



Pen Drive Attack

11/24/2022

0223826 Arcos Bravo David Gamaliel

0226594 Gómez Delgado Sara Carolina

0224969 Robles Jiménez Luis Eduardo

Distributed Computing

Disclaimer

This text is particularly written on an educational purposes only basis, and its use is permitted only with the same purpose. None of the material and technologies used here shall be linked to the authors of this project under any detrimental circumstances.

Overview

An attack involving a pendrive can be found nowadays in a number of different ways with a lot of different targets and objectives. Some of the best-known techniques are the USB drop, phishing through files that are in the drive, malware spreading and in this particular case, data stealing will be covered.

Chosen technique

This project is intended to retrieve valuable information of the user whose computer is being attacked with the USB. And the objective of this is realizing how secured our data is within our computers.

That being said, the stages of the strategy were:

- 1. Bait.** Since the operating systems keep evolving to avoid unknown software running freely on our computers, making a self-executable file becomes a challenge. So a workaround to this was to use the biggest security breach of a system: *the user*.

In order to achieve this, the attractive bait for an average user is a well-known game called *sudoku*.

Sudoku Final Project

How to play?

- Sudoku grid consists of 9x9 spaces.
- You can use only numbers from 1 to 9.
- Each 3x3 block can only contain numbers from 1 to 9.
- Each vertical column can only contain numbers from 1 to 9.
- Each horizontal row can only contain numbers from 1 to 9.
- Each number in the 3x3 block, vertical column or horizontal row can be used only once.
- The game is over when the whole Sudoku grid is correctly filled with numbers.

	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

2. **A sneak.** Whether the user is trying to play the sudoku or finding out what the underlying intentions of the suspicious game were. A program would already be crawling through the file storage looking for the information that web browsers store. This piece of software is programmed in golang and then built into a binary file in order to look as an ordinary executable file and has support for at least 40 different web browsers spread among 3 different operating systems.
3. **A gossipier.** Since trying to get the USB back represents a significant risk, another script is used instead. It is programmed in python and after the sneak does their job, the executable takes the files with the valuable information and with a previously given custom list, discriminates some files from the others, and the most important ones are sent to an anonymous, temporal server that is hosted on a 2x5 inches small computer somewhere in Mexico.
4. **Erase the trace.** Once the valuable information has been sent, the hacker already has the information in their hands and is able to do whatever they want with it. All the retrieved and temporary files are deleted from the USB in order to avoid leaving any pieces of evidence behind.

Results

After using this attack against several targets and the authors themselves, it's possible to see that technology is not able to protect our data by itself. We need an active effort to keep what we consider valuable secure. Throughout the process and a number of attempts, several warnings showed up in different computers calling out some stages of the attack, so a responsible user would be able to prevent it. However, a more complex trap could be used in order to meet the expectation, an example may be to physically distract the user if they are within our reach or several other social engineering techniques could be employed.

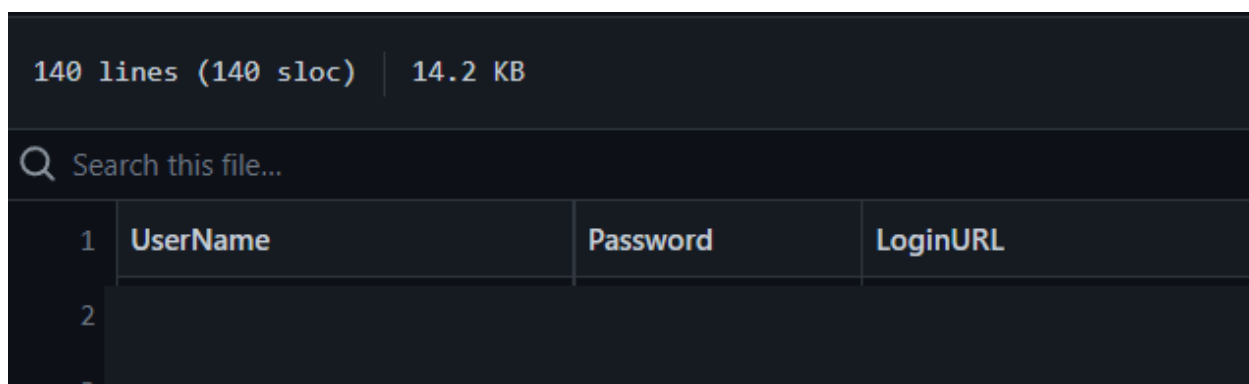
The following images are examples of the retrieved data with the respective users' consent.



The screenshot shows a file explorer interface with a dark theme. The breadcrumb path is 'PenDrive-Attack / server / goldPot /'. The file list contains 18 CSV files, each with a unique ID and a timestamp. The files are sorted by name, and the timestamps range from '3 hours ago' to '1 hour ago'. The file size is indicated as 'f15c65b' and '1 hour ago'.

File Name	Size	Time
1669322043.csv	f15c65b	3 hours ago
1669322045.csv	f15c65b	3 hours ago
1669322220.csv	f15c65b	3 hours ago
1669322222.csv	f15c65b	3 hours ago
1669322282.csv	f15c65b	3 hours ago
1669322285.csv	f15c65b	3 hours ago
1669324191.csv	f15c65b	2 hours ago
1669324196.csv	f15c65b	2 hours ago
1669324198.csv	f15c65b	2 hours ago
1669324478.csv	f15c65b	2 hours ago
1669324482.csv	f15c65b	2 hours ago
1669324485.csv	f15c65b	2 hours ago
1669324724.csv	f15c65b	2 hours ago
1669324727.csv	f15c65b	2 hours ago
1669324732.csv	f15c65b	2 hours ago
1669327568.csv	f15c65b	1 hour ago

A small sample of all the retrieved files.



The screenshot shows a file viewer interface with a dark theme. The file has 140 lines (140 sloc) and is 14.2 KB. The search bar is empty. The table has three columns: 'UserName', 'Password', and 'LoginURL'. The first row is the header, and the second row is the first data entry.

1	UserName	Password	LoginURL
2			
3			

The header of a file that contains a user's passwords with 140 entries.