Detecting Data Leakage from Pod Slurping Based Attacks on a Windows XP Platform

T. Kavallaris and V. Katos

Department of Electrical and Computer Engineering,
Democritus University of Thrace
e-mail: ttkavs@gmail.com, vkatos@ee.duth.gr

Abstract

Time is recognised to be a dimension of paramount importance in computer forensics. In this paper, we report on the potential of identifying past pod slurping type of attacks by constructing a synthetic metric based on information contained in filesystem timestamps. More specifically, by inferring the transfer rate of a file from last access timestamps and correlating that to the characteristic transfer rate capabilities of a suspicious USB found in the Windows registry, one could assess the probability of having suffered an unauthorised copy of files. Preliminary findings indicate that file transfer rates can be associated with the make and model of the USB storage device and give supporting information to the forensic analyst to identify file leakages.

Keywords

USB forensics, data transfer rate.

1. Introduction

Along with its commercial availability in 2004, the iPod raised security concerns with regards to the confidentiality aspects of corporate security (Rohde, 2004). The iPod seemed to be the vehicle to highlight the risks of allowing portable USB storage devices in the corporate, since soon after the published proof of concept *slurp.exe* (Usher, 2005), similar tools were published – see for example the USB switchblade (Hak5, 2008) – for a number of portable devices such as those with U3 functionality which offer autorun capabilities (U3, 2008). These tools run stealthy on the host machine and can copy specified directories or files, or execute potentially malicious payloads.

Provided that the slurping operation is covert and is not prevented or detected, there are no statistics currently available showing the amount of data leakage slurping is accounted for (Radcliff, 2008). Although Verizon (2008) in a breach investigation report announced that the majority of the attacks where external (73% external vs. 18% internal), such information if viewed in isolation could be misleading, as in the same study it was reported that the number of stolen records was substantially higher for the case of internal breaches (375,000 stolen via internal breaches against 30,000 via external breaches).

The contribution of this research is to provide a means to ascertain to some degree whether there was on a given computer a data leakage via a USB device through a covert, automated slurping attack. More precisely, the scenarios studied were based on the following three assumptions:

- there is no access control enforced on the USB ports of the computer;
- the adversary is not the owner or regular user of the computer;
- the files exist on the hard disk of the computer.

It is important to highlight that the methodology presented is not a post mortem investigation in the strict sense, as data leakage is not assumed to have occurred; the approach aims to assist in detecting whether any leakage occurred at some stage in the past on a certain computer. As such, this methodology would typically precede an incident response exercise.

This paper is structured as follows. In Section 2 an outline of the methodology is presented. In Section 3 the experimental data are presented and analysed. Section 4 contains the conclusions, issues identified and future directions.

2. Data Leakage Detection Approach

The main metric used for detecting the data leakage is the data transfer rate of the USB device which is correlated with the file timestamps and the device insertion timestamp. When a bulk file transfer takes place between an (internal) hard disk and a USB device, the bottleneck is normally placed on the communication channel between the USB controller and the USB device. We argue that once a file access timeline is constructed, one can probabilistically determine which files have been copied without permission. As expected, this probability is influenced by the contribution of peripheral circumstantial pieces of evidence by the owner of the computer, such as the recognition of a USB ownership, whether he or she has performed a file backup and so forth. In addition, the probability of a successful identification of the attack depends on the elapsed time from the event; the more recent the attack, the higher the probability of a successful identification.

2.1. Evidence sources

In a forensics investigation, the identification of the evidence domain (traditionally this is referred to as the crime scene) is a step of paramount importance, as potential omissions could lead to drawing wrong conclusions. The practice of identifying, acquiring, analysing and presenting electronic evidence from multiple sources in a forensically accepted manner is referred to as *e-discovery*. In this section we present the sources of evidence which are relevant to the USB based data leakage context.

In Windows XP operating systems, there are primarily two sources of USB related information, the Registry, and the human readable file *setupapi.log* (Thomas and Morris, 2008). The relevant information contained in the Registry includes the identification of the USB device (serial number, manufacturer, product), as well as

time related information, such as the last plugin timestamp. The file *setupapi.log* contains information relating to the first connection of the device to the USB port such as driver installation information. The data which are of particular relevance in the proposed approach are the USB serial number and the timestamps. It should be highlighted that it is not mandatory for a USB device to have a serial number and that depends on the manufacturer. Whenever there is a serial number though, the manufacturer is required to offer uniqueness. The existence of a serial number is important if a forensic investigation involves a seizure and suspect USB devices have been recovered, but it does not affect the accuracy of the proposed method, since the emphasis of this method is primarily on the timestamps. Consequently, the integrity of the timestamps is crucial to the success of the proposed method.

Another important source of evidence is the timestamps which accompany the files on a given filesystem. NTFS metadata contain last access information to the precision of the 100th nanosecond, but the granularity of the Windows operating system is much smaller, as it is limited to the precision of 1ms.

2.2. Correlation

Let U denote the set of USB storage devices and U_O the devices which belong to the legitimate user of the computer. Then $U_S = U \cap U_O$ will be the set of suspicious devices. The proposed method takes place if $U_S \neq \varnothing$. U_S is an unordered set with indexed elements.

Let t_i denote the date and time that device $u_i \in U_S$ was last connected to the computer and let T_a denote a time span constant. For each $u_i \in U_S$, $i = 1..\#U_S$, let r_{v,u_i} denote the average transfer rate of a file of size v for the specific device u_i. Finally, for a given set F of files $f \in F$, let t_{f_j} denote the access time of file $f_j \in F$, $j = 1..\#F$.

If a leakage took place, then it is expected that the access times of a subset of the leaked files are placed within the time range $[t_i, t_i + T_a]$. It should be evident that the older the leakage event, the smaller the subset of the leaked files and consequently the accuracy of the proposed method is reduced. Moreover, if the legitimate user performed a backup of the files after the leakage, the method cannot be applied.

Let $G_i \subseteq F$ such that for each g in G_i, it is $t_i < t_g < t_i + T_a$, where t_g is the last access timestamp of file g. Furthermore, G_i is ordered by the last access timestamp. That is, $G_i = \{g_1, g_2, ..., g_{\#G}\}$, such that $t_{g_m} < t_{g_{m+1}}$, $m = 1..\#G_i - 1$. Let A_i be the ordered, set of timestamps of files in G_i. That is, $A_i = \{a_1, a_2, ..., a_{\#G_i}\}$ with $a_m < a_{m+1}$, $m = 1..\#G_i - 1$.

For every file in $G_i \setminus \{g_{\#G_i}\}$ that was potentially transferred to the USB device, the maximum transfer rate would be equal to $r'_m = (\text{size of } g_m)/(a_{m+1} - a_m)$. This transfer rate is then compared with the average transfer rate of the USB device under investigation. As such, file g_m has potentially been leaked, if statistically $r'_m = r_{v_m,u_i}$, where v_m is the size of g_m.

A key metric for assisting in determining whether a slurping attack was performed at time t_i by device u_i is the leakage probability $L = (\text{potentially leaked files})/\#G_i$. In a production system, it is expected that *L* will be smaller than 1, but the user's contribution is crucial at this stage. For instance, if the user is aware of certain files that he or she has accessed, these could be excluded from the calculations and *L* could be recomputed more accurately. In addition, it is expected that the larger the number of files leaked, the more potentially accurate the information expressed by *L* (always under the assumption that there have been no external events that have influenced the file access timestamps).

The value of the time span T_a depends on a number of factors such as number of files in "attractive" directories, total size of "attractive" files the size of the USB storage device, and the current state of the art. Currently a typical size is between 6 and 12 minutes. In any case, a histogram showing the last access timestamps of files in certain windows near the device plugin timestamp, would contribute with valuable information in order to assist the analyst with assigning a reasonable value to the time span.

3. Empirical data

Three sets of experiments were conducted. The first was in order to find an acceptable approximation function of the average transfer rate, r_{v_m,u_i}. The second was for establishing whether there is discrimination between different makes and models of USB sticks with respect to the transfer rate; that is, $r_{v_m,u_i} \neq r_{v_m,u_j}$ for $i \neq j$. The third test was to investigate whether the probability *L* can be high enough, given a slurping attack.

3.1. Transfer rate

Figure 1 shows a plot of file sizes in log(kb) against the average transfer rate, in MB/sec. The linear regression produced the following:

$$y(x) = 0.8423\log(x) + 2.1069 \text{ Mb/sec} \qquad R^2 = 0.88278$$
$$[0.0000] \qquad [0.0006] \qquad \overline{R}^2 = 0.871057$$

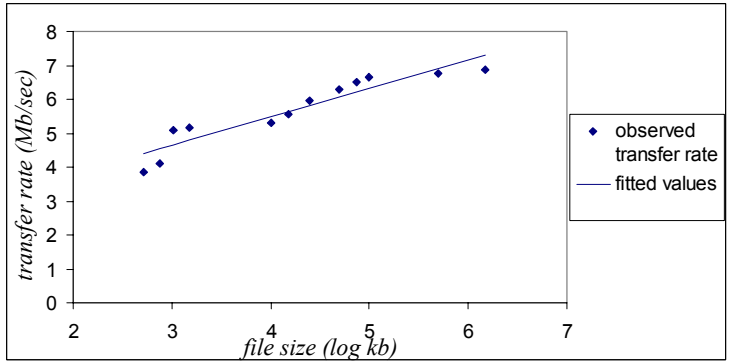

Figure 1: Scatter plot between file size (log) and average transfer rate.

The relation above refers to a Kingston DataTraveler 2.0 device. The total volume transfer for all each file size was 1Gb. The lower transfer rates referred to the smaller files as expected, as there were more operations involved, other than *read* and *write*. The same files were also transferred to a Kingston DT 101 II and a Kingston DataTraveler Mini. The results are summarised in Table 1 and presented in Figure 2.

USB device	$y(x) = a\log(x) + b$	R^2	$\overline{R}^2$	SE
1. Kingston DataTraveler 2.0	$y(x) = 0.8423\log(x) + 2.1069$ [0.0000] [0.0006]	.882779	.871057	.363236
2. Kingston DT 101 II	$y(x) = 0.6379\log(x) + 1.4190$ [0.0000] [0.0002]	.929115	.922027	.208513
3. Kingston DataTraveler Mini	$y(x) = 1.7179\log(x) - 1.3264$ [0.0000] [0.0767]	.925727	.918300	.575844

Table 1: Summary of statistical results for the three USB devices

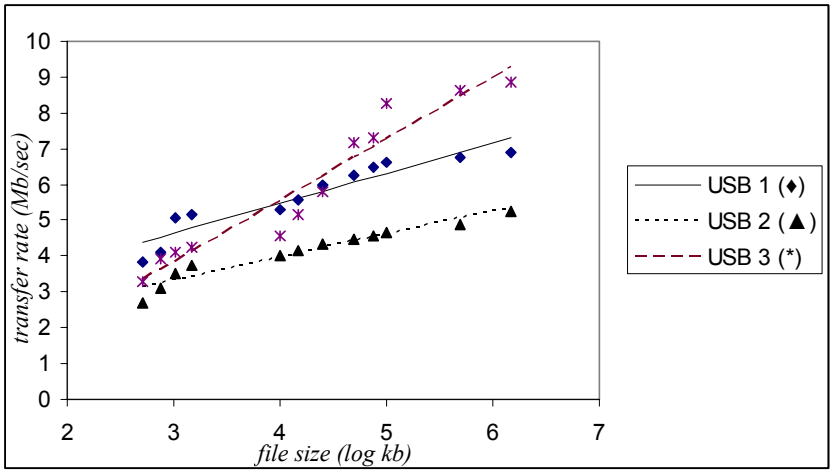

Figure 2: Scatter plot between file size (log) and average transfer rate for all three test USB subjects.

It can be seen from the above that a characteristic transfer rate footprint can be constructed and is distinct for different USB devices. The following section focuses on the detection potential of this information when correlated with the actual last access times.

3.2. Slurping detection potential

The third USB was used as a destination for the "unauthorised" copying of three folders. The first folder contained 13 files, the second folder contained 25 files, and the third folder contained 570 files. Initially all folders contained files of 3 different sizes. The probability of attack *L,* as presented earlier, was the number of potentially leaked files to the total number of files. A potentially leaked file was a file where the calculated transfer rate was within the interval $[y(x) - 2 * SE, y(x) + 2 * SE]$, where $y(x)$ is the modelled transfer rate $y(x) = 1.7179\log(x) - 1.3264$. The results are summarised in Table 2.

Test case	Number of potentially leaked files	Total number of files*	Probability of attack *L*
1. (13 files)	9	12	0.75
2. (25 files)	23	24	0.96
3. (570 files)	464	569	0.81

*the total number of files is the number of files in the case minus 1, since the total access time of the last file cannot be calculated

Table 2: The slurping detection attempts

It can be seen from the above that the probability of successfully detecting the attack is in the region of 0.80, given the relatively higher amount of files transferred in the

third test case. Provided that this probability was calculated for a given attack, this value should be a future reference point for establishing whether an attack took place at a given time in the past. As the above data were generated from a known pod slurping attack, *L* can be used to directly compute the rate of false negatives, which will be equal to 1-*L*.

4. Discussion, conclusions and outlook

It can be reconfirmed that the integrity of the system is of paramount importance in digital forensics and that a strong security policy enforcing, among other elements, integrity protection of the key system files (including the registry) is required. In this particular case, unauthorised modifications of the registry can render the proposed method ineffective. Apart from the obvious tampering of the USB device registry keys, a malicious payload could completely disable the file access times for NTFS filesystems. This is achieved by setting the following key:

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem]
Value Name: NtfsDisableLastAccessUpdate
Data Type: REG_DWORD (DWORD Value)
Value Data: 1 (enable)

The purpose of the above facility serves to improve performance, and as such this loss of information could have a significant negative impact on security. It should be highlighted that in Windows Vista this setting is enabled by default and this is done possibly due to the pressing need for elevating the overall performance of a resource-demanding operating system like Vista. Nevertheless, in XP if the last access update key was updated with a disable value through a malicious payload as part of a combined integrity breach and slurping attack, although it would not be possible to enumerate the leaked files, the type and model as well as the time of the device could be identified, provided that no other unauthorised changes were made in the registry (or the timestamps of the changes were not modified by unauthorised means).

In a “real life” XP system we need to accept that uncertainty will be higher than that of a sterile, reference system used for empirically establishing the transfer rates and therefore there can be a high deviation on the actual transfer rates. Although the list of influencing factors cannot be complete, the following points are expected to impair the effectiveness of the proposed approach:

- file maintenance activities, such as scheduled backups. If a backup is performed at a time between the slurping attack and the detection attempt, then this method cannot be applied, if the backup application modifies the last access times.

- resource-intensive operations, such as antivirus checks. Not only an anti-virus will influence the transfer rates, if it happens to run during the attack, but if it is performed between the attack time and the detection attempt, it may spoil the last access times.

Nevertheless, since the bottleneck on the transfer speed is placed on the USB device, it can be expected that the (internal) hard disk and CPU activity will not assume any part of the bottleneck. However this assumption will require re-examination following the introduction of USB 3 which is announced to be a protocol with faster rates than USB 2. In addition, future research should focus on considering additional information such as antivirus scans and backup batch jobs which can be found in the relevant application logs in order to allow the analyst to construct the operational environment during the suspected slurping period and specify a more accurate and custom USB transfer footprint.

Although the prediction power of the resulting regression was reported to be high, the actual detection probability was rather average. This is due to the fact that there are overheads during the copying process which are not accounted for in the regression model, but exist in the synthesis of the transfer rate from the last access times. Future work includes the investigation of the detections via more advanced statistical methods such as probit and logit, or even by experimenting with other regression equations such as multinomial, Box-Cox and logistic. It is expected that these will allow the detection method to compensate for the hidden overheads and yield a higher accuracy for the attack probability.

5. Acknowledgement

The authors are indebted to Sophoclis Kesinis for his invaluable technical input and constructive feedback in all stages of this paper.

6. References

Hak 5, (2008), http://wiki.hak5.org/wiki//USB_Switchblade (Accessed 19 Jan 2009)

Radcliff, D. (2008), Slurping the USB port, SC Magazine, http://www.scmagazineus.com/Slurping-the-USB-port/article/115762 (Accessed 19 Jan 2009)

Rohde, L. (2004), iPods pose security risk for enterprises, Gartner says, http://www.infoworld.com/article/04/07/06/HNipodsrisk_1.html (Accessed 12 Jan 2009).

Thomas, P. & Morris, A. (2008). An Investigation into the Development of an Anti-Forensic Tool to Obscure USB Flash Drive Device Information on a Windows XP Platform. Third Annual Workshop on Digital Forensics and Incident Analysis, Malaga, Spain, 9 October, pp. 60-66.

Usher A. (2005) Pod slurping, Sharp Ideas LLC available from: http://www.sharp-ideas.net/pod_slurping.php (Accessed 12 Jan 2009).

U3, (2008), http://www.u3.com/ (Accessed 19 Jan 2009)

Verizon, (2008), 2008 Data Breach Investigations Report. http://www.verizonbusiness.com/resources/security/databreachreport.pdf (Accessed 19 Jan 2009)