

CTI Report: Ninite Pro Remote Agent Leveraged for Persistence

barricadecyber.com/cti-report-ninite-pro-remote-agent-leveraged-for-persistence

Eric J. Taylor

June 30, 2026



A genuine login.microsoftonline.com link — no typosquat — is abused as an open redirect to deliver a malicious MSI that installs the Ninite Pro Remote agent as a SYSTEM service, which then deploys TeamViewer and AnyDesk for persistent remote control. Here is exactly how the redirect trick works.

ACTIVE THREAT — OBSERVED IN THE WILD

- **Platform:** Microsoft Windows
- **Initial access:** Phishing link using a genuine login.microsoftonline.com URL as an open redirect — no typosquat.
- **Impact:** Persistent, SYSTEM-level remote control via a rogue Ninite Remote agent, followed by TeamViewer and AnyDesk for redundant hands-on access.
- **Act now:** Hunt the indicators below; treat any unsanctioned Ninite, TeamViewer, or AnyDesk install as an incident.

Executive summary

Barricade Cyber's threat research team analyzed a malware delivery chain that abuses a **legitimate Microsoft sign-in URL** as a redirector to deliver a malicious Windows Installer (MSI) package. The MSI does not drop a conventional backdoor. Instead, it silently installs the **Ninite Pro "Remote" agent** — a legitimate IT patch-management tool — and enrolls the victim machine into an *attacker-controlled* Ninite Remote account, handing the operator persistent, SYSTEM-level remote-management and code-execution capability over the host. The operator then uses that foothold to install two commodity remote-access tools — **TeamViewer and AnyDesk** — for redundant, hands-on access.

The most important takeaway for defenders: **the phishing link is a real login.microsoftonline.com URL, not a typosquat.** There is no misspelling to catch. The attacker uses Microsoft's own OAuth endpoint as an open redirector, so the malicious destination never appears in the link a victim clicks or that a URL filter inspects.

Two halves of the attack:

1. **Delivery** — A genuine Microsoft OAuth authorize URL redirects the victim to an attacker-hosted MSI on consumer cloud storage.
2. **Persistence** — The MSI installs the Ninite Remote agent as a Windows service under SYSTEM, enrolled to the attacker's tenant.
3. **Second-stage access** — Through the rogue Ninite agent, the operator installs TeamViewer and AnyDesk, giving them redundant interactive remote control of the host.

The delivery trick, explained in plain language

This is the part that confuses most people on first read, so we are going to take it slowly.

It is the *real* Microsoft domain

The link the victim receives points to:

https://login.microsoftonline.com/common/oauth2/v2.0/authorize?state=ut&scope=openid&prompt=none&client_id=17c4bab1-4866-4277-8408-4600c155acfd

Read the host carefully: login.microsoftonline.com. That is Microsoft's genuine Entra ID (Azure AD) sign-in endpoint, with a valid Microsoft TLS certificate. It is **not** a look-alike domain like login.micros0ft.com or microsoftonline-login.com. There is no spelling trick. Standard end-user advice — "hover over the link and check the domain" — fails here, because the domain is exactly what it should be.

So how does a legitimate Microsoft URL end up handing the victim a malware download? Through how OAuth sign-in is designed to work.

The key idea: the link only carries an *ID*, not the destination

An OAuth sign-in request identifies an application by its `client_id` — a GUID. In this case:

```
client_id = 17c4bab1-4866-4277-8408-4600c155acfd
```

A `client_id` is just a **reference number**. It tells Microsoft *which* registered application is asking. It does **not**, by itself, contain any redirect destination.

The redirect destination lives in a **separate setting** called the `redirect_uri`, which is configured *on the application registration* and stored on Microsoft's side. Crucially, in this attack the `redirect_uri` is **not present in the URL at all**. When an OAuth request omits it, Microsoft falls back to the redirect URI registered on that application.

The attacker did two cheap things:

1. Registered their own application in a throwaway Entra tenant (anyone can, in minutes), producing the `client_id` above.
2. Set that application's `redirect_uri` to their **payload URL** — an MSI hosted on OVH cloud object storage.

So the malicious destination is attached to the application *behind* the `client_id`. The victim never sees it in the link.

The `prompt=none` switch forces the redirect to fire

The last ingredient is the parameter `prompt=none`. It tells Microsoft: *“Attempt single sign-on silently. Do not show any login screen or prompt.”*

When the victim has no active Microsoft session — which is the normal case when someone clicks a link out of an email, and is always the case in a sandbox — Microsoft **cannot** complete a silent sign-on. Per the OAuth standard, instead of showing a login page, it returns an **error**, and it returns that error by **redirecting the browser to the application's registered `redirect_uri`**.

That registered redirect URI is the attacker's MSI. So Microsoft itself issues the redirect to the malware.

Watching it happen

We observed exactly this in the sandbox. Microsoft redirected the browser to:

```
https://construction-quotation.s3.rbx.io.cloud.ovh.net/quotation_qu26022202.msi
?error=interaction_required
&error_description=Session+information+is+not+sufficient+for+single-sign-on.
&state=ut
```

Two details confirm the mechanism:

- **`error=interaction_required`** is the precise error Microsoft returns when `prompt=none` cannot complete silently. Its presence is the fingerprint of this technique.
- Microsoft will only redirect to a `redirect_uri` that is **pre-registered** on the application. The fact that the browser landed on the OVH MSI URL is itself proof that the attacker registered that URL as the application's redirect target.

The cloud storage server ignores the `?error=...` query string appended by Microsoft and simply serves the file. The victim's browser downloads `quotation_qu26022202.msi` — a fake “construction quotation.”

The flow, end to end

```
Victim clicks a clean-looking Microsoft link
      |
      v
login.microsoftonline.com/.../authorize?client_id=17c4bab1...&prompt=none
      |
      v
```

```
Microsoft looks up client_id 17c4bab1... -> finds the attacker's app
      |                                     (its redirect_uri = the OVH .msi)
      v
prompt=none + no active session -> silent SSO fails
      |
      v
Microsoft returns the error by REDIRECTING (302) to the app's redirect_uri
      |
      v
https://construction-quotation...ovh.net/quotation_qu26022202.msi?error=interaction_required...
      |
      v
Browser downloads the malicious MSI (victim only ever "visited" Microsoft)
```

Why this beats common defenses

- **It passes the eyeball test.** The link is a real Microsoft URL with a real certificate.
- **It bypasses allowlists.** Email gateways and web proxies that trust or allowlist `login.microsoftonline.com` pass the link straight through.
- **Static link inspection sees nothing bad.** The malicious destination is supplied by Microsoft *after* the click, from the application registration. It is not in the message or the URL, so pre-delivery scanners cannot extract it.
- **It usually leaves no trace in the victim's own tenant.** The malicious application lives in the *attacker's* Entra tenant, and the `prompt=none` request fails immediately without authenticating the victim. The victim organization's Entra sign-in logs typically show nothing. Detection has to happen at the **network and endpoint** layer, not in identity logs.

To be clear about what this technique is and is not: this is **open-redirect abuse for malware delivery**. It is *not* an illicit OAuth consent-grant attack — the victim is never asked to consent, and no token is issued or stolen. We are deliberately not mapping it to token-theft techniques, because that is not what happened.

The payload: a signed MSI that installs Ninite

The downloaded `quotation_qu26022202.msi` is executed by the Windows Installer service (`msiexec.exe`). Inside the MSI, a **custom-action DLL** drives the behavior. The MSI then runs Ninite:

- `msiexec.exe` launched **Ninite.exe** (running as **SYSTEM**, digitally signed by “*Secure By Design Inc.*” — Ninite's legitimate publisher) with a silent-install command line out of a temporary directory.
- Ninite.exe installed **NiniteAgent.exe** as a **Windows service** running under SYSTEM (observed across multiple process instances).

The Windows service is the persistence mechanism. The agent then “phoned home” to Ninite's Remote control plane and enrolled the host into the operator's account.

Why an attacker wants Ninite

Ninite Pro's “Remote” feature is a legitimate remote software-management product: install an agent on machines, and from a web console push application installs/updates and run actions across the fleet. In the hands of an attacker, that is:

- **Persistence** — an auto-starting SYSTEM service that survives reboots.
- **Remote code execution on demand** — the latent ability to install or run software later, from a legitimate-looking console.
- **Blends in** — signed by a legitimate vendor, talking to legitimate Ninite domains. This is the same “abuse a legitimate RMM tool” pattern seen with AnyDesk, Atera, and ScreenConnect, applied to a less-watched product.

In this case the agent enrolled as Ninite Remote **machine ID 7240347**.

The second stage: Ninite installs TeamViewer and AnyDesk

Ninite is the delivery mechanism, not the end goal. Using the remote-management capability it just established, the operator pushed two well-known remote-access tools onto the host: **TeamViewer** and **AnyDesk**. Installing two independent remote-access

products is a deliberate redundancy play — if one is uninstalled, blocked, or flagged, the other keeps the operator's hands-on access alive.

A note on how this surfaced, because it matters for analysts reading automated verdicts. The *public* sandbox report for the original sample carries the tags `teamviewer`, `rmm-tool`, and `stealer`, but its capture window showed only the Ninite agent enrolling and taking a software inventory — no second-stage installer had run yet within that window. It would have been easy to dismiss the `teamviewer` tag as a reconnaissance artifact. **A longer, hands-on detonation by Barricade's team confirmed the actual outcome: the rogue Ninite agent went on to install both TeamViewer and AnyDesk.** The large batch of `remote1.ninite.com/remote/user_apps/<id>/<hash>` requests seen in the original report — easily mistaken for routine catalog sync — is consistent with staging those application payloads for deployment.

The lesson: a sandbox verdict is a snapshot of a finite window. When the foothold is an RMM agent, the most damaging actions (pushing follow-on tooling) can happen *after* the automated capture ends. Treat the agent itself as the incident and assume follow-on deployment.

On the `stealer` tag specifically: we still treat that classification as a heuristic. The dominant, confirmed objective here is **interactive remote access via TeamViewer and AnyDesk**, layered on the Ninite foothold. We are not asserting a dedicated info-stealer payload absent evidence of credential or file exfiltration.

The bigger picture: Ninite (Nintune) can attach custom scripts to any deployment

What makes a hijacked Ninite Pro foothold especially dangerous is the platform's own feature set. Ninite's **Nintune** offering lets an administrator attach custom **pre-install, post-install, or uninstall scripts** to any application deployment as PowerShell (`.ps1`), batch (`.cmd`), or command (`.bat`) files. In other words, the same management channel used to push software can also run arbitrary code on the endpoint.

 Ninite Nintune Scripts panel: Post-Install Script Type set to PowerShell (`.ps1`) with a custom post-install script

Source: [Ninite \(ninite.com/nintune\)](https://ninite.com/nintune) — the Nintune interface for attaching custom pre/post-install scripts to a deployment package.

“You can add pre- and post-install or uninstall scripts to any app as a `.ps1`, `.cmd`, or `.bat` file. Want to preconfigure something for users? No problem. Delete some data after an app is gone?”

— *Ninite, describing the Nintune scripting feature*

Disclaimer: While Nintune allows custom scripts to be deployed alongside any software installation, **there is currently no evidence that the installer analyzed in this report is using that feature.** The scripting capability is documented here to illustrate the potential risk of an abused Ninite Pro account — not to assert it was exercised in this case.

MITRE ATT&CK mapping

Tactic	Technique	Detail
Initial Access	T1566.002 — Phishing: Spearphishing Link	Link to a real Microsoft OAuth URL used as a redirector
Execution	T1204.002 — User Execution: Malicious File	Victim runs the downloaded MSI
Execution / Defense Evasion	T1218.007 — System Binary Proxy Execution: Msiexec	<code>msiexec</code> executes the MSI custom-action DLL and launches Ninite
Persistence	T1543.003 — Create or Modify System Process: Windows Service	<code>NiniteAgent.exe</code> installed as a <code>SYSTEM</code> service

Tactic	Technique	Detail
Command & Control	T1219 — Remote Access Software	Ninite Remote agent enrolled to attacker account as RMM/C2; used to deploy TeamViewer and AnyDesk for redundant interactive access
Discovery	T1518 / T1518.001 — Software & Security Software Discovery	Agent enumerates installed software and antivirus presence

Note: the delivery trick (abusing a trusted vendor's OAuth endpoint as an open redirector) does not have a clean dedicated ATT&CK technique ID; it is captured here under the spearphishing-link initial-access vector.

Detection and hunting guidance

Because the identity layer is largely blind to this, focus on network and endpoint telemetry.

Network / proxy

- Hunt for web redirect chains that **originate at login.microsoftonline.com/.../authorize** carrying prompt=none and **terminate in a file download** (.msi, .exe, .zip) from non-Microsoft infrastructure.
- Flag the OAuth error fingerprint error=interaction_required immediately preceding a download from consumer cloud object storage.
- Alert on executable/MSI downloads from consumer cloud storage such as *.io.cloud.ovh.net, S3-style buckets, and similar, where not expected.

Endpoint (EDR)

- msixexec.exe spawning Ninite.exe from a user ... \AppData\Local\Temp\<GUID>\ path.
- NiniteAgent.exe registered/running as a Windows service under SYSTEM on hosts where Ninite is not a sanctioned IT tool.
- Outbound connections to remotel.ninite.com, remote.ninite.com, and lp1.ninite.com from non-IT endpoints.
- **Second-stage remote-access tooling installed by the Ninite agent.** Hunt for TeamViewer.exe / TeamViewer_Service.exe (%ProgramFiles(x86)%\TeamViewer\, key HKLM\SOFTWARE\WOW6432Node\TeamViewer) and AnyDesk.exe (%ProgramFiles(x86)%\AnyDesk\ or %APPDATA%\AnyDesk\, AnyDesk service) appearing on a host shortly after a Ninite agent install, especially when configured for unattended access. Either tool arriving with a Ninite-agent parent, or two remote-access products appearing together, is a strong signal.

Policy / RMM governance

Treat **unsanctioned RMM agents as incidents**. If Ninite Pro Remote is not an approved tool in your environment, *any* Ninite Remote agent traffic warrants investigation. Maintain an explicit allowlist of approved remote-management tools and alert/block on everything else.

Recommendations

- **Do not blindly trust Microsoft redirect chains.** Inspect the *final* download destination, not just the first hop. Consider proxy rules that break or flag login.microsoftonline.com redirects ending in file downloads from third-party hosts.
- **Block executable and MSI downloads from consumer cloud object storage** at the proxy where business needs allow.
- **Enforce application control** (WDAC / AppLocker) to prevent execution of unsigned or unsanctioned MSIs and RMM agents.
- **Inventory and govern remote-management software.** Alert on the install or execution of any RMM agent not on your approved list.
- **User awareness:** reinforce that a link can point to a real Microsoft domain and still be malicious. The defense is not “check the spelling” — it is “do not run unexpected files, including documents/quotes you did not request.”

Indicators of compromise

All hashes below are **derived from a public sandbox detonation** and are provided as “observed in analysis,” not as multi-vendor-vetted indicators. The Ninite enrollment identifiers (machine ID, agent key/ID, tokens) are **specific to this incident** and are useful for scoping this activity, not as durable network indicators.

Indicator	Type	Context
login.microsoftonline.com/common/oauth2/v2.0/authorize?...client_id=17c4bab1-4866-4277-8408-4600c155acfd	URL (redirector)	Legitimate Microsoft endpoint abused as open redirect; attacker app client_id
17c4bab1-4866-4277-8408-4600c155acfd	Entra app client_id	Attacker-registered OAuth application (not a known Microsoft first-party app)
construction-quotation.s3.rbx.io.cloud.ovh.net	Domain	Payload host (OVH cloud object storage, Roubaix)
https://construction-quotation.s3.rbx.io.cloud.ovh.net/quotation_qu26022202.msi	URL	Malicious MSI download
quotation_qu26022202.msi	Filename	Lure (“construction quotation”)
88174718adac0d95c5c2d87eab47b63d8b135dcec6c77b5962d9cc2c1433204e	SHA256	Malicious MSI (Ninite Agent installer); MD5 5aa904eb96c13f55b43a11ee0f2e4432
dbda294c39231d7b33cfcfd533fd4af4615628a48b4856000d1d5d0df8806d160	SHA256	MSI custom-action DLL (execution mechanism)
remote1.ninite.com, remote.ninite.com, lp1.ninite.com	Domains	Ninite Remote control plane (legitimate infrastructure; malicious use)
Ninite Remote machine ID 7240347	Identifier	Victim enrollment in attacker’s Ninite account
agent!90e58d00105a2e1b4dc4d4a7dea86a60cfe02972	Ninite account key	Operator’s Ninite Pro account key, embedded in the MSI — durable attribution (same across this operator’s victims until rotated). Reportable to Ninite for takedown.
E3AF290D2B5B084B3895A4BB8C57EF7BE5041B16	Agent ID	Ninite agent enrollment ID (per-incident)
17Y5OG37EOB2j94fGwbxnh3grn_e-DcUlpL9SySMs34	Machine token	Ninite Remote machine token (per-incident)
dc821fbd5ea68250c4d35f0f84682714b64ee459180a20d833d85d61d4cd2e00	Config hash	Ninite agent_config stage identifier
TeamViewer (TeamViewer.exe, TeamViewer_Service.exe)	Remote-access tool	Second-stage tool installed via the Ninite agent (incident-specific TeamViewer ID pending capture)

Indicator	Type	Context
AnyDesk (AnyDesk.exe)	Remote-access tool	Second-stage tool installed via the Ninite agent (incident-specific AnyDesk ID pending capture)

Need help responding to RMM abuse or OAuth-redirect phishing? Book time with Barricade: