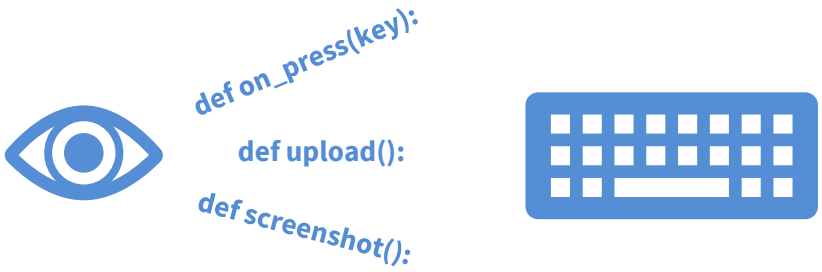

Erstellung einer Keylogger-Schadsoftware mit Python

Hausarbeit im Rahmen der Vorlesung „Hacking mit Python“

Einrichtung	Hochschule Albstadt-Sigmaringen
Vorlesung	Hacking mit Python – WS2022/2023
Dozent	Prof. Martin Rieger
Autor	Karsten Blank
Studiengang	TIB
Matrikel-Nr.	90294

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne unzulässige fremde Hilfe verfasst habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt oder veröffentlicht. Ich erkläre mich damit einverstanden, dass die Arbeit mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate überprüft werden kann.

München, 30.12.2022 

Datum Unterschrift

Zusammenfassung

Diese Arbeit befasst sich mit der Erstellung und Bereitstellung einer Keylogger-Schadsoftware in der Programmiersprache Python zu akademischen Zwecken. Die Software umfasst die folgende Grund-Funktionen: Erfassung von Tastatureingaben, Erfassung von Mausklicks, Erfassung der Zwischenablage sowie Erzeugung von Screenshots um Tastatureingaben und Mausklicks entsprechenden Programmen zuordnen zu können. Die Schadsoftware wird auf dem „Opfer"-Rechner innerhalb eines manipulierten Python-Paketes mittels „pip install" eingeschleust. Das eigentliche Executable wird dabei im Rahmen der „setup.py" von einem FTP-Server („Angreifer") per FTPS zur Verfügung gestellt. Nachdem die Software still gestartet wurde, wartet sie auf ein Startsignal des Angreifers. Nach Aktivierung durch den Angreifer werden die Daten schließlich aufgezeichnet und per FTPS zum Angreifer übermittelt.

Die Software dient ausdrücklich rein akademischen Zwecken. Die Verwendung der gezeigten Codes erfolgt auf eigenes Risiko und ist für böswillige Zwecke nicht erlaubt.

Für ein besseres Verständnis des Abgabeumfangs im zip-Archiv, sei an dieser Stelle bereits auf Kapitel 4.1 „Hinweise zum Abgabe-Umfang" (S.15/16) verwiesen.

Inhaltsverzeichnis

1 Netzwerk-Architektur & Konzept

Zur Umsetzung der Keylogger-Schadsoftware wird die folgende Netzwerk-Architektur gewählt:

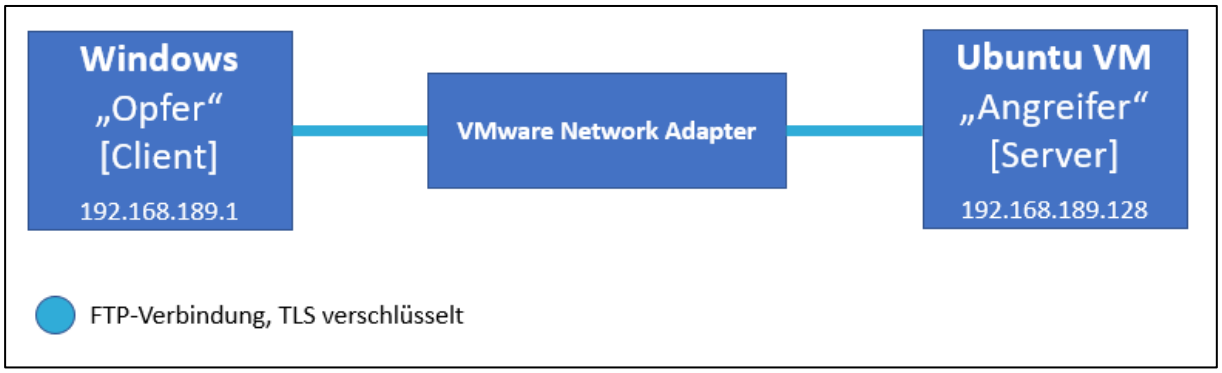

Abbildung 1 - Netzwerk-Architektur

Die Ubuntu Virtual Machine (Ubuntu 22.04 auf VMware Workstation 17) simuliert den Angreifer (Server), der dem Windows-Opfer (Windows 11 Home, Client) die Schadsoftware zur Verfügung stellt sowie die aufgezeichneten Daten entgegennimmt. Die Schadsoftware ist auf Windows-Betriebssysteme ausgelegt und wird auf dem Opfer-Rechner innerhalb eines manipulierten Python-Paketes mittels „pip install“ eingeschleust. Das eigentliche Executable wird dabei im Rahmen der „setup.py“ vom Angreifer per FTPS zur Verfügung gestellt. Die folgende Grafik verdeutlicht den Ablauf der Infektion:

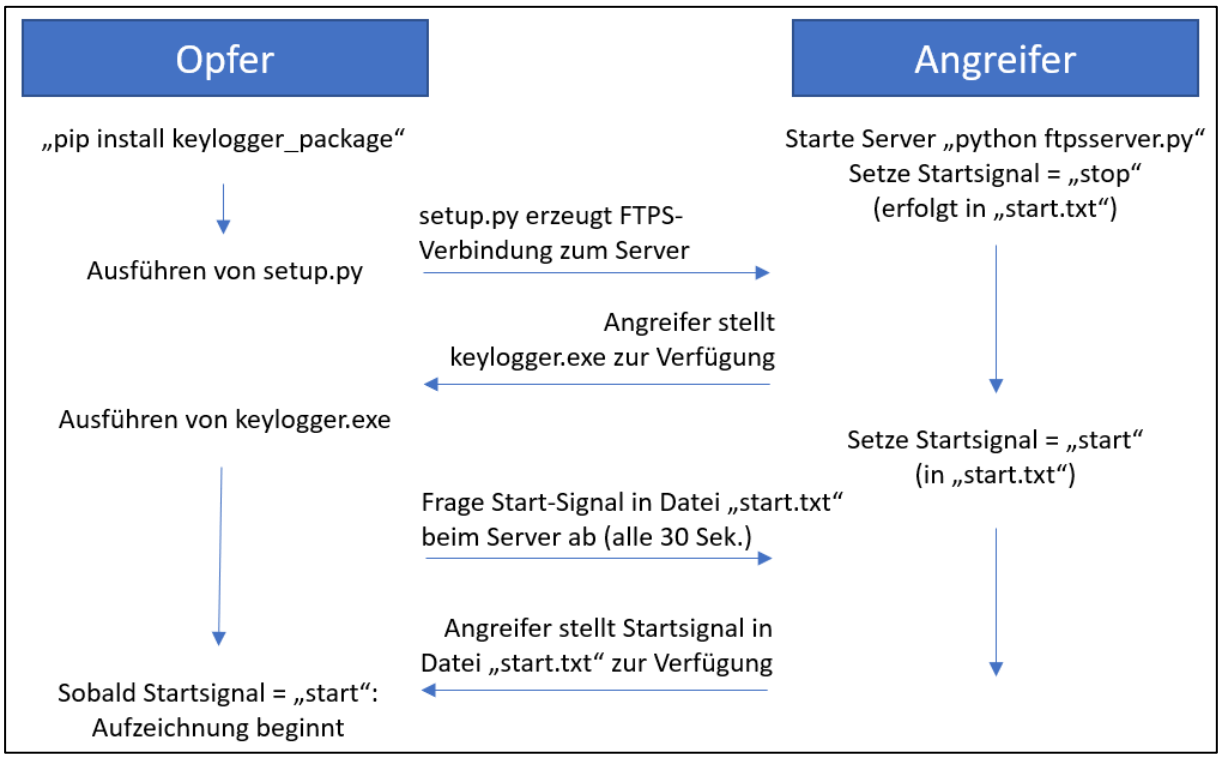

Abbildung 2 - Ablauf der Infektion

Nach erfolgreicher Infektion wird die Anwendung „keylogger.exe“ auf dem Opfer-Rechner ausgeführt. Diese liest alle 30 Sekunden per FTPS-Verbindung die Datei „start.txt“ aus, die auf dem Server abgelegt ist. Sobald der Angreifer in der ersten Zeile der Datei „start.txt“ das Start-Zeichen „start“ eingibt und die Datei abspeichert, wird beim nächsten Auslesen der „start.txt“ durch das Opfer die eigentliche Aufzeichnung der Tastatur- und Maus-Eingaben beim Opfer gestartet. Die folgende Darstellung verdeutlicht die Abläufe zwischen Opfer und Angreifer nach Beginn der Aufzeichnung:

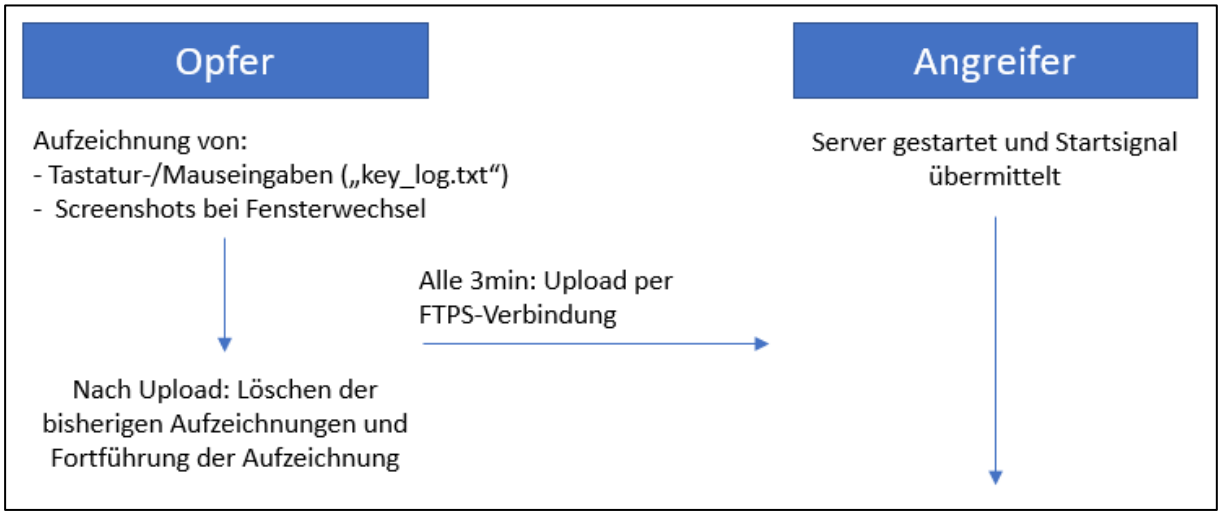

Abbildung 3 - Ablauf der Aufzeichnung

Die Tastatur- und Mauseingaben werden in der Datei „key_log.txt“ aufgezeichnet. Sobald der User auf dem Opfer ein Fensterwechsel vollzieht, wird ein Screenshot erstellt und der Fensterwechsel auch in der „key_log.txt“ dokumentiert um im Nachgang eine Zuordnung von Tastatur- und Mauseingaben zu entsprechenden Fenstern und Programmen zu ermöglichen.

2 Programmierung

Die Software ist analog der Netzwerk-Architektur (siehe Kapitel 1) zweigeteilt in Client- und Server-Bestandteile. Diese beiden Bestandteile werden anschließend erklärt. Sämtliche Programmteile wurden in der Programmiersprache Python entwickelt.

2.1 Client

Die wesentliche Client-Funktion verbirgt sich im Python-Skript „keylogger.py“, das zu Beginn dieses Kapitels näher erläutert wird. Anschließend wird das Set-Up des Clients mittels des manipulierten Python-Packages verdeutlicht. Um das Debuggen zu erschweren, wurde die Datei „keylogger.py“ mit Hilfe des Python-Moduls „PyArmour“ obfuskiert und mittels des Python-Moduls „PyInstaller“

inklusive aller benötigen Module und Abhängigkeiten zur Anwendung „keylogger.exe“ zusammengefügt. Dieses Executable wird dem Client dann vom Server zur Verfügung gestellt.

Die nachfolgenden Erläuterungen zum Code sind zur besseren Verständlichkeit und Übersichtlichkeit auf das Wesentliche reduziert. Für detailliertere Informationen sei auf das eigentliche Skript „keylogger.py“ verwiesen, welches eine nahezu zeilenweise Kommentierung des Codes enthält.

2.1.1 keylogger.py

Main | Python-Skript: Zeile 270 bis 315

Die Main-Funktion des Keyloggers besteht aus zwei While-True-Schleifen. Die **äußere While-True-Schleife** ist eine Polling-Schleife, in der eine FTPS-Verbindung zum Server hergestellt und der Start-Parameter erfragt wird. Der Start-Parameter ist auf dem Server in der ersten Zeile der Datei „start.txt“ gespeichert. Entspricht der Start-Parameter nicht dem String „start“, wartet der Client 30 Sekunden und fragt dann erneut den Start-Parameter ab (=„Polling“-Prinzip). Entspricht der Start-Parameter hingegen dem String „start“, d.h. der Server hat das Signal zum Aufzeichnen gegeben, so werden Tastatur- und der Maus-Listener gestartet (Nutzung des Python-Moduls „pynput“). Anschließend springt das Programm in die **innere While-True-Schleife**, in der alle 3 Minuten der Upload mittels der Funktion „upload“ gestartet wird.

on_press(key) | Python-Skript: Zeile 112 bis 150

Die Funktion wird aufgerufen wenn das Opfer eine Tastatureingabe tätigt. Es wird zunächst geprüft ob das Opfer das Fenster gewechselt hat (mittels Funktion get_active_window(), s.u.). Falls dies der Fall ist, wird ein Screenshot erzeugt (mittels Funktion screenshot2(), s.u.) und die entsprechende Information zum Screenshot in den Key-Storage „keys“ geschrieben. Dies ist wichtig um hinterher die Tastatur-Eingaben den entsprechenden Screenshots und Anwendungen zuordnen zu können. Anschließend wird der getippte Key dem Key-Storage hinzugefügt. Zudem wird geprüft, ob der User mittels STR+V einen Inhalt aus seiner Zwischenablage eingefügt hat und falls ja, wird der Inhalt ebenfalls in den Key-Storage eingefügt (mittels Funktion copy_clipboard(), s.u.). Abschließend wird geprüft ob es schon wieder an der Zeit ist den Key-Storage in die Datei „key_log.txt“ zu schreiben. Dies erfolgt mit Hilfe der Funktion write_file(keys), s.u. und geschieht in der Regel alle 10 Sekunden.

on_click(x,y,button,pressed) | Python-Skript: Zeile 161 bis 194

Die Funktion wird bei Maus-Klicks des Opfers aufgerufen. Analog zur Funktion on_press(key) wird geprüft ob das Fenster gewechselt wurde und ggf. ein Screenshot erzeugt sowie die Fenster-

Information in den Key-Storage geschrieben. Zudem wird die Information, dass die Maus geklickt wurde zum Key-Storage hinzugefügt. Des Weiteren erfolgt ein Abgleich der Zeit und ggf. das Schreiben des Key-Storage in die Datei „key_log.txt“ (ebenfalls analog zur Funktion on_press(key)).

get_active_window() | Skript: Zeile 37 bis 58

Die Funktion greift mittels Windows-API (Python-Modul ctypes) auf das aktuell geöffnete Fenster zu und gibt das geöffnete Fenster, den Titel des geöffneten Fensters sowie den Titel des zugehörigen Programms zurück. Die Implementierung orientiert sich dabei an der Funktion get_current_process() im Studienbrief 3: Python-Hacks (Rieger, Eisoldt, Schlichtenberger, Welte, & Schneider, 2021) sowie an einer Diskussion auf Stackoverflow (Stackoverflow - Obtain Active Window using Python, 2022).

copy_clipboard() | Skript: Zeile 60 bis 73

Die Funktion greift mittels Windows Clipboard API (Python-Modul win32 clipboard) auf die Zwischenablage zu und schreibt die Daten in die Datei „key_log.txt“. Die Implementierung orientiert sich an der Umsetzung im Tutorial „Create an Advanced Keylogger in Python“ (Collins, 2022).

screenshot2() | Skript: Zeile 82 bis 110

Die Funktion greift mittels Windows-API auf den Desktop zu, erfasst die Größe aller Monitore und erzeugt schließlich ein Screenshot (png-File) mit dem Titel „Screenshot<Screnshot-Nr.><Titel der Anwendung>“. Die Screenshot-Nummer wird dabei immer weiter hochgezählt. Der Titel der Anwendung wurde vorher über die Funktion get_active_window() ermittelt. Die Datei-Bezeichnung erlaubt somit eine Zuordnung der Screenshots zu Tastatur- und Maus-Eingaben da die Bezeichnung auch in den Key-Storage und somit in die Datei „key_log.txt“ geschrieben wird (s.o.). Die Implementierung orientiert sich an der Funktion SaveScreenshot(Filename) im Studienbrief 3: Python-Hacks (Rieger, Eisoldt, Schlichtenberger, Welte, & Schneider, 2021).

write_files(keys) | Skript: Zeile 197 bis 225

Die Funktion schreibt den übergebenen Key-Storage „keys“ in die Datei „key_log.txt“. Dabei werden bestimmte Sonderzeichen wie z.B. Leerzeichen, Backspace oder Enter herausgefiltert und in der key_log.txt kenntlich gemacht beziehungsweise ersetzt.

upload() | Skript: Zeile 227 bis 268

Die Funktion lädt schließlich die Datei „key_log.txt“ sowie die erzeugten Screenshots vom Client per TLS-verschlüsselter FTP-Verbindung zum Server. Hierbei werden die Dateien mit einem Zeitstempel versehen und je Upload-Vorgang ein neuer Ordner auf dem Server erzeugt. Dies dient der

Übersichtlichkeit der Ordner- und Dateien-Struktur auf dem Server. Zuletzt werden die entsprechenden Dateien beim Client gelöscht und der Upload-Vorgangs-Zähler erhöht.

2.1.2 Set-Up des manipulierten Python-Packages

Da die Schadsoftware aus Sicherheitsgründen nicht als wirkliches Modul auf PyPI veröffentlicht werden sollte, wird das manipulierte Python-Package nur lokal erzeugt. Hierzu wird die folgende Ordner- und Dateistruktur angelegt (siehe Abbildung 4).

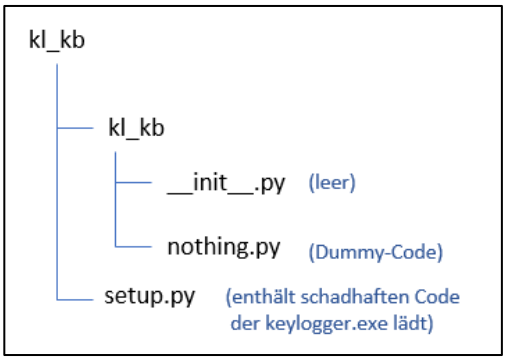

Abbildung 4 - Manipuliertes Python-Package

Wird nun im oberen Ordner „kl_bb“ mit dem Befehl „pip install .“ das manipulierte Package installiert, wird der schadhafte Code in der „setup.py“ ausgeführt. Dieser Schadcode enthält folgende Funktion:

- Herstellung einer TLS-verschlüsselten Verbindung mit dem FTP-Server (Angreifer)
- Download & Ausführung der vom Server bereitgestellten Anwendung „keylogger.exe“

Die Keylogger-Anwendung wurde somit erfolgreich beim Client gestartet und verrichtet nun im Verborgenen ihre Arbeit, indem sie nach Erhalt des Start-Signals die Tastendrücke und Maus-Klicks beim Client aufzeichnet, entsprechende Screenshots erstellt und alles an den Server übermittelt.

2.2 Server

Die wesentliche Server-Funktion verbirgt sich im Python-Skript „ftpsserver.py“, das zu Beginn dieses Kapitels näher erläutert wird. Anschließend wird die Umsetzung des „Start-Parameters“ mithilfe der Datei „start.txt“ verdeutlicht. Schließlich wird die Obfuskation der „keylogger.py“ (siehe Kapitel 2.1.1) mittels „PyArmor“ und die Erzeugung der vom Server zur Verfügung gestellten „keylogger.exe“ mittels „PyInstaller“ kurz erklärt.

2.2.1 ftpsserver.py

Der FTP-Server wird mit Hilfe des Python-Moduls „pyftpdlib“ im Python-Skript „ftpsserver.py“ definiert und auf dem angreifenden Rechner ausgeführt. Die Realisierung orientiert sich dabei an der offiziellen Modul-Dokumentation und nutzt zur Vereinfachung auch das dort zur Verfügung gestellte Zertifikat für die Verschlüsselung mittels TLS (Rodola, 2022). Der Server legt zunächst einen User an und erzeugt einen „TLS_FTP Handler“. Dieser Handler wird genutzt um „FTP over TLS“ analog RFC-4217 umzusetzen (insbesondere die Verschlüsselung mit dem Zertifikat). Anschließend wird eine

Server-Instanz erzeugt, der der „TLS_FTP Handler“ übergeben wird. Der Server ist nun bereit und wird mittels server.serve_forever() gestartet und hört auf dem ebenfalls angegebenen Port 2121.

2.2.2 Umsetzung des Start-Parameters in der Datei „start.txt“

Da die Keylogger-Anwendung nicht sofort „scharf geschaltet“ werden soll sobald sie auf dem Opfer-Rechner gestartet wird, bedarf es einer Funktionalität mit der der Angreifer dem Opfer signalisieren kann, dass die Aufzeichnung beginnen soll – den sogenannten „Start-Parameter“. Um es möglichst einfach zu halten, wurde eine recht simple Variante der Umsetzung gewählt. Wie bereits in der Erläuterung der Main-Funktion in Kapitel 2.1.1 „keylogger.py“ erwähnt, öffnet die Keylogger-Anwendung alle 30 Sekunden die Datei „start.txt“ auf dem Server und liest die erste Zeile ein. Entspricht diese Zeile dem String „start“, wird dies als Start-Signal angesehen und die eigentliche Keylogger-Funktionalität beginnt. Möchte der Server die Aufzeichnung noch hinauszögern, so muss er auf seinem Rechner die „start.txt“ entsprechend so anpassen, dass in der ersten Zeile etwas anderes als der String „start“ steht.

2.2.3 Bereitstellung der „keylogger.exe“ auf Basis von „keylogger.py“

Der Server stellt dem Client die Anwendung „keylogger.exe“ zur Verfügung, die die eigentliche Keylogger-Funktionalität aus „keylogger.py“ (siehe Kaptitel 2.1.1 „keylogger.py“) sowie sämtliche benötigte Module und Abhängigkeiten enthält. Zur Obfuskation der „keylogger.py“ und der anschließenden Erstellung der „keylogger.exe“ wurden die Python-Module „PyArmor“ sowie „PyInstaller“ eingesetzt (Dashingsoft Corp., 2022).

Mithilfe des folgenden Befehls wird das Python-Skript zunächst obfuskiert und anschließend zu einem Executable zusammengefügt:

pyarmor pack -e “ --onefile --noconsole ” keylogger.py

Der Zusatz --noconsole bewirkt dabei, dass die Anwendung auf dem Opfer-Rechner „im Stillen“, d.h. ohne Konsolen-Ausgabe ausgeführt wird.

3 Ausführung & Demo

Nachdem das Set-Up (Kapitel 1) sowie die zugrundeliegenden Software-Bausteine (Kapitel 2) vorgestellt wurden, soll die Funktionsweise der Keylogger-Schadsoftware nun anhand eines Beispiels mit Screenshots verdeutlicht werden. Die Demo wird in die folgenden Schritte unterteilt: 1. Starten des Servers (Kapitel 3.1), 2. Installation des manipulierten Python-Packages und

Ausführung der Keylogger Schadsoftware auf dem Opfer-Rechner (Kapitel 3.2) sowie 3. Start-Signal auf dem Server setzen und Aufzeichnung beginnen (Kapitel 3.3).

3.1 Server starten

Die Server-Umgebung ist wie in Abbildung 5 auf dem Angreifer-Rechner (VM: Ubuntu 22.04) eingerichtet. Neben dem Server-Skript „ftpsserver.py“ sind auf dem Server noch die Datei „start.txt“ (enthält Start-Parameter), die Datei „keylogger.exe“ (Schadsoftware-Executable für den Client) sowie im Ordner sec die Datei „keycert.pem“ (TLS-Zertifikat) angelegt.

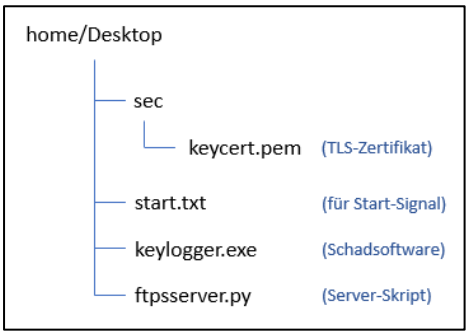

Abbildung 5 - Server-Umgebung

Zum Starten des Servers wird das Skript „ftpsserver.py“ ausgeführt (siehe Abbildung 6). Der Server ist somit gestartet und hört auf Port „2121“. Die Text-Datei „start.txt“ enthält zunächst den String „stop“ (siehe Abbildung 7), d.h. falls sich der Client mit dem Server verbindet, wird die Aufzeichnung von Tastatur und Maus noch nicht gestartet. Der Server (Angreifer) ist nun bereit für die Verbindung mit dem Client (Opfer).

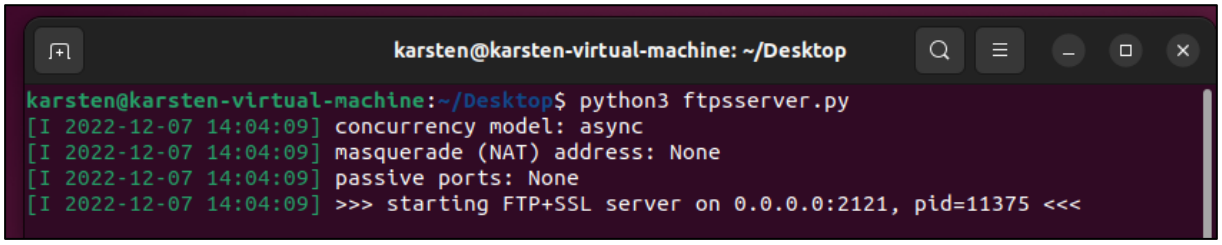

Abbildung 6 - Start des Servers

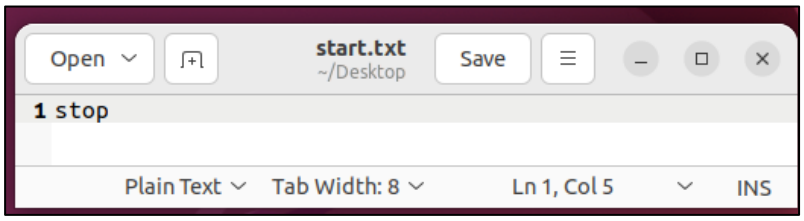

Abbildung 7 - Start.txt

3.2 Installation des manipulierten Python-Packages

Die Installation des manipulierten Python-Packages auf dem Opfer-Rechner erfolgt im angelegten Package-Ordner (Ordner-Struktur siehe Abbildung 4, Seite 9). Mittels „pip install .“ wird das entsprechende Package installiert und dabei die infiltrierte „setup.py“ ausgeführt (siehe Abbildung 8). Diese stellt eine Verbindung zum FTP-Server her, lädt die Schadsoftware „keylogger.exe“ vom

Server herunter und führt diese vom Client unbemerkt im Hintergrund aus (siehe Task-Manager in Abbildung 9).

```
PS C:\Users\kabla\OneDrive\Desktop\Studium\WS2223\HackingMitPython\HackingBsp\kl_kb> pip install .
Processing c:\users\kabla\onedrive\desktop\studium\ws2223\hackingmitpython\hackingbsp\kl_kb
  Preparing metadata (setup.py) ... done
Installing collected packages: kl-kb
```

Abbildung 8 - Installation manipuliertes Package

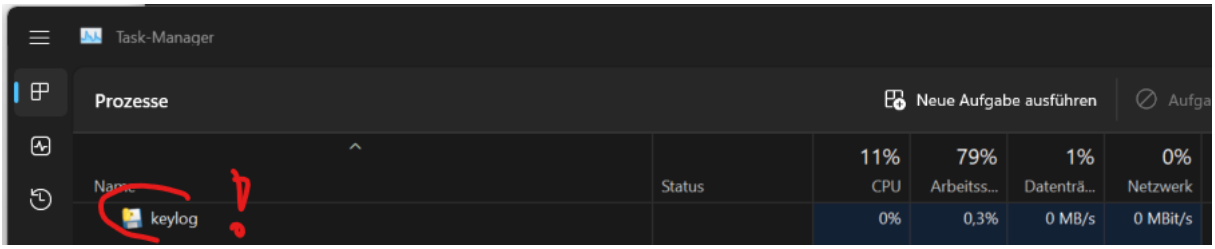

Abbildung 9 - Keylogger im Task-Manager sichtbar

Der Server meldet entsprechend den Verbindungsaufbau und den Download der „keylogger.exe“ durch den Client (siehe Abbildung 10).

```
[I 2022-12-07 14:15:23] 192.168.189.1:53740-[] FTP session opened (connect)
[I 2022-12-07 14:15:23] 192.168.189.1:53740-[user] USER 'user' logged in.
[I 2022-12-07 14:15:23] 192.168.189.1:53740-[user] RETR /home/karsten/Desktop/keylogger.exe completed=1
[I 2022-12-07 14:15:23] 192.168.189.1:53740-[user] FTP session closed (disconnect).
```

Abbildung 10 - Server liefert keylogger.exe

Außerdem meldet der Server im Anschluss alle 30 Sekunden, dass der Client die Datei „start.txt“ ausgelesen hat (siehe Abbildung 11). Dies zeigt das „Polling“ des Start-Parameters durch den Client. Da der Start-Parameter in der Datei „start.txt“ aber noch auf „stop“ gesetzt ist, beginnt der Client noch nicht mit der Aufzeichnung.

```
[I 2022-12-07 14:15:26] 192.168.189.1:53744-[user] USER 'user' logged in.
[I 2022-12-07 14:15:26] 192.168.189.1:53744-[user] RETR /home/karsten/Desktop/start.txt completed=1
[I 2022-12-07 14:15:26] 192.168.189.1:53744-[user] FTP session closed (disconnect).
[I 2022-12-07 14:15:56] 192.168.189.1:53762-[] FTP session opened (connect)
[I 2022-12-07 14:15:56] 192.168.189.1:53762-[user] USER 'user' logged in.
[I 2022-12-07 14:15:56] 192.168.189.1:53762-[user] RETR /home/karsten/Desktop/start.txt completed=1
[I 2022-12-07 14:15:56] 192.168.189.1:53762-[user] FTP session closed (disconnect).
```

Abbildung 11 - Client fragt Start-Parameter ab

3.3 Beginn der Aufzeichnung

Passt der Angreifer die „start.txt“ auf dem Server dahingehend an, dass in der ersten Zeile der String „start“ geschrieben ist, so beginnt der Client beim nächsten „Polling“-Durchlauf mit der Aufzeichnung. In der Demo erfolgt nun beim Client folgendes Szenario: Der User öffnet zunächst Chrome und tippt in die Chrome-Suchleiste „Wie programmiere ich einen Keylogger?“ + ENTER ein Anschließend wechselt er zu Notepad und schreibt dort „Genug gegoogelt. Ich schreibe nun ein Gedicht“. Anschließend fügt er ein zuvor kopiertes Gedicht („Zauberlehrling“) aus der

Zwischenablage ein. Die entsprechenden Screenshots sind wurden nun beim Client erstellt und die Tastatur-/Maus- und Zwischenablage-Eingaben beim Client in der Datei „key_log.txt“ gespeichert.

Beim nächsten Upload (nach 3 Minuten) werden diese Dateien zum Server hochgeladen und beim Client gelöscht. Auf dem Server wird ein neues Vereichnis („targetfiles“) sowie ein neues Unterverzeichnis („0“, für „nullten“ Upload) angelegt. Abbildung 12 und Abbildung 13 zeigen die Ausgabe auf der Server-Konsole.

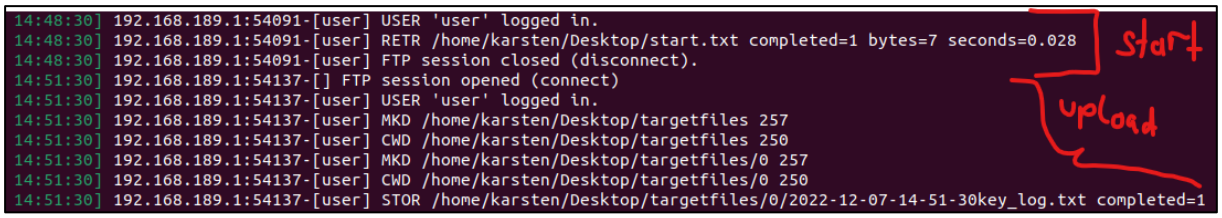

Abbildung 12 - Start der Aufzeichung und 1. Upload

In Abbildung 12 ist zu erkennen, dass der Client um 14:48:30 Uhr das letzte Mal die Datei „start.txt“ vom Server ausliest. Sie enthält nun das Startsignal „start“ und der Client beginnt die Aufzeichnung. Genau 3 Minuten später um 14:51:30 Uhr erfolgt der erste Upload. Der Client erstellt zunächst das Verzeichnis „targetfiles/0“ und lädt dort die mit dem Zeitstempel versehene Datei „key_log.txt“ hoch. Zudem erfolgt der Upload der zwei Screenshots zu Chrome und Notepad (Abbildung 13).

Abbildung 13 - 1. Upload - Screenshots

Der Angreifer hat somit erfolgreich die Tastatur-/Maus- und Zwischenablage-Eingaben des Clients sowie die zugehörigen Screenshots erhalten. Er kann das aufgezeichnete Material nun nach Username/Passwort-Eingaben durchforsten. Der Client hat dabei vom Angriff nichts gemerkt.

Schauen wir uns die Ergebnisse des Uploads im Ordner „targetfiles/0“ auf dem Server genauer an:

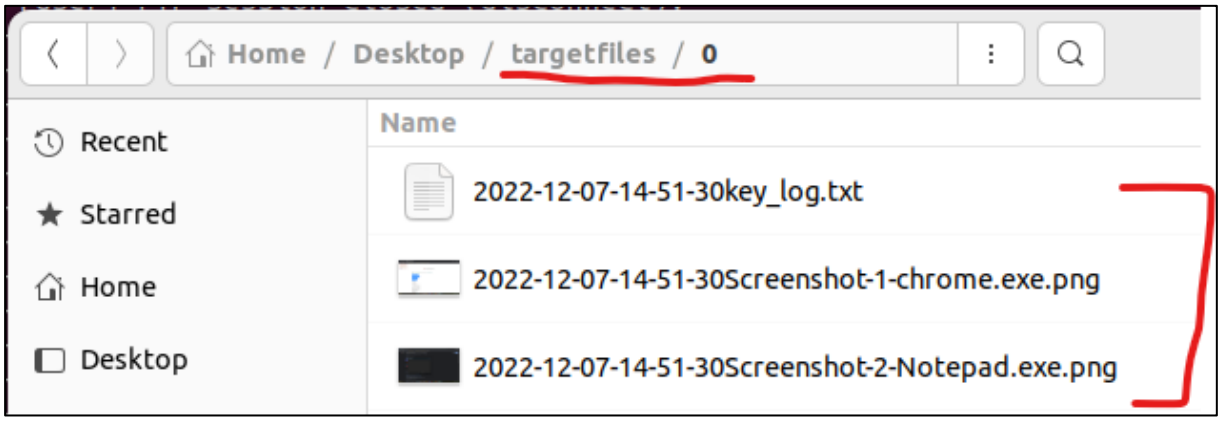

Abbildung 14 - Inhalt des Ordners targetfiles/0

Die Datei mit der Endung „key_log.txt“ enthält die aufgezeichneten Tastatur-/Maus- und Zwischenablage-Eingaben des Clients:

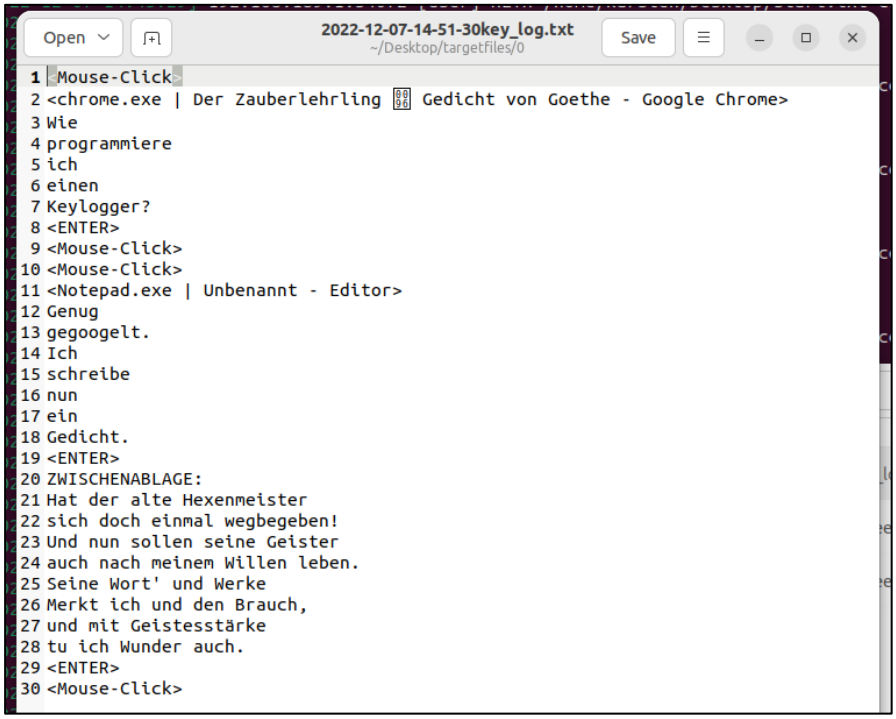

Abbildung 15 - Inhalt von "key_log.txt"

Der Angreifer kann der Datei entnehmen in welchem Fenster das Opfer welche Eingaben gemacht hat. Auch das Einfügen des Gedichtes aus der Zwischenablage in Notepad ist klar gekennzeichnet. Mittels der Screenshots (siehe Abbildung 16 & 17) kann der Angreifer weitere Details zu den genutzten Oberflächen erkennen.

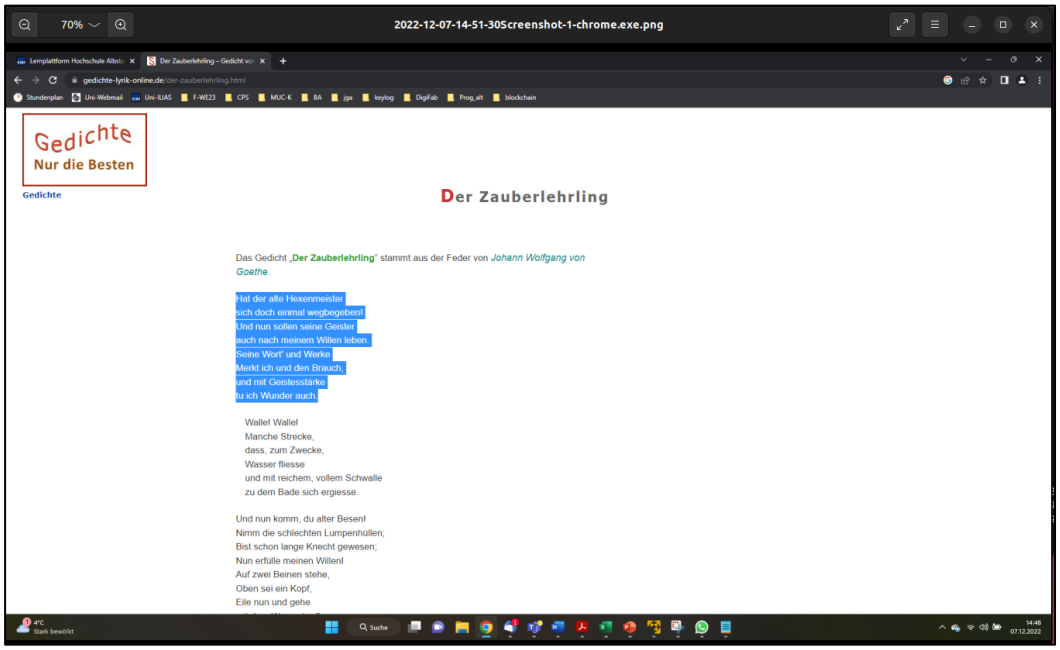

Abbildung 16 - screenshot-1chrome.exe.png

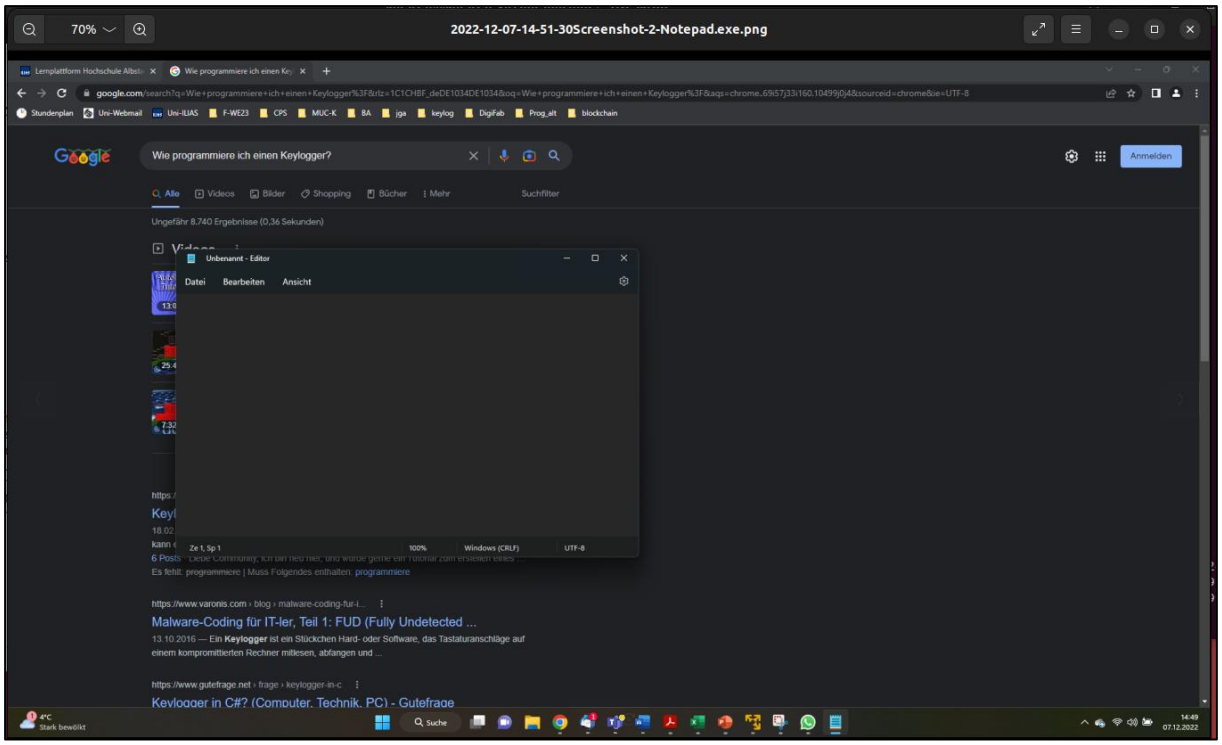

Abbildung 17 - screenshot-2-Notepad.exe.png

4 Bemerkungen & Fazit

Nachdem die Funktionsweise der Keylogger-Schadsoftware anhand eines Beispiels verdeutlicht wurde, folgen nun noch Hinweise zum Abgabe-Umfang der Hausarbeit sowie einige abschließende Bemerkungen.

4.1 Hinweise zum Abgabe-Umfang

Das abgegebene Zip-Archiv „Hausarbeit_Keylogger_PythonHacking_KarstenBlank_90924.zip“ ist wie folgt aufgebaut (siehe auch Abbildung 18 auf nächster Seite): Auf oberster Ebene liegt die schriftliche Ausarbeitung im PDF-Format („Hausarbeit_PythonHacking_KarstenBlank_90294.pdf“) sowie der Source-Code der Schadsoftware als Python-Skript („keylogger.py“). Zudem befinden sich auf der Ebene die Ordner „Client“ sowie „Server“, deren Inhalt genau den Darstellungen in Kapitel 2.1.2 „Set-Up des manipulierten Python-Packages“ (Ordner „Client“) sowie Kapitel 3.1 „Server starten“ (Ordner „Server“) entspricht. Im Ordner „Client“ befindet sich somit das manipulierte Python-Package „kl_kb“ (**k**ey**l**ogger_**k**arsten**b**lank) mit der infiltrierten „setup.py“, die die Schadsoftware vom Server lädt und ausführt. Im Ordner „Server“ befinden sich alle „Server-seitig“ benötigten Dateien, insbesondere das eigentliche Server-Skript „ftpsserver.py“ sowie die obfuskierte Schadsoftware „keylogger.exe“, die der Server dem Client bereitstellt.

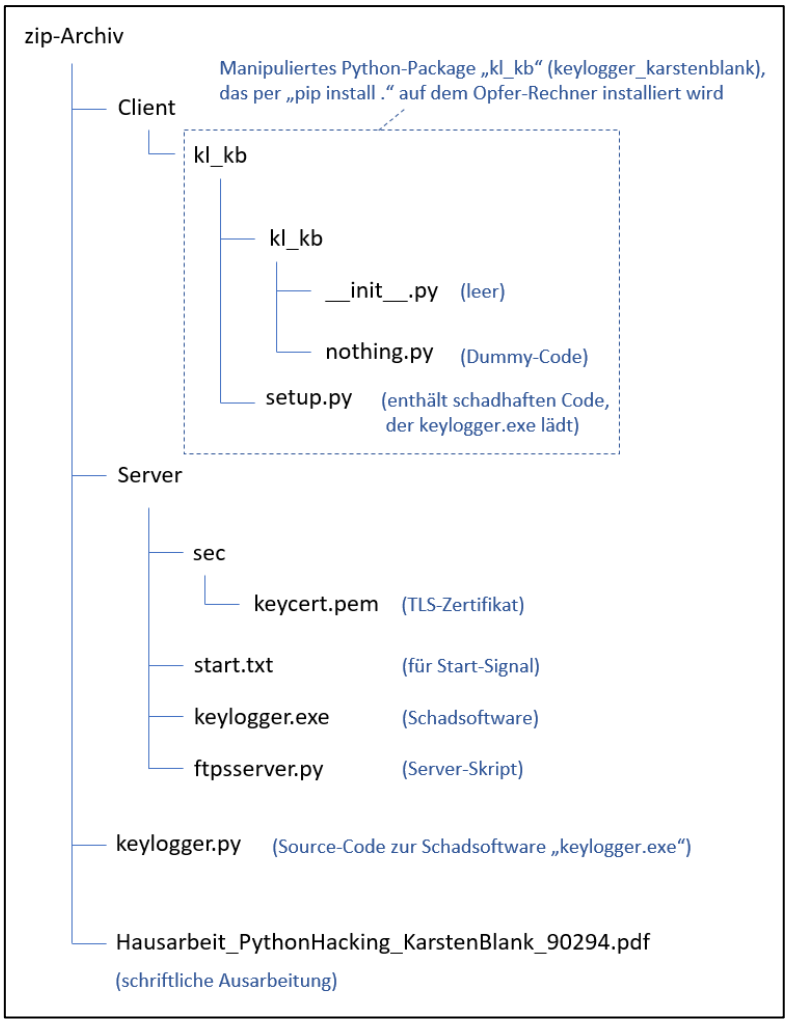

Abbildung 18 - Abgabeumfang zip-Archiv

4.2 Fazit

Diese Hausarbeit hat gezeigt, wie Angreifer mit relativ überschaubaren Mitteln Python-Packages manipulieren und über böswillige Anpassungen der „setup.py“ Schadsoftware auf Opfer-Rechnern ausführen können. Dies zeigt einmal mehr wie aufmerksam Nutzer von Paket-Managern wie PyPI sein müssen. Häufig werden Packages veröffentlicht, die sehr ähnlich wie bekannte und häufig genutzte Python-Packages benannt sind (Menge-Sonnentag, 2022). Ist der Nutzer hier nicht auf der Hut, kann er sich mittels „pip install“ schnell mit Schadsoftware infizieren. Häufig werden die Packages durch die Paket-Manager zwar auf Schadsoftware geprüft (PyPI prüft z.B. die setup.py automatisiert), doch eine absolute Sicherheit hat der Nutzer hierdurch nicht. Nutzer sollten daher idealerweise den Inhalt der Paketdateien vor Installation überprüfen sowie Pakete mit Wheels-Datei gegenüber der tendenziell veralteten tar.gz/setup.py-Kombination bevorzugen (Augsten, 2022).

Während der vorgestellte Keylogger in der gegebenen Test-Umgebung zwar die gewünschte Funktion erfüllt, sei zum Abschluss noch auf einige Einschränkungen hingewiesen. Zunächst würde

die manipulierte „setup.py“ den automatisierten Check von PyPI vermutlich nicht überstehen da sie als Malware erkannt werden würde. Durch diverse Hinweise ist PyPI bezüglich dieser Sicherheitslücke nämlich bereits sensibilisiert. Zudem wäre es für Experten trotz der Obfuskation mittels PyArmor vermutlich relativ einfach die Funktionalität und Intention der Anwendung „keylogger.exe“ zu entschlüsseln. In der Realität müsste zudem der FTP-Server richtig gehostet werden und eine Möglichkeit eingebaut werden um unterschiedliche Clients zu identifizieren. Zudem sollten natürlich andere Datei-Bezeichnungen genutzt werden als solch offensichtliche wie „keylogger.exe“. Weitere Funktionen könnten außerdem sein, die Dateien nicht nur verschlüsselt zu übertragen sondern diese bereits beim Opfer in verschlüsselter Form abzuspeichern oder (für die Schadsoftware) feindliche Prozesse zu erkennen und die Schadsoftware sodann „schlafen zu legen“. Der Phantasie und den Möglichkeiten sind bei der Entwicklung von Schadsoftware also keine Grenzen gesetzt.

Literaturverzeichnis

Augsten, S. (2022, 09 16). *Security-Insider.de - Python-Sckript setup.py als Malware-Angriffsvektor*. Retrieved from https://www.security-insider.de/python-skript-setuppy-als-malware-angriffsvektor-a-a995b8b97a62b7d249ed53597592e410/

Collins, G. (2022, 11 20). *Youtube - Create an Advanced Keylogger in Python - Crash Course*. Retrieved from https://www.youtube.com/watch?v=25um032xgrw

Dashingsoft Corp. (2022, 11 20). *PyArmor's Documentation*. Retrieved from https://pyarmor.readthedocs.io/en/latest/index.html

Menge-Sonnentag, R. (2022, 11 15). *Heise.de - Erneut Schadcode im Python-Paketmanager aufgetaucht*. Retrieved from https://www.heise.de/developer/meldung/Erneut-Schadcode-im-Python-Paketmanager-aufgetaucht-4605789.html

Python Software Foundation. (2022, 11 15). *ftplib - FTP protocol client - Python 3.11.1 Documentation*. Retrieved from https://docs.python.org/3/library/ftplib.html#module-ftplib

Rieger, M., Eisoldt, P., Schlichtenberger, D., Welte, B., & Schneider, C. (2021). *Studienbrief 3: Python-Hacks aus "Python Penetration Testing".* Albstadt.

Rodola, G. (2022, 11 20). *pyftpdlib Documentation*. Retrieved from https://pyftpdlib.readthedocs.io/en/latest/index.html

Stackoverflow - Obtain Active Window using Python. (2022, 11 15). Retrieved from https://stackoverflow.com/questions/10266281/obtain-active-window-using-python

Abbildungsverzeichnis