Hacks for sale: inside the Buer Loader malware-as-a-service

news.sophos.com/en-us/2020/10/28/hacks-for-sale-inside-the-buer-loader-malware-as-a-service/

Sean Gallagher October 28, 2020

During our investigation of a Ryuk attack in September 2020, we found the Ryuk actors had used a relatively new method for gaining initial access: a malware dropper called Buer. The September attack was part of a low-volume spear phishing attack tracked by Sophos. Over the next month, it evolved into a much larger spam campaign, carrying Buer as well as a number of other types of "loader" malware, as the Ryuk operators sought to ramp up their attacks.

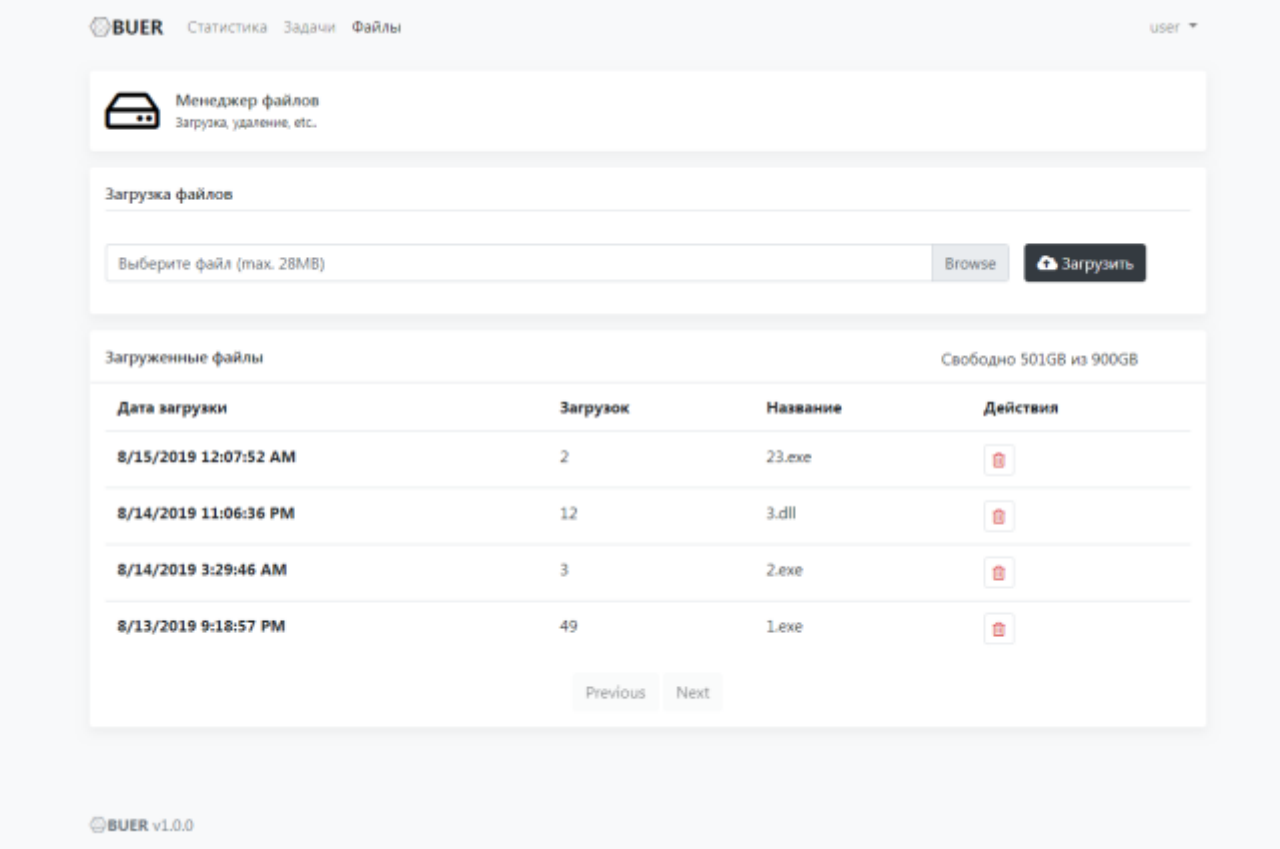
First introduced in August of 2019, Buer is a malware-as-a-service offering that is used to deliver whatever package the service customer desires, providing initial compromise of targets' Windows PCs and allowing them to establish a digital beachhead for further malicious activity. Buer has previously been tied to banking trojan attacks and other malware deployments—and now, apparently, has been embraced by ransomware operators. In many ways, Buer is positioned as an alternative to Emotet and Trickbot's emerging Bazar loader (which both use similar behaviors to deploy).

Full-service bots

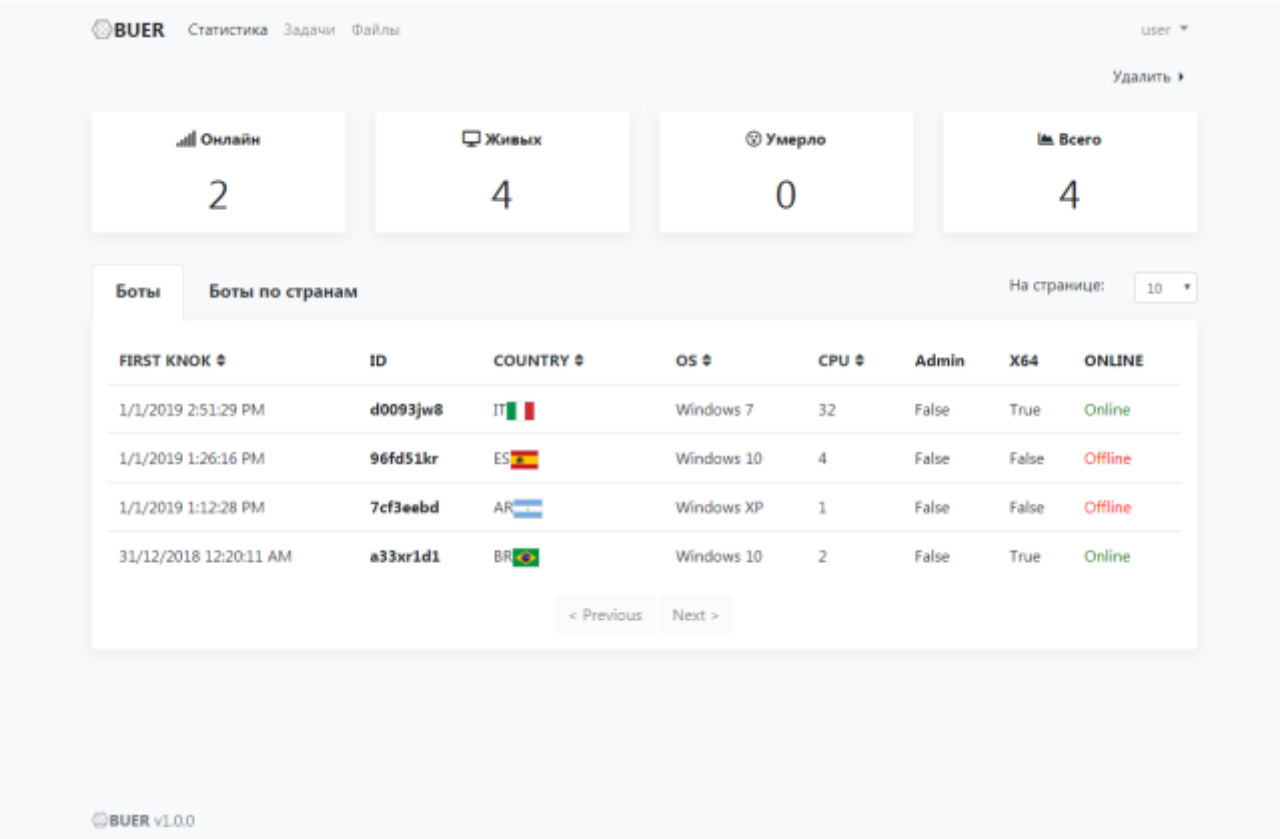
Buer was first advertised in a forum post on August 20, 2019 under the title “Modular Buer Loader”, described by its developers as “a new modular bot…written in pure C” with command and control (C&C) server code written in .NET Core MVC (which can be run on Linux servers). For $350 (plus whatever fee a third-party guarantor takes), a cybercriminal can buy a custom loader and access to the C&C panel from a single IP address—with a $25 charge to change that address. Buer’s developers limit users to two addresses per account.

The bot code, compiled for each user specific to a download, has an advertised size between 22 and 26 kilobytes—though the sample we looked at was about 40 kilobytes after being unpacked from its dropper. The bot can be configured for execution either as a 32-bit Windows executable or as a DLL.

The C&C can be used to track the number of successful downloads in a campaign, and to assign tasks to bots by filters such as the country they’re in, the “bitness of the operating system” (32 or 64 bit), the number of processors on the infected machine and the level of permissions obtained by the bot. Bots detected to be operating within the Commonwealth of Independent States will be shut down—which is a common behavior of malware developed in the ex-USSR region, as an attempt to avoid attention from local authorities.



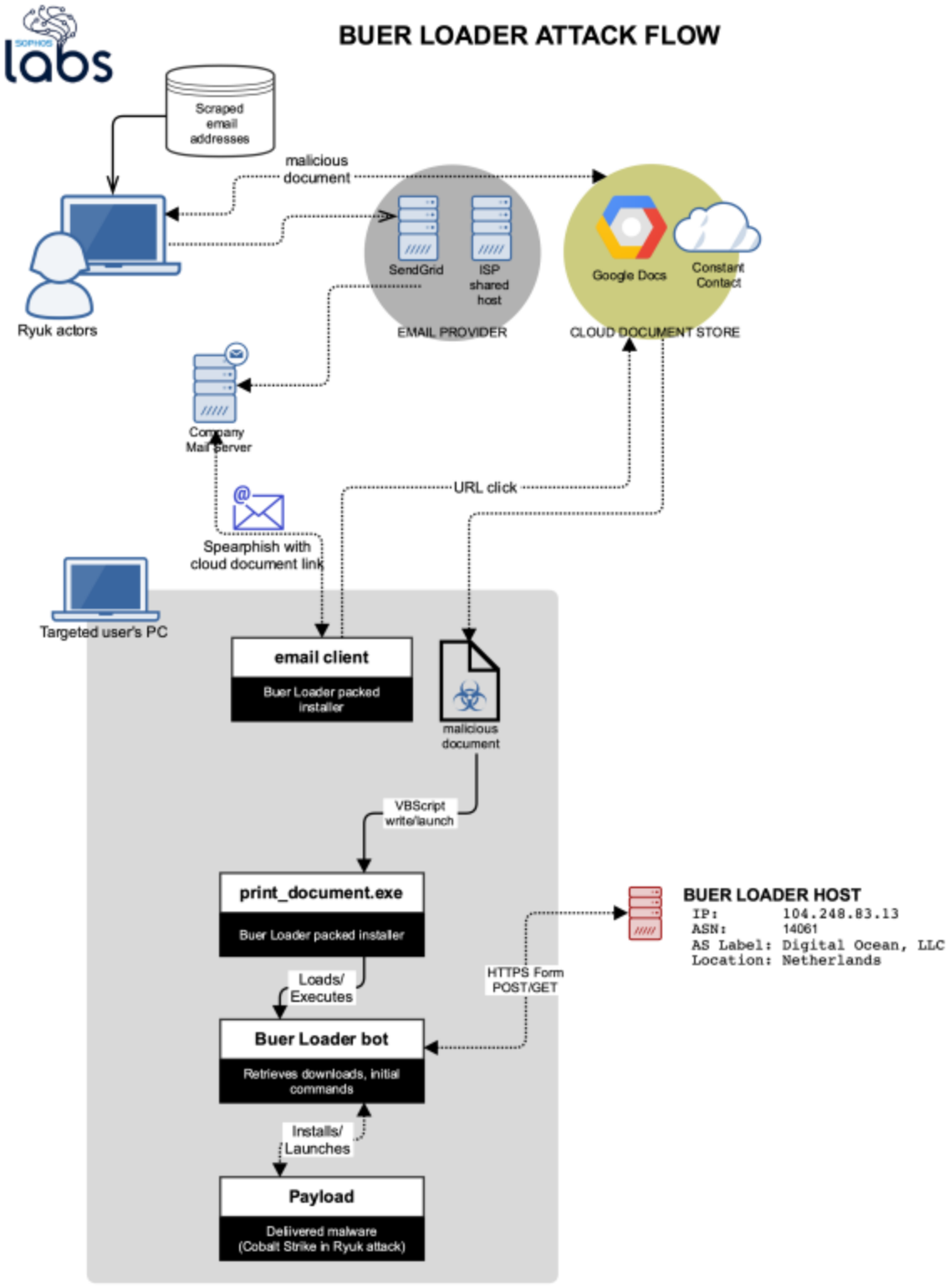
The “file manager in the command and control “panel” for the Buer loader bot. Files can be uploaded for distribution here—the maximum size is 28 megabytes.



The Buer panel tracks installations by operating system, CPU, “bitness” (32 or 64), activity, and which geographic region they’re in based on localization settings and other fingerprinting.

Tasks can be scheduled to run for a specific amount of time, or suspended upon command, with telemetry for the task sent back to the panel. The panel can also be used to deploy updates to bots, including (at least based on the advertisement) deployment of modules, with prebuilt modules to be added “over time” as part of the service. And of course, setup consulting and technical support are provided.

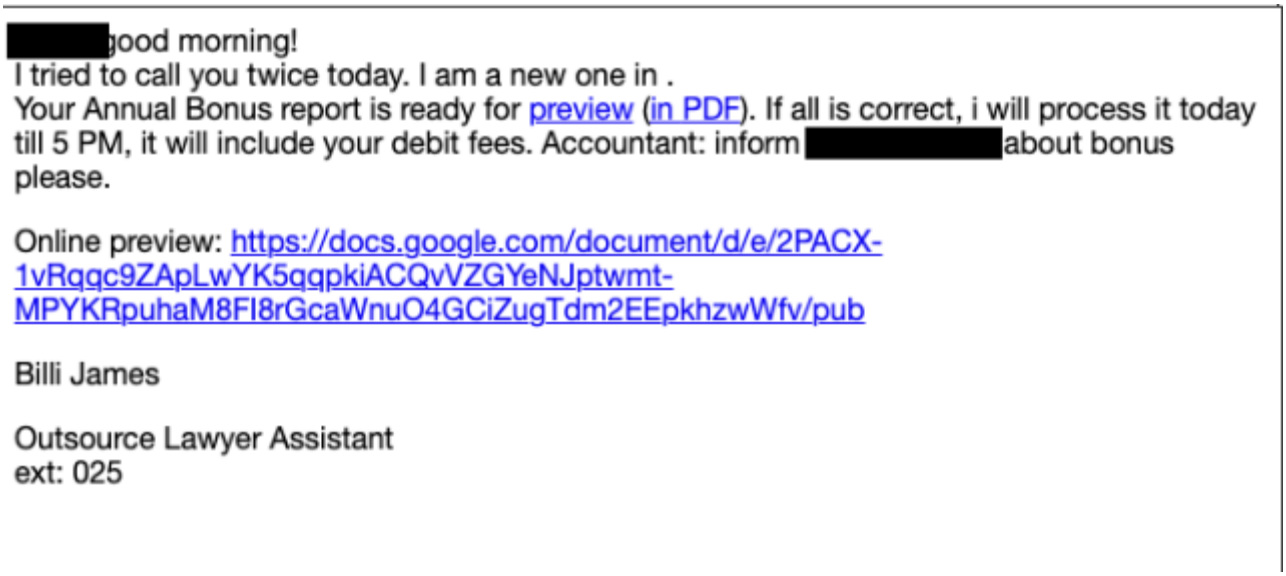
Prize inside every doc



Buer loader attack flow.

Sophos' Rapid Response team discovered a sample of Buer at the root cause of a September Ryuk attack. The loader was delivered by a malicious document stored on Google Docs, which required the victim to enable scripted content to activate—a behavior similar to Emotet and other loader attacks via malicious spam emails but leveraging cloud storage to make forensic analysis more difficult.

We collected other messages from the same campaign in Sophos' spam traps during the same period. The messages all used Google Docs files, and were sent using a popular commercial email distribution service—further obscuring the source and the link associated with the malicious document.



[redacted]good morning!
I tried to call you twice today. I am a new one in .
Your Annual Bonus report is ready for preview (in PDF). If all is correct, i will process it today till 5 PM, it will include your debit fees. Accountant: inform [redacted] about bonus please.

Online preview: https://docs.google.com/document/d/e/2PACX-1vRqqc9ZApLwYK5qqpkiACQvVZGYeNJptwmt-MPYKRpuhaM8FI8rGcaWnuO4GCiZugTdm2EEpkhzwWfv/pub

Billi James

Outsource Lawyer Assistant
ext: 025

An example of the initial run of Buer bot distributing spear phishes.
The payload of that malicious document was named print_document.exe. Like other Buer dropper samples we've analyzed, it was a digitally signed binary, using a stolen and now-revoked certificate issued by DigiCert to "NEEDCODE SP Z O O," a Polish software developer, issued on September 17, 2020. The dropper was built using modified code from a Microsoft sample application for image capture, AcquireTest, using the code's function for "file enumeration" to delete and drop code.

The dropper does a number of things to ensure proper delivery. It first checks for the presence of a debugger to evade forensic analysis, and then checks language and localization settings to determine the geographic region of the system being attacked. If the settings match a CIS country, it will exit without depositing the malware. Otherwise, the dropper then dumps the Buer bot in memory and executes it.

Intriguingly, the Buer Loader and Ryuk ransomware uses same shellcode loader to execute the unpacked malware code in memory:

```
seg000:0000002D ; ---------------------------------------------------------------------------
seg000:0000002D
seg000:0000002D loc_2D:                                 ; CODE XREF: seg000:00000024↑p
seg000:0000002D                 sub     esp, 48h
seg000:00000030
seg000:00000030 loc_30:                                 ; DATA XREF: Calc_Hash_func+3↓r
seg000:00000030                                         ; sub_95E+3DD↓r ...
seg000:00000030                 and     dword ptr [esp+18h], 0
seg000:00000035                 mov     ecx, 726774Ch   ; DATA XREF: seg000:0000B1D4↓o
seg000:00000035                                         ; seg000:0000B0CC↓o ...
seg000:0000003A                 push    ebx
seg000:0000003B                 push    ebp
seg000:0000003C                 push    esi
seg000:0000003D                 push    edi
seg000:0000003E                 xor     esi, esi
seg000:00000040                 call    Calc_Hash_func
seg000:00000045                 mov     ecx, 7802F749h  ; GetProcAddress hash
seg000:0000004A                 mov     [esp+1Ch], eax
seg000:0000004E                 call    Calc_Hash_func
seg000:00000053                 mov     ecx, 0E553A458h ; VirtualAlloc hash
seg000:00000058                 mov     [esp+20h], eax
seg000:0000005C                 call    Calc_Hash_func
seg000:00000061                 mov     ecx, 0C38AE110h ; VirtualProtect hash
seg000:00000066                 mov     ebp, eax
seg000:00000068
seg000:00000068 loc_68:                                 ; DATA XREF: seg000:0000B678↓o
seg000:00000068                 call    Calc_Hash_func
seg000:0000006D                 mov     ecx, 945CB1AFh  ; NtFlushInstructionCache hash
seg000:00000072                 mov     [esp+2Ch], eax
seg000:00000076                 call    Calc_Hash_func
seg000:0000007B                 mov     ecx, 959E0033h  ; GetNativeSystemInfo hash
seg000:00000080                 mov     [esp+30h], eax
seg000:00000084                 call    Calc_Hash_func
seg000:00000089                 mov     ebx, eax
seg000:0000008B                 mov     eax, [esp+5Ch]
seg000:0000008F                 mov     edi, [eax+3Ch]
seg000:00000092                 add     edi, eax
seg000:00000094                 mov     [esp+10h], edi
seg000:00000098                 cmp     dword ptr [edi], 4550h
seg000:0000009E                 jz      short loc_A7
seg000:000000A0
seg000:000000A0 loc_A0:                                 ; CODE XREF: seg000:000000B0↓j
```

Buer loader shell code…

```
seg000:0000002D ; ---------------------------------------------------------------------------
seg000:0000002D
seg000:0000002D loc_2D:                                 ; CODE XREF: seg000:00000024↑p
seg000:0000002D                 sub     esp, 48h
seg000:00000030
seg000:00000030 loc_30:                                 ; DATA XREF: Calc_Hash_func+3↓r
seg000:00000030                                         ; sub_95E+3DD↓r ...
seg000:00000030                 and     dword ptr [esp+18h], 0
seg000:00000035                 mov     ecx, 726774Ch
seg000:0000003A                 push    ebx
seg000:0000003B                 push    ebp
seg000:0000003C                 push    esi
seg000:0000003D                 push    edi
seg000:0000003E                 xor     esi, esi
seg000:00000040                 call    Calc_Hash_func
seg000:00000045                 mov     ecx, 7802F749h  ; GetProcAddress hash
seg000:0000004A                 mov     [esp+1Ch], eax
seg000:0000004E                 call    Calc_Hash_func
seg000:00000053                 mov     ecx, 0E553A458h ; VirtuAlloc hash
seg000:00000058                 mov     [esp+20h], eax
seg000:0000005C                 call    Calc_Hash_func
seg000:00000061                 mov     ecx, 0C38AE110h ; VirtualProtect hash
seg000:00000066                 mov     ebp, eax
seg000:00000068                 call    Calc_Hash_func
seg000:0000006D                 mov     ecx, 945CB1AFh  ; NtFlushInstructionCache hash
seg000:00000072                 mov     [esp+2Ch], eax
seg000:00000076                 call    Calc_Hash_func
seg000:0000007B                 mov     ecx, 959E0033h  ; GetNativeSystemInfo hash
seg000:00000080                 mov     [esp+30h], eax
seg000:00000084                 call    Calc_Hash_func
seg000:00000089                 mov     ebx, eax
seg000:0000008B                 mov     eax, [esp+5Ch]
seg000:0000008F                 mov     edi, [eax+3Ch]
seg000:00000092                 add     edi, eax
seg000:00000094                 mov     [esp+10h], edi
seg000:00000098                 cmp     dword ptr [edi], 4550h
seg000:0000009E                 jz      short loc_A7
seg000:000000A0
seg000:000000A0 loc_A0:                                 ; CODE XREF: seg000:000000B0↓j
seg000:000000A0                                         ; seg000:000000B6↓j ...
seg000:000000A0                 xor     eax, eax
seg000:000000A2                 jmp     loc_44E
```

and Ryuk loader shell code.
This may not be an indication of shared authorship; the developers may have simply used the same sample code as their source.

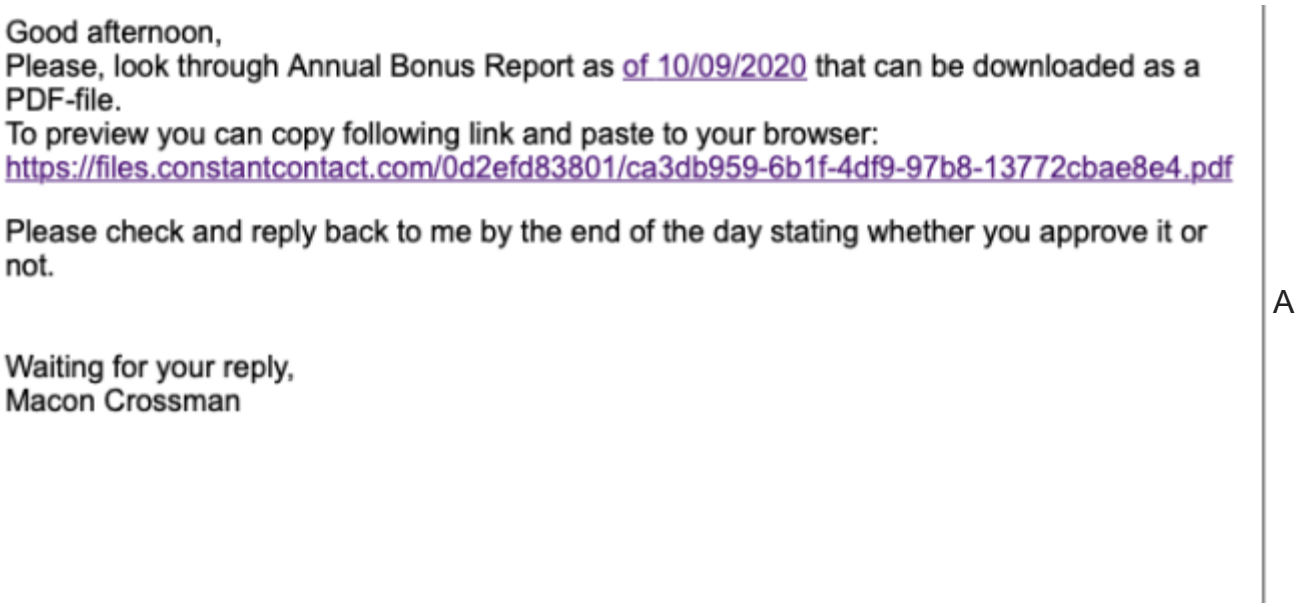
Upon launch, the Buer bot does a number of things to set up shop. The bot executes two sets of PowerShell commands—one to bypass execution policies to allow PowerShell commands executed by the bot to go through without warnings (Set-ExecutionPolicy Bypass), and another (add-mppreference -exclusionpath) to make changes to Windows Defender's exclusion list—concealing files it downloads from Windows' built-in malware protection.

Buer queries the Windows Registry for the value of \Microsoft\Cryptography\MachineGuid to get the unique identifier for the infected machine. And the bot calls home, interacting with the command and control server (in this case, 104[.]248.83.13) through a series of secure HTTP "POST" and "GET" messages.

Then there's the "loader" part of what Buer does. The files packaged to be dropped by Buer are retrieved from a designated source and dropped in a folder created in the C:\ProgramData\ directory—the directory name is created programmatically and varies with deployments. In the September attack, Buer was used to deploy a Cobalt Strike beacon to the infected computer, which was then in turn used to exploit the network and launch a Ryuk attack.

Mixing it up

The malicious spam campaign that resulted in the Buer loader and Ryuk ransomware infections evolved at the end of September, as we observed the actors behind it shift the same tactics away from low volume on SendGrid to mail sent through Internet hosting providers—predominantly through a single Russian ISP. Then in October, the volume of spam rose dramatically—shifting away from Google Docs (as Google shut down the old files for terms of service violations) to another commercial email and file delivery service.


Good afternoon,
Please, look through Annual Bonus Report as of 10/09/2020 that can be downloaded as a PDF-file.
To preview you can copy following link and paste to your browser:
https://files.constantcontact.com/0d2efd83801/ca3db959-6b1f-4df9-97b8-13772cbae8e4.pdf

Please check and reply back to me by the end of the day stating whether you approve it or not.

Waiting for your reply,
Macon Crossman

A somewhat less targeted spam message with a link to a malicious document stored by Constant Contact.

In the last two phases, while the tactics remained similar and other hallmarks suggested the spam actor was the same, multiple types of "dropper" malware were deployed as attachments. In addition to Buer, samples of Bazar and ZLoader were also found, with delivery payloads varying. For one Bazar loader payload, the attackers used a password-protected Excel spreadsheet. During the same timeframe, Bazar and ZLoader were also known to be involved in Ryuk attacks.

It's clear that Ryuk is back, and that the actors behind it are evolving their methods for initial compromise, using multiple loader bots to achieve initial access. It's not clear if the same actor is behind all of these attacks, using multiple malware-as-a-service platforms to deliver Ryuk, or if there are multiple Ryuk actors. But the similarity in techniques across these

campaigns suggests that there is at least coordination between them: they use targeted emails with cloud-based malicious documents and a lure to spur immediate action (often related to wages or taxes).

The best mitigation for these attacks is to reinforce training on phishing attacks. While these malicious emails are targeted, they are usually awkwardly worded and use the target's name in odd ways. Careful reading of the email will tip off most educated users. But these attacks are growing in sophistication, and even well-trained users may eventually click on the wrong link in an email if spam detection doesn't catch them first.

Sophos detects and blocks Buer both with custom detections (Troj/BuerLd-A) and machine learning, and detects the spear phishing messages as spam. Indicators of compromise associated with Buer Loader can be found on SophosLabs' GitHub.

Sophos would like to acknowledge the contributions of Peter Mackenzie, Elida Leite, Syed Shahram and Bill Kearny of the Sophos Rapid Response team, and Anand Ajjan, Brett Cove and Gabor Szappanos of SophosLabs for their contributions to this report

---