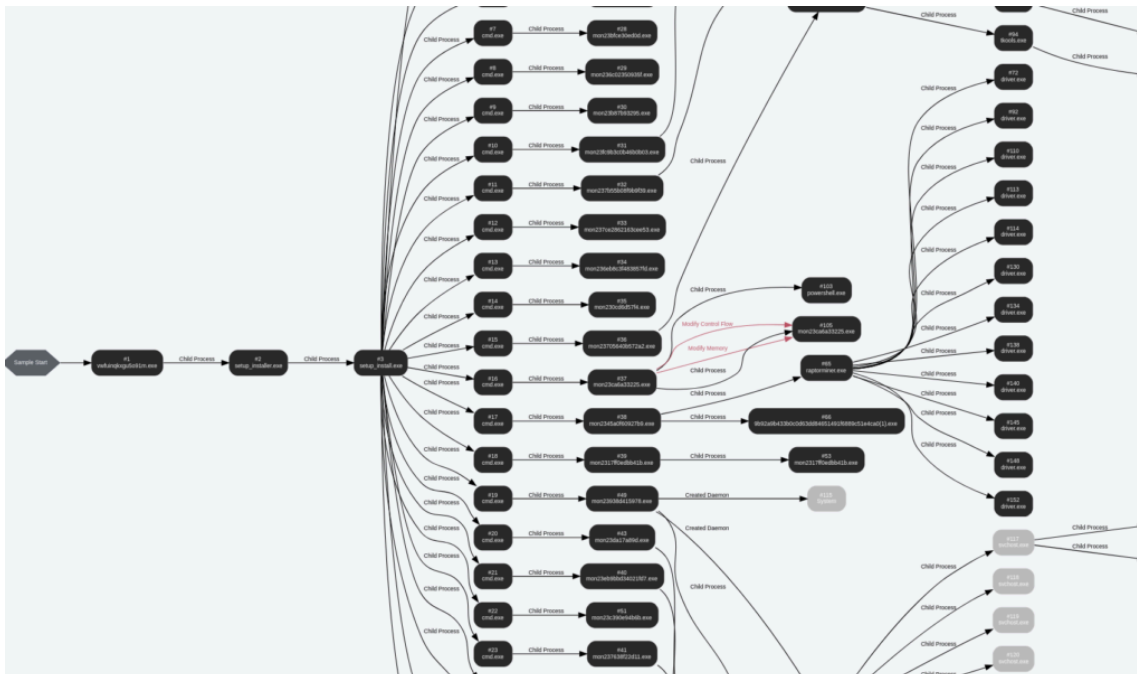
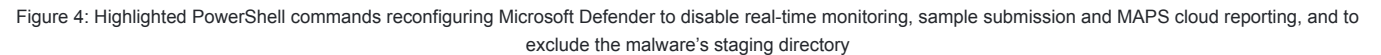


Figure 3: Process Tree Monitored by VMRay Platform During Dynamic Analysis

The bundle invests heavily in remaining undetected. As shown in Figure 4, it uses PowerShell to reconfigure Microsoft Defender: it disables real-time monitoring, blocks automatic sample submission, turns off MAPS cloud-based reporting, and adds its own staging directory under %LOCALAPPDATA%\Temp as a scan exclusion. It also attempts to detect the presence of eleven security products (including 360 Total Security, AVG, Avast, Avira, Bitdefender, Comodo, ESET, Kaspersky, Norton, Panda and Sophos) by file artifact and WMI query, and stops or deletes the Windows Update service to inhibit remediation.



5/23

```

[0032.503] NtQuerySystemInformation (SystemInformationClass=0x0, SystemInformation=0x10e704, Length=0x0, ResultLength=0x0)
[0032.503] NtQueryInformationProcess (ProcessHandle=0xffffffff, ProcessInformationClass=0x7, ProcessInformation=0x10e70c, ProcessInformationLength=0x4, ResultLength=0x0)
[0032.504] GetModuleHandleA (lpModuleName="ntledit") returned 0x0
[0032.504] GetModuleHandleA (lpModuleName="sshook") returned 0x0
[0032.504] GetModuleHandleA (lpModuleName="hook") returned 0x0
[0032.504] LocalAlloc (uFlags=0x0, uBytes=0x104) returned 0x23cd08
[0032.504] RtlCreate (in: lpString1="", lpString2="\\REGISTRY\\MACHINE\\System\\CurrentControlSet\\Enum\\IDE" | out:
  lpString1="\\REGISTRY\\MACHINE\\System\\CurrentControlSet\\Enum\\IDE" returned: "\\REGISTRY\\MACHINE\\System\\CurrentControlSet\\Enum\\IDE"
[0032.504] RtlInitUnicodeString (in: DestinationString=0x10e6d8, SourceString="\\REGISTRY\\MACHINE\\System\\CurrentControlSet\\Enum\\IDE" | out:
  DestinationString="\\REGISTRY\\MACHINE\\System\\CurrentControlSet\\Enum\\IDE")

```

Figure 5: Evasive Function Calls Logged by VMRay Platform

The infection establishes long-term control through several dropped child components: one creates the fake system process C:\Windows\rss\css.exe and registers it in the Windows startup registry, while another schedules logon-triggered tasks following the Schedule.Service.1 naming pattern (created via COM) for both the fake css.exe and a second implant staged under %APPDATA%\Roaming\teieedr. The miner component persists separately through a RaptorMiner.exe run key and a Driver.url entry in the Windows Startup folder. The fake css.exe additionally installs a set of services with VirtualBox-style names (VBoxGuest, VBoxMouse, VBoxSF, VBoxService, VBoxVideo, VBoxWddm), opens local TCP listeners (observed on ports 31461 and 49703), adds a Windows Firewall rule for itself, and invokes bcdedit, behaviour consistent with a rootkit/botnet implant establishing resilient, low-level control. The creation of these VirtualBox-named services through repeated CreateServiceW calls is captured in Figure 6. Across the run the malware enables an extensive set of process privileges (for example SeDebugPrivilege, SeImpersonatePrivilege, SeLoadDriverPrivilege and SeTakeOwnershipPrivilege) and performs process hollowing and injection (including MSHTA abuse and a dropped injector.exe), giving it the access required to manipulate other processes and the system.

```

[0197.451] GetProcAddress (hModule=0x76120000, lpProcName="CreateServiceW") returned 0x7614712c
[0197.451] CreateServiceW (in: hSManager=0xe258a8, lpServiceName="VBoxWddm", lpDisplayName="VBoxWddm", dwDesiredAccess=0xf003f, dwServiceType=0x10, dwStartType=0x4, dwErrorControl=0x0, lpBinaryPathName="VBoxWddm", lpLoadOrderGroup=0x0, lpdwTagId=0x0, lpDependencies=0x0, lpServiceStartName=0x0, lpPassword=0x0 | out: lpdwTagId=0x0) returned 0xe256f0
[0197.488] GetProcAddress (hModule=0x76120000, lpProcName="CloseServiceHandle") returned 0x7613369c
[0197.488] CloseServiceHandle (hSObject=0xe256f0) returned 1
[0197.489] CreateServiceW (in: hSManager=0xe258a8, lpServiceName="VBoxSF", lpDisplayName="VBoxSF", dwDesiredAccess=0xf003f, dwServiceType=0x10, dwStartType=0x4, dwErrorControl=0x0, lpBinaryPathName="VBoxSF", lpLoadOrderGroup=0x0, lpdwTagId=0x0, lpDependencies=0x0, lpServiceStartName=0x0, lpPassword=0x0 | out: lpdwTagId=0x0) returned 0xe25740
[0197.503] CloseServiceHandle (hSObject=0xe25740) returned 1
[0197.504] CreateServiceW (in: hSManager=0xe258a8, lpServiceName="VBoxMouse", lpDisplayName="VBoxMouse", dwDesiredAccess=0xf003f, dwServiceType=0x10, dwStartType=0x4, dwErrorControl=0x0, lpBinaryPathName="VBoxMouse", lpLoadOrderGroup=0x0, lpdwTagId=0x0, lpDependencies=0x0, lpServiceStartName=0x0, lpPassword=0x0 | out: lpdwTagId=0x0) returned 0xe256f0
[0197.518] CloseServiceHandle (hSObject=0xe256f0) returned 1
[0197.518] CreateServiceW (in: hSManager=0xe258a8, lpServiceName="VBoxGuest", lpDisplayName="VBoxGuest", dwDesiredAccess=0xf003f, dwServiceType=0x10, dwStartType=0x4, dwErrorControl=0x0, lpBinaryPathName="VBoxGuest", lpLoadOrderGroup=0x0, lpdwTagId=0x0, lpDependencies=0x0, lpServiceStartName=0x0, lpPassword=0x0 | out: lpdwTagId=0x0) returned 0xe25740
[0197.556] CloseServiceHandle (hSObject=0xe25740) returned 1
[0197.556] CreateServiceW (in: hSManager=0xe258a8, lpServiceName="VBoxService", lpDisplayName="VBoxService", dwDesiredAccess=0xf003f, dwServiceType=0x10, dwStartType=0x4, dwErrorControl=0x0, lpBinaryPathName="VBoxService", lpLoadOrderGroup=0x0, lpdwTagId=0x0, lpDependencies=0x0, lpServiceStartName=0x0, lpPassword=0x0 | out: lpdwTagId=0x0) returned 0xe256f0
[0197.588] CloseServiceHandle (hSObject=0xe256f0) returned 1
[0197.588] CreateServiceW (in: hSManager=0xe258a8, lpServiceName="VBoxVideo", lpDisplayName="VBoxVideo", dwDesiredAccess=0xf003f, dwServiceType=0x10, dwStartType=0x4, dwErrorControl=0x0, lpBinaryPathName="VBoxVideo", lpLoadOrderGroup=0x0, lpdwTagId=0x0, lpDependencies=0x0, lpServiceStartName=0x0, lpPassword=0x0 | out: lpdwTagId=0x0) returned 0xe25740

```

Figure 6: Creation of VirtualBox-named Windows services (CreateServiceW), logged by the VMRay Platform

Collection and command-and-control: For data theft, a dropped payload searches sensitive data from Google Chrome and Internet Explorer / Edge, harvests stored browser credentials, queries a cryptocurrency-wallet authenticator extension, and captures the screen via the BitBlt API. For communication it reaches a broad, geographically dispersed set of endpoints, several over uncommon ports, and uses dead-drop resolvers on legitimate platforms (Telegram, Mastodon and Pastebin) to retrieve next-stage addresses while blending with normal traffic. Critically for this investigation, a dropped child process issues the requests to campaign infrastructure described later in this report, specifically a GET to 212.193.30[.]29/server.txt and contact with 212.193.30[.]45/proxies.txt; Figures 7 and 8 capture these two requests in the function log. This link is observable only at the behavioural level: in isolation, the dropper and these endpoints carry no static indicator that would associate them, yet at execution time the malware itself reaches out and binds them together. It is this runtime behaviour that connects the dropper directly to the proxy and botnet infrastructure, and to the operator-side coordination backend, analysed in the following sections.

```

[0028.691] WinHttpOpen (pszAgentW="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36", dwAccessType=0x0, pszProxyW=0x0, pszProxyBypassW=0x0, dwFlags=0x0) returned 0x3f3870
[0028.691] WinHttpConnect (hSession=0x3f3870, pszServerName="212.193.30.29", nServerPort=0x50, dwReserved=0x0) returned 0x40ba30
[0028.691] WinHttpRequest (hConnect=0x40ba30, pszVerb="GET", pszObject="server.txt", pszVersion=0x0, pszReferrer=0x0, ppwszAcceptTypes=0x0, dwFlags=0x0) returned 0x408128
[0028.691] RtlAllocateHeap (HeapHandle=0x3d0000, Flags=0x0, Size=0x70) returned 0x40be50
[0028.691] WinHttpSendRequest (hRequest=0x408128, lpzHeaders=0x0, dwHeadersLength=0x0, lpOptional=0x0, dwOptionalLength=0x0, dwTotalLength=0x0, dwContext=0x0) returned 1

```

Figure 7: GET request for /server.txt to 212.193.30[.]29, seen in the VMRay Platform function log

```

[0029.243] WinHttpOpen (pszAgentW="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36", dwAccessType=0x0, pszProxyW=0x0, pszProxyBypassW=0x0, dwFlags=0x0) returned 0x6640c0
[0029.260] WinHttpConnect (hSession=0x6640c0, pszServerName="212.193.30.45", nServerPort=0x50, dwReserved=0x0) returned 0x671610
[0029.263] WinHttpRequest (hConnect=0x671610, pszVerb="GET", pszObject="proxies.txt", pszVersion=0x0, pszReferrer=0x0, ppwszAcceptTypes=0x0, dwFlags=0x0) returned 0x6716f8
[0029.263] RtlAllocateHeap (HeapHandle=0x640000, Flags=0x0, Size=0x70) returned 0x673c88
[0029.263] WinHttpSendRequest (hRequest=0x6716f8, lpzHeaders=0x0, dwHeadersLength=0x0, lpOptional=0x0, dwOptionalLength=0x0, dwTotalLength=0x0, dwContext=0x0) returned 1

```

Figure 8: GET request for /proxies.txt to 212.193.30[.]45, seen in the VMRay Platform function log

In summary, “setup_x86_x64_install.exe” is best characterised not as a stealer in isolation but as the delivery and orchestration stage of the campaign: a loader whose purpose is to install Raccoon Stealer alongside additional stealers, a loader/botnet layer and a coin miner, to disable defences and remediation, to establish rootkit-grade persistence, and to enrol the compromised

host into the wider operation, both as a proxy-botnet node and as a managed agent of the operator console, examined in the remainder of this report.

2.2 Command-and-Control Relations

Among the many endpoints contacted during execution, two are central to this campaign and are the focus of this section: 212.193.30[.]29, from which a dropped child process retrieves /server.txt, and 212.193.30[.]45, from which the same process retrieves /proxies.txt. Both servers are hosted in Russia on infrastructure operated by TimeWeb Ltd. While several of the dropped payloads carry their own commodity command-and-control endpoints (for example the RedLine Stealer C2 at 185.215.113[.]44:23759, the Socelars URL www.wgqpw.com, and the XMRig pool pool.supportxmr.com:3333), the 212.193.30.0/24 pair, by contrast, is specific to the campaign under investigation and represents the layer that ties the infected host to the actor's own infrastructure. The remaining payload-specific endpoints are catalogued in the indicators section and are not the subject of this analysis.

The contact with these servers is performed at the WinHTTP layer rather than through a browser or a higher-level component. Execution traces show a dropped child process calling WinHttpOpen, WinHttpConnect to “212.193.30[.]29” on port 80, and WinHttpOpenRequest with the method “GET” and the resource `/server.txt`. A notable and recurring artifact is the User-Agent string supplied to WinHttpOpen, which is consistently malformed, as Figure 9 shows: the intended value Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36” appears with its leading token corrupted into non-Latin (Korean) code points, for example rendered as `꺀꺀꺀 Mozilla/5.0 ...`. This malformation is reproduced across multiple processes and requests, indicating it originates from the malware’s own hardcoded request-building routine rather than from transmission error. Packet captures confirm that the corruption reaches the wire: in the captured `/server.txt` request the transmitted User-Agent header is reduced to a single non-printable control byte (0x02), consistent with an uninitialised or otherwise defective User-Agent buffer in the malware. Whether observed in memory as a corrupted wide string or on the wire as a lone control character, the value serves as a strong, distinctive host and network-level detection opportunity, and its consistency across samples is itself useful for clustering activity belonging to this campaign.

```
[0028.690] RtlAllocateHeap (HeapHandle=0x3d0000, Flags=0x0, Size=0x20) returned 0x4060f8
[0028.690] RtlAllocateHeap (HeapHandle=0x3d0000, Flags=0x0, Size=0x20) returned 0x406148
[0028.690] HeapFree (In: hHeap=0x3d0000, dwFlags=0x0, lpMem=0x4060f8) | out: hHeap=0x3d0000) returned 1
[0028.690] HeapFree (In: hHeap=0x3d0000, dwFlags=0x0, lpMem=0x4060d8) | out: hHeap=0x3d0000) returned 1
[0028.691] RtlAllocateHeap (HeapHandle=0x3d0000, Flags=0x0, Size=0x70) returned 0x403e10
[0028.691] WinHttpOpen (pszAgent=L"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36", dwAccessType=0x0,
[0028.691] WinHttpConnect (hSession=0x3f3870, pszServerName="212.193.30.29", nServerPort=0x50, dwReserved=0x0) returned 0x403a30
[0028.691] WinHttpOpenRequest (hConnect=0x403a30, pszVerb="GET", pszObjectName="/server.txt", pszVersion=0x0, pszReferer=0x0, ppwszAcceptTypes=0x0, dwFlags=0x0) returned 1
[0028.691] RtlAllocateHeap (HeapHandle=0x3d0000, Flags=0x0, Size=0x70) returned 0x40bbe0
[0028.691] WinHttpSendRequest (hRequest=0x408128, lpszHeaders=0x0, dwHeadersLength=0x0, lpOptional=0x0*, dwOptionalLength=0x0, dwTotalLength=0x0, dwContext=0x0) returned 1
[0028.790] HeapFree (In: hHeap=0x3d0000, dwFlags=0x0, lpMem=0x40bbe0) | out: hHeap=0x3d0000) returned 1
[0028.790] WinHttpReceiveResponse (hRequest=0x408128, lpReserved=0x0) returned 0
[0029.036] HeapFree (In: hHeap=0x3d0000, dwFlags=0x0, lpMem=0x403e10) | out: hHeap=0x3d0000) returned 1
```

Figure 9: The malformed User-Agent string (WinHttpOpen) seen in the VMRay Platform function log

Packet captures of this request clarify the true nature of the endpoint. The malware issues a plaintext request, GET /server.txt over port 80, carrying the single-byte User-Agent noted above, and the server, identifying itself as nginx, replies with an HTTP 301 Moved Permanently whose location header is https://212.193.30[.]29/server.txt and whose body is the standard 162-byte nginx redirect page. This 301 is therefore a routine HTTP-to-HTTPS upgrade pointing back to the same path on the same host; it is not a redirect toward GitHub, and it does not in itself return any configuration or steering data. The GitHub-blending behaviour described below belongs to the proxy host “212.193.30[.]45”, not to “212.193.30[.]29”. Within the dynamic analysis the implant does not follow the upgrade to HTTPS, its routine proceeds directly to its next hardcoded resource, so the meaningful server-side content sits behind the HTTPS endpoint and is examined in Section 2.3. The significance of “212.193.30[.]29” is consequently not this redirect but the host itself: as Figure 10 shows, browsing to it over HTTPS returns the “Standoff Coord” console and a TLS certificate issued for the operator-side domain “russianhackers[.]online” by Let’s Encrypt, providing the first concrete link between the malware’s runtime traffic and the actor’s coordination backend.

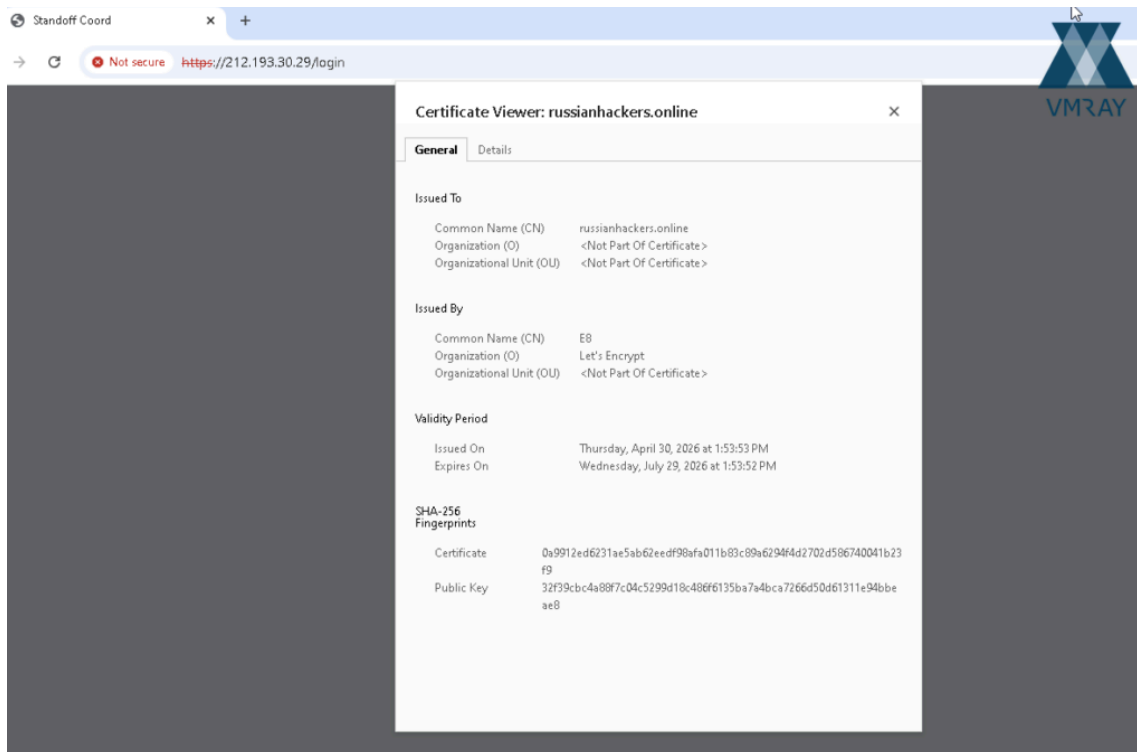


Figure 10: TLS certificate for russianhackers[.]online s

The request to 212.193.30[.]45/proxies.txt, shown in Figure 11, is, on the available evidence, the clearest indication of the campaign's botnet orientation. The resource name and observed behaviour indicate that the implant retrieves a list of proxy endpoints. This pattern is consistent with a proxy-oriented botnet in which compromised hosts are supplied with, and in turn contribute to, a rotating pool of proxy nodes, effectively turning victim machines into relay infrastructure ("zombies") that the operators can use to launder and redirect their own traffic. This interpretation is reinforced by the fact that the same endpoint behaves differently depending on how it is queried (for example returning a successful response to a direct request while returning a 406 status to requests carrying a /proxy-style path), behaviour typical of a gated distribution endpoint that filters requests rather than a static file. Treated conservatively, the confirmed fact is the retrieval of a proxy list from actor-controlled infrastructure; the botnet/zombie-relay role is a well-supported assessment built on that fact together with the surrounding behaviour.

```
[0029.243] RtlAllocateHeap (HeapHandle=0x640000, Flags=0x0, Size=0x20) returned 0x662ba8
[0029.243] HeapFree (in: hHeap=0x640000, dwFlags=0x0, lpMem=0x662b58 | out: hHeap=0x640000) returned 1
[0029.243] HeapFree (in: hHeap=0x640000, dwFlags=0x0, lpMem=0x6607e0 | out: hHeap=0x640000) returned 1
[0029.243] RtlAllocateHeap (HeapHandle=0x640000, Flags=0x0, Size=0x70) returned 0x663860
[0029.243] WinHttpOpen (pszAgentId="微付网flla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36", dwAccessType=0x0,
[0029.260] WinHttpConnect (hSession=0x6640c0, pszServerName="212.193.30.45", nServerPort=0x58, dwReserved=0x0) returned 0x671610
[0029.263] WinHttpOpenRequest (hConnect=0x671610, pszVerb="GET", pszObjectName="/proxies.txt", pszVersion=0x0, pszReferrer=0x0, ppwszAcceptTypes=0x0, dwFlags=0x0) retu
[0029.263] RtlAllocateHeap (HeapHandle=0x640000, Flags=0x0, Size=0x70) returned 0x673c88
[0029.263] WinHttpSendRequest (hRequest=0x6716f8, lpzHeaders=0x0, dwHeadersLength=0x0, lpOptional=0x0*, dwOptionalLength=0x0, dwTotalLength=0x0, dwContext=0x0) returned 0
[0030.505] HeapFree (in: hHeap=0x640000, dwFlags=0x0, lpMem=0x673c88 | out: hHeap=0x640000) returned 1
[0030.505] HeapFree (in: hHeap=0x640000, dwFlags=0x0, lpMem=0x663860 | out: hHeap=0x640000) returned 1
[0030.505] HeapFree (in: hHeap=0x640000, dwFlags=0x0, lpMem=0x6607a8 | out: hHeap=0x640000) returned 1
[0030.505] HeapFree (in: hHeap=0x640000, dwFlags=0x0, lpMem=0x6637e8 | out: hHeap=0x640000) returned 1
```

Figure 11: Requesting /proxies.txt via port 80 seen in VMRay Platform's function log

A distinct concealment technique is exhibited by the proxy-list host, 212.193.30[.]45, and by a broader set of Russian servers related to it: in response to any incoming HTTP request, for example from a security scanner, an automated crawler, or a researcher probing the host, they return an HTTP 301 redirect to https://github[.]com/ with an empty body, a response Figure 12 captures in the "Shodan" banner for the host, so that to a casual scanner or analyst the host appears to be a benign redirector toward a trusted domain. It must be emphasised that 212.193.30[.]29 does not behave this way; as shown above, it performs an HTTP-to-HTTPS upgrade and serves the operator console. The GitHub-redirect characteristic belongs to 212.193.30[.]45 and the wider cluster. This GitHub-blending frustrates infrastructure-level threat hunting; as stated previously, it does not implicate GitHub itself, which is used only as a trusted-looking redirect target. The shared TLS certificate and JARM fingerprint, together with the consistent "301 redirect to https://github[.]com/" response across hosts on the same Russian provider, make this an effective pivot for enumerating related infrastructure, which is the basis for the expanded footprint reconstructed in the following section.



Figure 12: The HTTP 301 redirect to github[.]com in the Shodan banner for 212.193.30[.]45 (port 443)

Beyond the two primary servers, the malware demonstrates redundancy in how it locates its next stages. As noted in Section 2.1, components in the bundle resolve follow-on addresses through dead-drop resolvers hosted on legitimate platforms, Telegram (<http://t.me/borderxra> and <http://t.me/jredmankun>), Mastodon instances ([noc.social/@menaomi](https://social.menaomi.org/) and qoto.org/@mniami), and Pastebin (a raw paste that has since been taken down by Pastebin for harmful/abusive content), and through additional loader panels and pay-per-install hosting. From a C2 perspective, this layering means the loss of any single endpoint does not sever operator control. Configuration and proxy information can be re-served from the primary servers, while dead-drops provide a low-cost, takedown-resistant mechanism for redirecting implants to fresh infrastructure. This resilience, taken together with the proxy-list distribution and the persistent rootkit component described earlier, supports the overall assessment of an actively maintained operation rather than a single static C2. The actor's wider command-and-control infrastructure, and the multi-operator console behind it, are examined in the section below.

It is also worth noting that many of the malware's distribution endpoints and domains are currently unavailable or unreachable; the proxy-botnet infrastructure and the operator command-and-control console, however, remain active and in use by the threat actors responsible for this campaign. For this reason, this report focuses on the active command-and-control infrastructure and the operator-side intrusion capability behind the campaign.

2.3 Infrastructure Pivoting and the Operator Command-and-Control Platform

By pivoting from the UniqueSignal threat intelligence feed, VMRay Labs team moved from already-known indicators to the discovery of previously undetected infrastructure, revealing a novel Russian-speaking threatactor campaign.

The investigation focused on the two command-and-control nodes the malware contacts at runtime, both hosted in Russia by TimeWeb Ltd.: 212.193.30[.]45, from which it requests `/proxies.txt`, and 212.193.30[.]29, from which it requests `/server.txt`. As established in Section 2.2, these are not independent observations, a single malware process issues both requests from one hardcoded WinHTTP routine, in immediate succession and sharing the same private heap (0x3d0000), so that 212.193.30[.]29 was discovered simply by following the implant's own call sequence outward from the proxy host 212.193.30[.]45. The two nodes nonetheless serve very different roles, examined in turn below: 212.193.30[.]29 fronts the operators' coordination console, while 212.193.30[.]45 is the proxy/redirector node from which the wider campaign infrastructure was enumerated.

Investigation of "212.193.30[.]29" showed that it presents a TLS certificate issued in the name of `russianhackers[.]online` and indeed serves that domain. As established in Section 2.2, the malware's plaintext request to `http://212.193.30[.]29/server.txt` receives only a routine HTTP-to-HTTPS upgrade redirect, which the implant does not follow; the HTTPS endpoint, however, returns the operators' web console, a single-page application delivered from `https://212.193.30[.]29/` whose HTML loads a JavaScript bundle at `https://212.193.30[.]29/assets/index-CxeLM22L.js`. As Figure 13 shows, this console is branded "STANDOFF COORD" and presented behind a login panel, and it exposes the internal operations, structure and human workflow of the threat actor's command-and-control backend, the heart of this investigation.

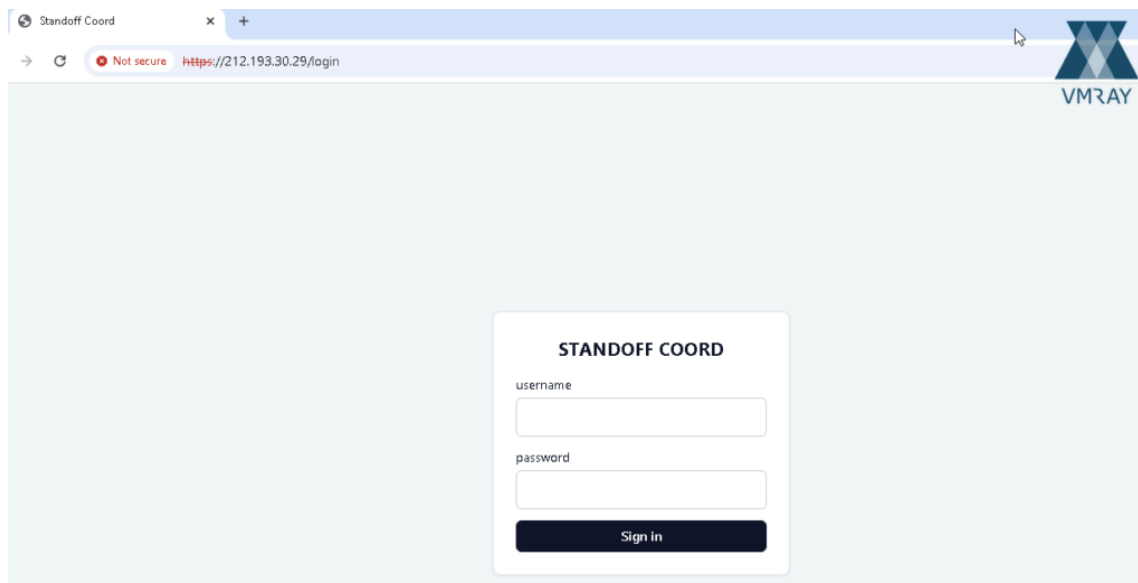


Figure 13: The STANDOFF COORD operator console login panel, served over HTTPS by 212.193.30[.]29

The retrieved bundle is not a malware payload but a minified production React single-page application, the human-facing operator console of the backend. It is hardcoded to call a backend API at "https://api.russianhackers[.]online" and authenticates operators with a bearer token, persisting session and selected-workspace state under the browser keys "standoff.token" and "standoff.workspace". Architecturally the console is a multi-tenant, collaborative operations platform built around "workspaces", discrete engagements, each with its own members, targets and artefacts, far closer in design to a commercial penetration-testing or red-team management product than to a commodity malware panel. Its API surface (among others /workspaces/{id}/assets, /credentials/{id}/reveal, /findings, /risks, /tickets, /claims, /notes/feed and /members, alongside the /agents endpoints discussed below) describes a complete operational backend rather than a single victim log.

Recovered from the console bundle (index-CxeLM22L.js):

```
Gm="https://api.russianhackers[.]online"
config.headers.Authorization=Bearer ${n}
localStorage.getItem("standoff.token") ; localStorage.setItem("standoff.workspace",i[0].id)
G.get(/workspaces/${e}/assets/); G.get(/workspaces/${e}/credentials/${t}/reveal);
G.get(/workspaces/${e}/members)
```

The most significant component is the agents subsystem, the operator-side counterpart to the implants of Section 2.1. The bundle defines endpoints to enrol an agent and issue it a token (POST /agents/issue-token), to receive periodic check-ins (POST/agents/{id}/heartbeat), and to halt one (POST/agents/{id}/stop). The interface explicitly separates automated "agents" from human "operators", and references a "progress agent" that posts a status digest every five minutes. This is the operator-side model under which a compromised host, having retrieved its configuration from /server.txt on this same host, is enrolled and maintained as a managed agent; the token-based enrolment and heartbeat cadence correspond directly to the implant's beaconing observed during execution, closing the loop between the malware's runtime traffic and the coordination backend.

Recovered from the console bundle:

```
issueToken:e=>G.post("/agents/issue-token",e)
heartbeat:(e,t)=>G.post(/agents/${e}/heartbeat,t); stop:e=>G.post(/agents/${e}/stop)
"🤖 любой агент" / "👤 операторы" ("any agent" / "operators")
"История дайджестов пуста. Прогресс-агент пишет сюда раз в 5 мин." ("Digest history is empty. The progress agent writes here every 5 min.")
```

Through the assets and agents components, the platform maintains a real-time inventory of targeted and compromised hosts. Each host record carries:

- Hostname

- IP Address
- Operating System
- User Context
- A Version Value
- Free-Form Labels and Tags
- First and Last Seen Timestamps
- Live State, Shown as a Coloured Status Marker (Online, Idle or Stopped)

Together these fields give operators an at-a-glance view of active footholds. Hosts are grouped by network “segment”, with values internal, external and dmz that mirror enterprise network zoning, and each engagement is bounded by a scope expressed as CIDR ranges – both indicating an operation that reasons explicitly about its position within a victim network. A companion artefacts repository stores operational evidence under defined types – screenshot, image, log, file, binary, dump and pcap – meaning the operators retain memory dumps, packet captures and recovered binaries, and attach screenshots as proof of compromise (with a 50 MB upload limit and backend magic-byte validation of uploads).

Recovered from the console bundle:

```
const t=["ip","hostname","os","segment","services","tags","first_seen","last_seen"]
s.status==="stopped"?:s.status==="idle"?:
"option",{value:"dmz",children:"dmz"};"option",{value:"dump",children:"dump"};"option",{value:"pcap",children:"pcap"}
```

The credentials component is especially revealing. Rather than storing free-form secrets, it constrains each captured credential to a defined type, and the available types are themselves a statement of intent: password, hash, kerberos_ticket, api_token, cookie and private_key. This is the toolkit of an actor focused on credential theft and lateral movement in Windows and Active Directory environments, NTLM hashes for pass-the-hash, Kerberos tickets for pass-the-ticket and ticket forging, session cookies for authentication-token hijacking, and private keys and API tokens for onward access. Each entry is tied to a target principal recorded in domain form (for example “corp\svc-backup”) and exposes an explicit “reveal” action that discloses the secret value only on demand.

Recovered from the console bundle:

```
"option",{children:"password"};{children:"hash"};{children:"kerberos_ticket"};{children:"api_token"};{children:"cookie"};
{children:"private_key"}
reveal:(e,t)=>G.get(/workspaces/${e}/credentials/${t}/reveal)
placeholder:"principal (например: corp\svc-backup)"
```

The platform’s workflow is formal and measured. Findings and risks carry a severity of critical, high, medium, low or info, while tasks are triaged on a P0-P3 priority scale (P0 denoting a blocker), a vulnerability-management discipline more typical of a professional security team than of a criminal panel. Work items advance through an explicit state machine (not_started, in_progress, attempted, confirmed_exploit, failed, blocked) and targets through a parallel lifecycle (new, confirmed, exploited, reported), with the confirmed_exploit state awarding points to the responsible operator. A live “claims” mechanism lets an operator claim a target or task and later release it, deconflicting concurrent work so that two operators do not act on the same target at once, in itself evidence of genuine multi-operator use. For clarity, and consistent with this report’s evidence-driven approach, confirmed_exploit is a progress label on the operators’ own scoreboard and is not evidence of a software exploit within the malware.

Recovered from the console bundle:

```
["not_started","in_progress","attempted","confirmed_exploit","failed","blocked"]
(n.data??[]).filter(S=>S.status==="confirmed_exploit").reduce((S,f)=>S+(f.points||0),0)
```

The console also exposes the operators’ collaboration surface and working language. It provides an “Agent Chat”, per-item comment threads, an acknowledged notes feed, and a structured internal Knowledge Base, a navigable tree of directories and

articles that supports embedded Mermaid diagrams – in which the operators maintain and visually document their own playbooks and methodology; authored content is tagged by source, distinguishing entries written by human operators from those posted automatically by agents. The interface text is written predominantly in Russian, and operators are explicitly instructed to write “на русском” (in Russian). Its placeholder examples are not generic filler but operational tradecraft, instructing operators to record, for instance, that they “received SYSTEM on dc01 via PrintNightmare” (CVE-2021-34527, against a domain controller), to “select a wordlist for AD spray” (Active Directory password spraying), to set a target_scope as a CIDR or hostname, to enter service-account principals such as “corp\svc-backup”, and to triage unread “P0-blocker” items – the working vocabulary of a team operating against enterprise Active Directory.

Recovered from the console bundle:

```
“🗨 Agent Chat” ; “/kb/tree” ; “/kb/article” ; “/kb/search” ; “/kb/history”  
placeholder: “описание (markdown, на русском)” (“description (markdown, in Russian)”)  
“например: получил SYSTEM на dc01 через PrintNightmare” (“e.g.: got SYSTEM on dc01 via PrintNightmare”)  
“title (например: «Подберите wordlist для AD spray»)” (“e.g.: pick a wordlist for AD spray”)
```

Finally, several engineering and operational-security choices point to a mature, in-house development effort rather than an off-the-shelf panel. Record search is performed server-side with PostgreSQL pattern matching (the interface refers explicitly to “ILIKE” queries), indicating a conventional relational backend; an export function is deliberately constrained to omit stored secrets, which remain obtainable only through the reveal action, shielding the actor’s most sensitive holdings even from their own bulk exports; and the platform distinguishes privileged operators by an administrative flag, implying a tiered team rather than a flat group.

Recovered from the console bundle:

```
“🔍 поиск по тексту (server-side ILIKE)” (“text search (server-side ILIKE)”)  
“Экспорт по текущему фильтру (без секретов, они только через /reveal)” (“export by current filter – without secrets;  
only via /reveal”)  
{id:n.id,username:n.username,is_admin:!!n.is_admin}
```

These artefacts move the assessment beyond infrastructure indicators and into the operation’s internal character: an organised, Russian-speaking offensive team working through a deliberately engineered, multi-operator platform – credential vaults, segment-aware target tracking, an evidence repository, a shared knowledge base and a points-based scoreboard – to coordinate managed agents on compromised hosts.

By considering all of the above C2 behaviours and the relationship between the two servers, VMRay Labs continued the research with “212.193.30[.]45”, which behaves not as a console but as a proxy/redirector that fronts the actor’s infrastructure behind the GitHub domain. A request to the bare host is answered with a redirect toward “http://github[.]com”, whereas a request to the very resource the malware fetches – /proxies.txt – is met instead with an HTTP 406 response referencing the invalid path http://github[.]com/proxies.txt. This inconsistency is assessed as a misconfiguration in the threat-actor infrastructure, and it betrays the redirector’s non-genuine nature.

Because GitHub operates no official servers in Russia, a campaign-linked Russian IP address presenting a TLS certificate in the name of http://github[.]com and redirecting all requests toward it was itself a strong anomaly and was selected for further pivoting. The following content was served on port 443 (TCP) by “212.193.30[.]45”:

```
HTTP/1.1 301 Moved Permanently  
Content-Length: 0  
Location: https://github[.]com/  
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
```

This IP address is likewise hosted by TimeWeb Ltd. (AS9123) and also exposes an open SSH service on port 22. By pivoting on the server’s characteristics, its HTML content, certificate specifications and ASN, we identified 48 previously unattributed servers, assessed with high confidence to be command-and-control nodes of the campaign. The set numbered 48 at the outset of the investigation and has since reduced to the 44, which are included as indicators of compromise in this report.

It was also observed that some of these C2 servers expose port 10050 and run proxy software such as Traefik and TinyProxy; almost all of them additionally expose an open SSH port and, as a defining trait of this cluster, redirect their traffic to GitHub. On one server, alongside the GitHub domain, several Microsoft subdomains were also being redirected to, together with a Russian-language website, <http://loopaai.ru>. Although that server later became unreachable, VMRay Labs recorded the domains it had redirected to:

- <http://github.com>
- <http://www.github.com>
- <http://microsoft.com>
- <http://www.microsoft.com>
- <http://privacy.microsoft.com>
- staticview.microsoft.com
- <http://c.s-microsoft.com>
- <http://i.s-microsoft.com>
- <http://loopaai.ru>
- <http://www.loopaai.ru>

It is highly unusual for a Russian server to proxy toward all of these domains, which may indicate that the threat actors are also attempting to conceal their infrastructure behind Microsoft-owned domains.

Having mapped this hidden infrastructure, VMRay Labs continued by pivoting on “STANDOFF COORD”. Although the platform was unfamiliar at first, the discovery of one particular server on TimeWeb Ltd. -the same ASN on which the rest of the infrastructure resides, shed light on the situation: “217.198.13[.]211”.

This server hosts several domains which, at the time of publication, are likewise undetected by any security vendor as malicious and remain hidden:

- bull-drops.online
- bull-drops.ru
- bulldrops.online
- <http://bulldrops.ru>
- <http://xn--90aguaqgfu.xn--plai> (буллдроп.рф)
- <http://www.xn--90aguaqgfu.xn--plai> (www.буллдроп.рф)
- ggstandoff.online
- zadrot.gg
- influencesite.ru/gginfluence.influencesite.ru
- www.xn----9sbhgocsfmg4alkfg.xn--plai
- www.mobilearena.online
- xn----9sbhgocsfmg4alkfg.xn--plai

- mobilearena.online
- xn--90aguaqgfu.xn--plai

2.4 The Influence and Mass Outreach Capability: a Telegram Account Farm and an AI Driven Multi Channel Automation Platform

The six domains above do not front a decoy or placeholder website; the server “217.198.13[.]211” carries the threat actor’s own operator tooling. Alongside the public-facing site described below, the host serves two distinct, bespoke operator applications, a Telegram account-farm and a node-based automation platform (the latter delivered as the production JavaScript bundle `index-Caak4qkG.js` on port 80). Both rest on the same technology stack as the STANDOFF COORD console examined in Section 2.3, React single-page applications built with Vite, Zustand and axios against a JSON `/api/v1` backend secured by JWT bearer tokens, with access, and refresh-token rotation and a single-sign-on endpoint, which points to a common development team and toolchain across the operation.

These operator applications sit within a broader service footprint. The host, again on TimeWeb Ltd. (AS9123), listens on ports 443, 3000, 3001 and 8090, and presents a Let’s Encrypt TLS certificate issued for the punycode domain `xn--90aguaqgfu[.]xn--plai`. As Figure 14 shows, port 8090 returns a page titled “GG Influence”, evidently the operators’ own name for the apparatus, and a label whose “GG” prefix echoes the “ggstandoff” branding seen elsewhere across the infrastructure.

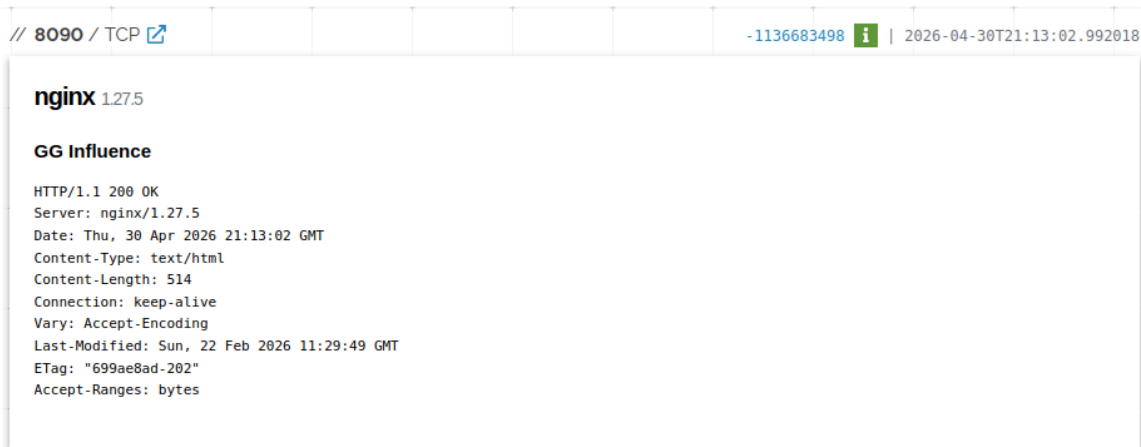


Figure 14: The “GG Influence” page returned by port 8090 of 217.198.13[.]211, seen in Shodan

Port 3000 is the most revealing of all. As Figure 15 shows, it serves a polished, Russian-language website built on the Next.js framework and branded “Мобильная Арена” (“Mobile Arena”), which presents itself as “киберспортивный портал о Standoff 2, PUBG Mobile и мобильных играх”, an e-sports portal devoted to Standoff 2, PUBG Mobile and other mobile games. Its content is a search-engine-optimised stream of material aimed squarely at young mobile gamers. The articles promise “free skins” and ways to earn in-game “gold without donating” in Standoff 2, offer promotional codes for Standoff 2 and PUBG Mobile, and feature tier lists and curated round-ups of “case” (loot-box) gambling sites. All of it is interlinked with category pages, a “bonuses” section, a “cooperation” (advertising and affiliate) page, and outbound Telegram and Discord links. The site’s own keyword metadata explicitly includes “ggstandoff”, binding it to the ggstandoff family of domains documented earlier in this report. This portal is best understood as the public, top-of-funnel face of the operation: a content lure that aggregates a gaming audience. It is positioned to channel that audience toward the monetisation and engagement-manipulation machinery described elsewhere in this report, using promotional codes, “free” in-game currency, skin-gambling referrals and Telegram links.

Мобильная Арена - Киберспортивный портал о Standoff 2, PUBG Mobile и мобильных играх

```
HTTP/1.1 200 OK
X-Content-Type-Options: nosniff
Referrer-Policy: strict-origin-when-cross-origin
X-Frame-Options: DENY
Permissions-Policy: camera=(), microphone=(), geolocation=(), payment=()
Cross-Origin-Opener-Policy: same-origin
Content-Security-Policy: default-src 'self'; base-uri 'self'; object-src 'none'; frame-ancestors 'none'; form-action 'self'; img-src 'self' data: blob: https: http:; font-src 'self' data:; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-inline'; connect-src 'self' https:; media-src 'self' https: http:; upgrade-insecure-requests
Strict-Transport-Security: max-age=15552000; includeSubDomains
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Accept-Encoding
Link: </_next/static/media/21350d82a1f187e9-s.p.woff2>; rel=preload; as="font"; crossorigin="", type="font/woff2", </_next/static/media/e4af272ccee01ff0-s.p.woff2>; rel=preload; as="font"; crossorigin="", type="font/woff2"
X-Powered-By: Next.js
Cache-Control: private, no-cache, no-store, max-age=0, must-revalidate
Content-Type: text/html; charset=utf-8
Date: Thu, 23 Apr 2026 11:36:09 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Transfer-Encoding: chunked
```

Figure 15: The “Мобильная Арена” (Mobile Arena) gaming portal on port 3000 of 217.198.13[.]211, seen in Shodan

The first application is a Russian-language Telegram account-farm and engagement-automation platform, served from a backend API at [http://217.198.13\[.\]211:8002/api/v1](http://217.198.13[.]211:8002/api/v1). It is divided into fourteen functional areas:

- Sessions
- Session Management
- Channels
- Channel Joins
- Unavailable Channels
- Comments
- Chats
- Dialog Builder
- Participant Parser
- Proxies
- Reputation
- Tasks
- Thread Monitor
- TG Codes

Its own interface text states its purpose without ambiguity, describing the “Автоматическое вступление ботов в каналы и дискуссии” (automatic joining of bots into channels and discussions) and inviting the operator to “Добавляйте и управляйте Telegram аккаунтами для ботов” (add and manage Telegram accounts for bots). The platform is organised around “sessions”, individual Telegram accounts operated as bots, and provides the full machinery required to acquire, maintain and weaponise them at scale.

For each account, the platform automates the Telegram authentication sequence from end to end, submitting the telephone number, the login code and, where required, the two-factor password (through the API endpoints `/sessions/auth/start`, `/sessions/auth/code` and `/sessions/auth/password`). To obtain those login codes automatically, it monitors Telegram’s official

service account, chat 777000, across the controlled accounts and through their assigned proxies, a function the interface labels “Мониторинг кодов входа из Telegram (чат 777000) через прокси сессии (Monitoring login codes from Telegram (chat 777000) via the session proxy)”. Each account is routed through a proxy drawn from a dedicated proxy-management module, which imports and validates proxy lists in bulk (the endpoints /proxies/import and /proxies/import-async), thereby consuming, directly, the pool of relay nodes produced by the proxy-botnet described in Sections 2.1 and 2.2. A reputation module supports the gradual “warming” of accounts in order to evade Telegram’s anti-abuse controls. Account warming refers to the practice of building up a new account’s activity history incrementally, joining a small number of channels, generating modest interaction, before deploying it for bulk operations; Telegram’s automated systems apply restrictions to accounts that engage in mass activity immediately after creation, and an account with an established activity history is significantly less likely to be flagged or suspended.

With a stock of authenticated accounts in hand, the platform drives them to join target channels and their linked discussion groups in bulk, importing channels en masse and tracking those that become “unavailable” once the bots are banned, and to scrape the participants of those channels into an exportable “Готовый список username для рассылок” (ready-made username list for mass distribution). The whole process is coordinated by a task scheduler, timed in Moscow time (“МСК”) and exposing scheduled tasks, execution histories and per-bot progress, so that joining, parsing and posting can be run continuously and at volume.

The platform’s defining feature is its content-generation engine. Through a visual “Dialog Builder”, the operators assemble automated conversation flows whose central element is a GPT node: a large-language-model component that produces Telegram comments and replies in the voice of a configurable persona, defined by a name, an age, a writing style and a goal; and bound to one of the farmed accounts. The engine can comment on channel posts, reply to individual messages, watch discussions for keywords and answer them automatically (“ИИ-ответы”, or AI replies), and sustain real-time conversations between its bots and genuine users. This is, in effect, a system built for coordinated inauthentic behaviour: the manufacture of synthetic engagement, opinion and conversation on Telegram at scale, under the guise of plausible human personas.

The second application, served as a production bundle on port 80, is a bespoke, node-based automation and orchestration platform, comparable in concept to commercial workflow-automation products, but custom-built. Presented in English, governed by role-based access control (with distinct Admin and Manager roles) and Google OAuth single sign-on, it allows operators to construct workflows on a visual canvas, embed code through an integrated editor, and execute them on schedules or webhooks (recorded as “runs”). Its catalogue of integrations is the clearest statement of its function: electronic mail through Gmail and IMAP, Google Sheets, the Anthropic “Claude” large-language model, Telegram, WhatsApp, generic HTTP requests, and arbitrary scripts, each triggered by cron expressions, webhooks or schedules.

The workflow primitives recovered from the code show precisely how those integrations are used: harvesting email addresses, reading inboxes over IMAP, drafting and sending mail, generating text and replies with Claude, and dispatching messages over Telegram and WhatsApp, every step backed by a typed vault of stored credentials that accommodates generic and Anthropic API keys, HTTP basic and bearer tokens, OAuth client identifiers and secrets, Gmail OAuth tokens and Google service-account keys. In practice, it is a multi-channel, AI-driven mass-outreach and engagement engine capable of running automated campaigns across e-mail, Telegram and WhatsApp; equally, through its scripted HTTP requests and stored credentials, it can serve as the connective tissue that binds the operation’s other platforms together.

Taken together, these platforms establish a third dimension to the campaign, alongside the access-and-monetisation layer of Sections 2.1 and 2.2 and the hands-on-keyboard intrusion console of Section 2.3. Several threads bind them to that wider operation: shared hosting on the same TimeWeb Ltd. autonomous system; the `bull-drops` and `xn-90aguaqgf[.]xn-plai` domains they serve, which remained undetected by security vendors at the time of writing; an identical bespoke technology stack and `/api/v1` authentication pattern; Russian-language interfaces and Moscow-time scheduling; and, most concretely, the Telegram farm’s reliance on the very proxy pool into which the malware’s victims are conscripted. The capture shown in Figure 16 corroborates several of these threads: it records “217.198.13[.]211” serving all of its resources from the same TimeWeb network and reveals that its frontend was served from a Vite/React development server, a configuration error that exposed the applications’ unminified source. Its construction is unmistakable: a Russian-speaking team has assembled an industrial capability for AI-assisted, multi-channel, coordinated inauthentic engagement and mass outreach.

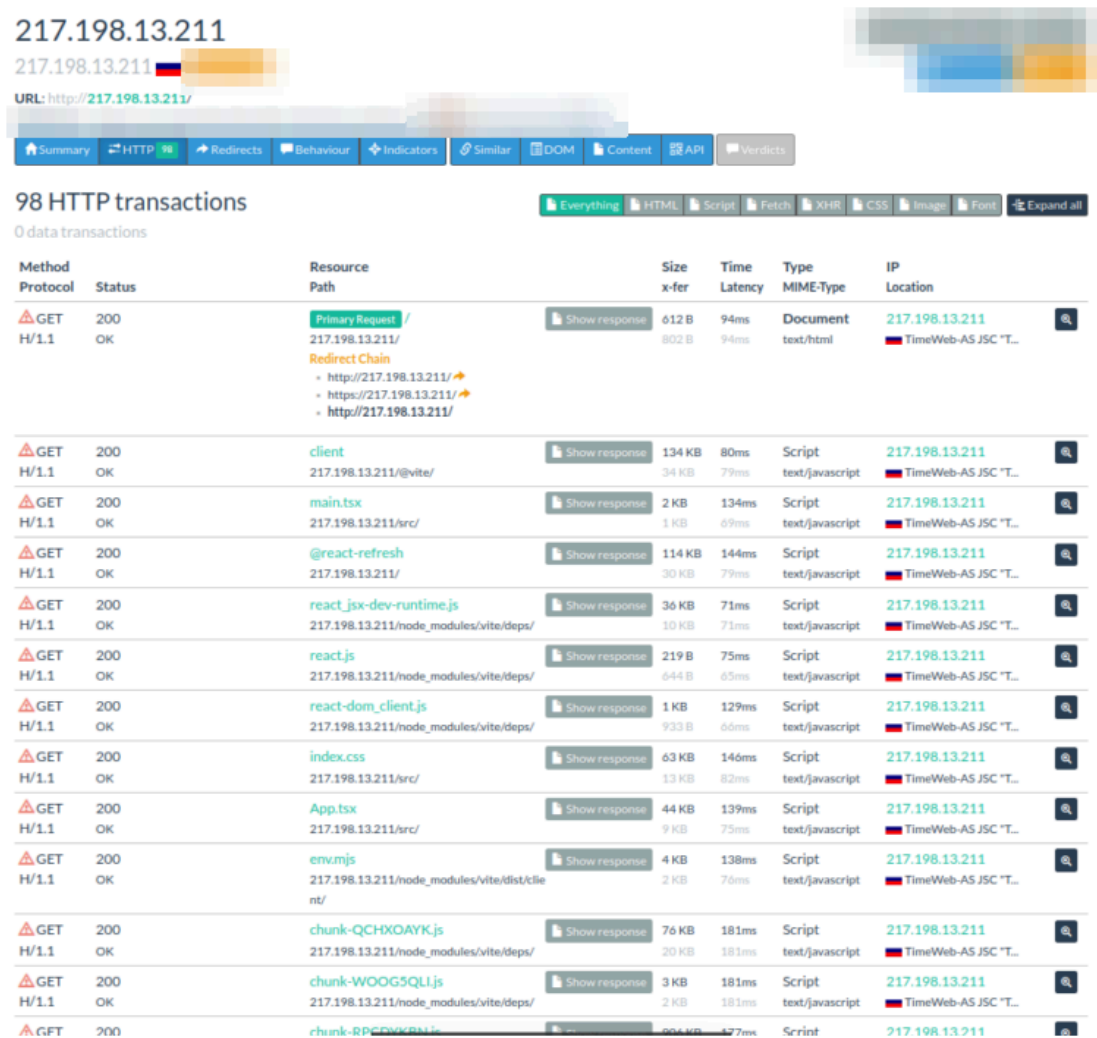


Figure 16: urlscan.io Capture of 217.198.13.211, Showing its HTTP Transactions and the Exposed Vite/React Development Server.

VMRay Labs assesses, with high confidence, that this infrastructure constitutes a large-scale, AI-driven disinformation and engagement-manipulation operation run by Russian-speaking actors. The combination of an industrial Telegram account-farm, artificial-intelligence personas that comment and converse under fabricated identities, a multi-channel automation hub spanning electronic mail, Telegram and WhatsApp, and a public gaming-content portal aimed at a young, Russian-speaking audience amounts to a coordinated capability for shaping opinion, manufacturing artificial consensus and engagement, and conducting mass outreach at scale. On the available evidence, the operation is assessed to serve one or more of the following ends: the dissemination of disinformation and the artificial amplification of narratives; the posting of fake product reviews and endorsements, together with gambling and fraud-related promotion across the Standoff 2 and mobile-gaming ecosystem; and the building and steering of a gaming audience toward the malware and proxy-botnet infrastructure documented earlier in this report. The specific narratives, targets and beneficiaries of the operation reside in server-side data that was beyond the scope of this analysis and are not asserted here as fact; the existence, sophistication and industrial scale of the capability, however, are established directly by the actors' own tooling.

3. Conclusion

Operation STANDOFF demonstrates how a contemporary Russian-speaking cybercriminal group can operate well beyond the boundaries of a conventional malware campaign. On a single body of shared infrastructure, the actors run four mutually reinforcing capabilities: the mass distribution of commodity malware, a proxy-botnet that conscripts victims into relay infrastructure, a bespoke console for the targeted hands-on-keyboard intrusion of enterprise networks, and an AI-driven, multi-channel apparatus for influence and mass outreach. These are bound together by common tooling, a common development team, and a common concealment strategy built around the GitHub domain.

The campaign's defining characteristic is convergence. Automated, opportunistic cybercrime; stealers, a coin miner and a proxy-botnet, is fused with deliberate, human-operated intrusion tradecraft, an Active Directory-oriented credential vault, segment-aware targeting, shared playbooks and a multi-operator scoreboard, and with a coordinated influence operation built on an industrial Telegram account-farm, AI personas, and a Standoff 2 and PUBG Mobile gaming lure. This breadth, combined with Russian-language tooling, Moscow-time scheduling and consistent "ggstandoff" branding, supports a high-confidence assessment of a single, organised Russian-speaking group pursuing multiple revenue and influence objectives in parallel.

Critically, the link between these disparate capabilities was established not by isolated indicators but by execution-level intelligence: by observing how the malware actually behaved, individually unremarkable artefacts, a malformed User-Agent, a proxy-list retrieval, a redirect to GitHub, were connected into a single operation and used to surface infrastructure that had evaded vendor detection.

Two points bear emphasis for defenders. First, much of this infrastructure remains active and undetected at the time of writing, including operator-controlled domains that no security vendor currently flags as malicious; the indicators provided in this report are intended to enable their detection and blocking. Second, the operation's most novel element, a purpose-built, AI-assisted influence and engagement-manipulation platform sharing infrastructure with a criminal intrusion set, reflects a broader trend in which financially motivated cybercrime, targeted intrusion and information operations increasingly converge within a single threat actor.

4. Indicators of Compromise

The indicators below are provided to support detection, hunting and blocking. Network indicators are listed as observed; many of the campaign domains and servers were not flagged as malicious by security vendors at the time of writing.

File name: setup_x86_x64_install.exe

MD5: e77221d7a4b47b9107ba1b61a551ca89

SHA-1: 95c5ae3fec0d900e4634e11b3ad81971e78e2b31

SHA-256: 22ebb950592ccc987fd1dab9ddcd34c4fc519975dc1b82e4a793dc038d2d8e41

212.193.30.29 (serves /server.txt; hosts russianhackers.online and the "STANDOFF COORD" console)

212.193.30.45 (serves /proxies.txt; GitHub-proxied redirector node)

russianhackers.online

api.russianhackers.online

217.198.13.211 (operator host: Telegram account-farm, automation platform and gaming-lure portal)

bull-drops.online

bull-drops.ru

bulldrops.online

http://bulldrops.ru

http://xn--90aguaqgfu.xn--plai

ggstandoff.online

zadrot.gg

influencesite.ru

www.mobilearena.online

xn----9sbhgocsfmg4alkfg.xn--plai

www.xn----9sbhgocsfmg4alkfg.xn--plai

mobilearena.online

5.129.196.85

5.129.208.108

5.129.209.17

5.129.209.58

5.129.210.32

5.129.210.139

5.129.214.85

5.129.216.104

5.129.217.228

5.129.219.114

5.129.225.220

5.129.226.97

5.129.213.59

5.129.213.241

5.129.227.196

5.129.231.176

5.129.231.240

5.129.233.99

5.129.236.52

5.129.236.68

5.129.237.19

5.129.237.53

5.129.238.90

5.129.238.104

5.129.238.105

5.129.239.229

5.129.242.37

37.252.21.227

45.139.78.67

46.149.70.18

89.223.71.207

90.156.224.57

92.51.22.34

93.183.80.126

147.45.183.198

147.45.237.23

185.247.185.85

188.225.39.252

188.225.72.157

188.225.82.125

194.87.56.156

194.87.131.30

195.133.73.225

212.60.21.249

RedLine Stealer: 185.215.113.44:23759 (MissionID "@Tui")

Socelars: www.wgqpw.com

XMRig (Monero): pool.supportxmr.com:3333 (wallet
8BFyHJmwhhxXo29aFXZrTJTWDbkiQFEsBBnj1VnHBcy9ZQ2NKEUGdKvZbWGRNYamgAgJ75jsX1bzDiVh21D5WShJPJVqaMU)

Amadey / Loader Panels: wfsdragon.ru/api/setStats.php (104.247.81[.]99) ; 212.192.241[.]62/base/api/statistics.php ;
185.215.113[.]35/d2VxjasuwS/index.php

SmokeLoader: rcacademy.at/upload/ (188.40.141[.]211)

"borderxra" download paths: 91.219.236[.]207/borderxra ; 91.219.237[.]227/borderxra ; 185.225.19[.]18/borderxra

Other Downloaders / C2: server5.trumops.com (3.229.117[.]57) ; www.listincode.com / listincode.com (172.237.145[.]27) ;
cloudjah.com/77_1.exe ; 65.108.69[.]168:16278 ; 23.88.118[.]113:23817

Pay-per-Install Hosting (Scaleway S3): coffee-music-laptop.s3.pl-waw.scw.cloud/publisher_installer/power0ff.exe (151.115.10[.]x) ; hammajawa7dou.s3.nl-ams.scw.cloud/advertiser_Installer/power0ff.exe (51.158.211[.]x)

Payload hosting: cdn.discordapp.com/attachments/915539163787460658/917347672489349130/myfile.exe

Dead-Drop Resolvers: http://t.me/borderxra; http://t.me/jredmankun;noc.social/@menaomi; qoto.org/@mniami ;
pastebin.com/raw/A7d5G1te

Victim Tracking: http://iplogger.org – paths /2ANpP6, /143up7, /1FRbw7, /1FEbw7

Decoy / Unresolved Domains Referenced in Payload Configurations: all-mobile-palments.com.mx ; buy-fantasy-football.com.sg ; buy-fantasy-gxmex.com.sg ; new-androidapps.me ; topniemannpickshop.cc ; blvckxxx.beget.tech

Staging directory: %LOCALAPPDATA%\Temp\7zSCB82E89C\

Dropped payloads: MonXXXXXXXXX.exe (randomised names)

Fake system process: C:\Windows\rss\csrss.exe

Persistence: RaptorMiner.exe (Run key) ; Driver.url (Startup folder) ; scheduled tasks named “Schedule.Service.1” ;
%APPDATA%\Roaming\teieedr ; %APPDATA%\Roaming\Sysfiles\Driver.exe

Services installed: VBoxGuest, VBoxMouse, VBoxSF, VBoxService, VBoxVideo, VBoxWddm

Local listeners: TCP 31461, TCP 49703

Mutexes: Global\h48yorbq6rm87zot ; Global\qtxp9g8w ; kEecfMwgjiZ5i-O1fR-8gT0

Malformed WinHTTP User-Agent: The intended “Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36” transmitted with corrupted leading bytes, rendered as non-Latin (CJK) code points in memory and as a single control byte (0x02) on the wire.

“STANDOFF COORD” console; browser storage keys standoff.token and standoff.workspace.
217.198.13[.]211:8002 (Telegram account-farm API) and the “GG Influence” / “Мобильная Арена” services on the same host;
browser storage key auth-storage.

5. Abbreviations

API Application Programming Interface

ASN Autonomous System Number

C2 Command-and-Control
CIB Coordinated Inauthentic Behaviour
CIDR Classless Inter-Domain Routing (an IP address-range notation)
DMZ Demilitarised Zone (a perimeter network segment)
IMAP Internet Message Access Protocol (e-mail retrieval)
IOC Indicator of Compromise
JARM An active TLS server fingerprinting method
JWT JSON Web Token
NSIS Nullsoft Scriptable Install System (an installer format)
NTLM NT LAN Manager (Windows authentication / password hashes)
OAuth Open Authorisation (a delegated-access standard)
PPI Pay-Per-Install (a malware-distribution model)
RBAC Role-Based Access Control
SSH Secure Shell
SSO Single Sign-On
TLS Transport Layer Security
TTP Tactics, Techniques and Procedures
WMI Windows Management Instrumentation

6. REFERENCES

[1] abuse.ch ThreatFox. "Raccoon Stealer botnet C2, Telegram dead-drop resolver t[.]me/borderxra." IOC ID 1676437, IOC type: URL, threat type: botnet_cc, confidence: moderate (50%). First seen: 2025-12-12. Reporter: jurroots. Available: <https://threatfox.abuse.ch/ioc/1676437/>.

[2] abuse.ch ThreatFox. "Raccoon Stealer botnet C2, Telegram dead-drop resolver t[.]me/jredmankun." IOC ID 1676438, IOC type: URL, threat type: botnet_cc, confidence: moderate (50%). First seen: 2025-12-12. Reporter: jurroots. Available: <https://threatfox.abuse.ch/ioc/1676438/>.