Towards Fully Automated Digital Alibis with Social Interaction

Stefanie Beyer*, Martin Mulazzani*

Sebastian Schrittwieser§, Markus Huber*, Edgar Weippl*

* SBA Research, Vienna

[1stletterfirstname][lastname]@sba-research.org

§ University of Applied Sciences St. Pölten

sebastian.schrittwieser@fhstp.ac.at

Abstract

Digital traces found on local hard drives and in online activity have become the most important data source for the reconstruction of events in digital forensics. As such, digital alibis have already been used in court decisions and during investigations. In this paper we want to show that forged alibis can be created, which include online activity and social interactions. We are the first to use social interactions in digital alibis, and implemented a proof of concept that can be automated to create false digital alibis. Our framework simulates user activity, is fully automated and able to communicate using email as well as instant messaging using a chatbot. We evaluate our framework by extracting forensic artifacts and comparing them with results of real users in a user study.

Keywords: Digital Forensics, Automated Alibis

1 Introduction

Digital forensics techniques are nowadays applied to more and more criminal investigations due to the ever increasing prevalence of computers, smartphones and the involvement of modern technology in crimes. Traces like MAC-timestamps and OS-specific log files left on hard drives and information transmitted over network connections often are combined to produce a holistic reconstruction of events and for specific times of interest [3, 16]. Digital alibis as such are commonly used in reliable expert witness opinions to charge and discharge suspects in court, as well as during the investigation itself.

In this paper we present a framework that can fully simulate user interaction and implements the automated creation of digital alibis with special focus on online social interactions like writing emails and chatting with friends. Social interactions have been so far neglected in previous work. The framework is highly configurable, and we are planning to release it under an open source license[1]. Our framework is able to counter hard drive and network

[1]Note to the reviewers: we will include the link here once the paper is accepted for

forensics as it is conducted today. We evaluate our framework by comparing it with the usage of real world users, and show that digital forensic analysis methods are not reliable if they are specifically targeted. The goal of this paper is to raise awareness that digital alibis can be forged. We want to question the reliability of digital alibis, intend to show that digital alibis can be forged.

The rest of the paper is organized as follows: Section 2 gives an overview of the relevant research areas of digital forensics for this paper, as well as related work. Section 3 presents our framework which was implemented as proof-of-concept to show the feasibility to forge digital alibis. In Section 4, we compare and evaluate our framework compared with real-world users. We discuss the results in Section 5, and conclude in Section 6.

2 Background

Several cases can be found where evidence for digital alibis played an important role. In one case, Rodney Bradford was charged of armed robbery but was released because of digital evidence that confirms that he was performing activities on his Facebook account during the time of the crime [17]. This digital evidence was later called an "unbeatable alibi" [8] by his attorney. In another case, a suspected murderer was acquitted because of digital evidence [9, 7]. During the time of the crime, working activity was found on

publication

the suspect's laptop.

Usually a forensic analyst is confronted with multiple hard drives that have been imaged using hardware write blockers [2], and is asked specific questions about user actions that have or have not been conducted on the corresponding computers [4]. Modern communication technologies like online social networks [15] or smartphones [14, 13] can additionally increase the broad spectrum of digital traces. In the future, due to ever increasing storage capacity of consumer devices and overall case complexity, automated analysis will be crucial [12] to extract information of interest [11] in a reasonable amount of time.

2.1 Related Work

In the literature, several concepts for the analysis as well as the automatic construction of digital alibis can be found [10, 6] . However, existing alibi generators often use proprietary languages like AutoIt (Windows) or Applescript (OS X) and thus are specific to the underlying operating system that they run on e.g., Android [1], OS X [7], or Windows [10, 6]. Our framework does not rely on any os-specific components or software, and can thus run (with minor adaptions) on any Desktop operating system that runs Python. We implemented our prototype for Linux as there weren't any related frameworks available on Linux so far. We furthermore compared numerous configuration parameters with real world users, making our approach statistically harder to detect (with randomized values) and the persistently stored evidence more realistic compared to related work.

The previously described approaches try to hide the program, for instance with the help of harmless file names or on a separate storage device. Instead of using a file wiper for post processing to remove suspicious traces, our framework is constructed in such a way that there are no obvious traces left behind (despite the framework itself). For a forensic analyst, it should be not decidable if the artifacts on disc originate from the framework or a human user as we instrument (among other things) keyboard signals and mouse click events.

3 Alibi Generator Framework

The goal of our framework can be put simply: to simulate user activity as realistic and thorough as possible. To such extend, we want to simulate standard user activity: browsing the web, chatting with instant messaging software, writing emails and creating & editing documents of various kind. The concrete actions run by the framework should not be scripted or predictable, but randomized yet still realistic and convincing for a forensic investigator. Different word lists and online sources like Google Trends are used as inputs to capture a snapshot of current online interests with the need to incorporating specific user preferences at the same time. Many factors of computer usage are highly user dependent, and for the alibi framework to be as realistic as possible, factors like common online social interaction partners for chatting and email, language of communication as well as time delays between actions and usual concurrent actions need to be configured beforehand. Social interaction in particular is vulnerable to traffic analysis, as not only the content of the messages is of interest, but who is communicating with whom. The response time is also dependent on the length of the messages, which needs to be considered when simulating social interactions.

3.1 Implementation

Our proof-of-concept was implemented on Ubuntu 12.04 LTS using Python. The core features of the framework were chosen similar to the approaches in [7, 6]. In essence our implementation is split into three main components: the *scheduler*, the *program manager* and the *social interaction component*. Once the framework is started, the *scheduler* is in charge of overall management, it controls startup and triggers shutdown of all programs involved. It has the purpose of deciding which actions to take, either local or online, and when. The *program manager* runs and manages all applications, including the browser, the email- and chat software. The *social interaction component* consists of a chatbot for instant messaging and the email manager, responsible for email communications.

The framework can start and use local applications by sending key strokes and mouse clicks. Our framework comes pre-configured to use and handle the following applications in a meaningful manner: Firefox, gedit, LibreOffice, Thunderbird, Skype, and VLC. For automation the Python libraries *xautomation*[2], *skype4py*[3] and the chatbot implementation *pyAIML*[4] are used. Furthermore, we use *splinter*[5] for the automation of Firefox, which is based on *Selenium*[6]. Thus our implementation can browse the web, send and receive emails, chat in Skype, open and edit documents (LibreOffice and gedit) and start programs like music or video players. Figure 1 shows the main features of our proof-of-concept. Related features which are not yet implemented are marked in grey. Frequency and content of alibi events were derived from the analysis of several typical office workstations at our university.

More specifically, our framework queries Google and follows suggested links, tweets on Twitter and logs into Facebook, can search for Youtube videos and browses websites with random mouse clicks and following links. New emails are drafted, received emails are forwarded, and answers to emails are sent with a reasonable delay. Additionally it is possible to mark new emails as read and delete emails. The action to be performed is chosen at random, not every email that is read will be replied to. The subject and content of emails can be predefined and stored in lists. Regarding instant

[2]http://hoopajoo.net/projects/xautomation.html
[3]http://sourceforge.net/projects/skype4py
[4]http://pyaiml.sourceforge.net
[5]http://splinter.cobrateam.info/
[6]http://www.seleniumhq.org

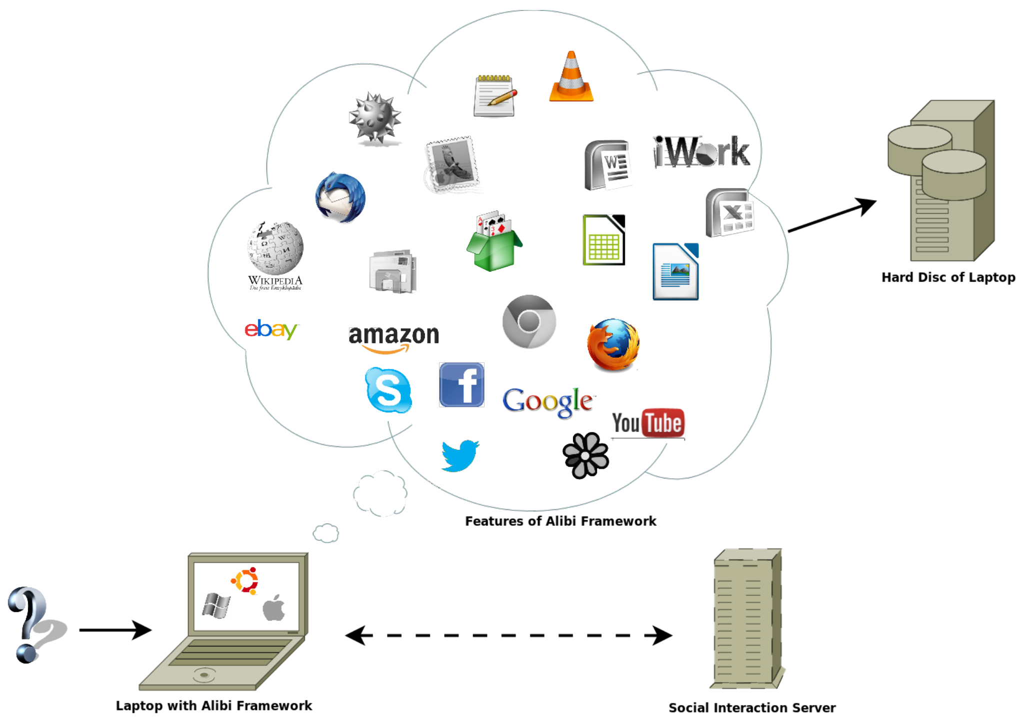
Hard Disc of Laptop
Features of Alibi Framework
Laptop with Alibi Framework
Social Interaction Server
iWork
WIKIPEDIA
ebay
amazon
Google
YouTube

Figure 1: Conceptional Alibi Framework

messaging answering to incoming messages of buddies is supported, with the help of a chatbot. Reasonable time delays depending on the size of the response or on a random delay are implemented. Chat templates can be easily adapted due to the use of AIML [18]. If the timer of the scheduler expires, the chatbot says goodbye to his buddies and shuts down. Editing of local documents is implemented either by deleting a random amount of content, inserting predefined texts which fits the content of document at a random position. We implemented the use of LibreOffice by simulating key strokes and mouse clicks, as there are no Python bindings for LibreOffice available.

However, one post-processing step is necessary: Splinter has to use a seperate firefox profile, and can not work on the user's profile directly. On startup splinter copies the user's profile to /tmp, and we overwrite the user's profile on shutdown by moving it back. Depending on the user's threat model, additional steps might be appropriate.

4 Evaluation

To evaluate our implementation we compare the framework with real world users. Since the particular usage is highly dependent on the person using it, this evaluation is intended to show that the default configuration is reasonable. We asked nine volunteers to use a virtual machine for 30 minutes just like they usually use their computer, and compared it with a single run of our framework. They were asked to browse the web, send emails, chat, edit documents or do anything they usually would do in their own environment. Forensic analysis was applied afterwards on the image of the virtual machine using *Sleuth Kit* and *Autopsy*, to extract data in a forensic manner. For that, the timestamps of the entire file system are extracted, as well as files that contain user data of interest are manually inspected. Manual analysis was conducted on the browser history *places.sqlite* of Firefox, the local mailbox files by Thunderbird, and the chat history *main.db* of Skype to extract timestamps, message content, conversation partner and the time interval between messages, emails and visited websites. Network forensics would have been an alternative approach for evaluation, by inspecting the network traffic. However, hard drive analysis allows to extract unencrypted as well as additional information like local timestamps, and has thus been the evaluation method of choice. We then used different metrics to compare our framework with real world users during the half hour period e.g., the number of visited websites, duration of visit on websites and the number of chat messages sent and received.

Figure 2 shows the exemplary timeline of 20 minutes framework runtime. The social interaction can be clearly observed, in total 12 messages are sent from the social interaction component to various recipients, either as responses in ongoing conversations or as new messages to initialise conversations. The browser is directed to different websites and follows links to simulate further browsing (not shown in the figure). This includes news sites like nytimes.com and prominent news stories on the front page, youtube.com and videos from the "most popular" section, as well as the top videos from random search queries. Local timestamps as well as history logs are written to disk in all cases. Furthermore VLC is started and a local video from the hard drive is opened, which is also reflected in the timestamps on disk.

4.1 User Survey

Regarding the browsing behavior of the user group, target websites and time patterns have been extracted. The most popular websites have been

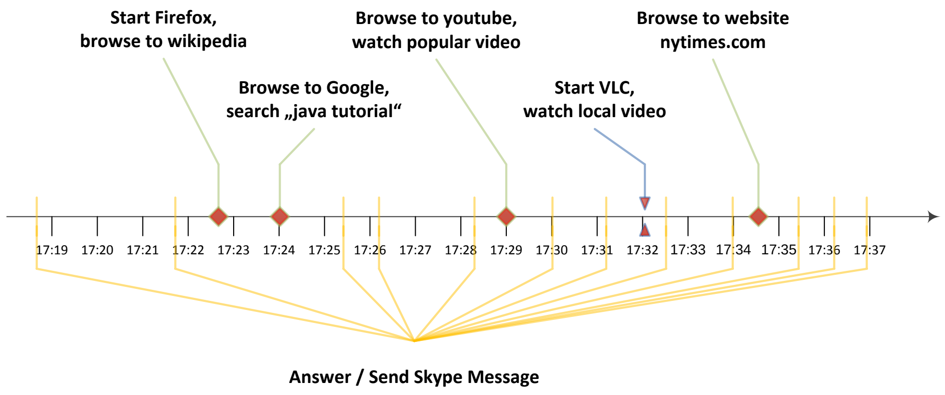
Start Firefox, browse to wikipedia
Browse to youtube, watch popular video
Browse to website nytimes.com
Browse to Google, search „java tutorial“
Start VLC, watch local video
17:19
17:20
17:21
17:22
17:23
17:24
17:25
17:26
17:27
17:28
17:29
17:30
17:31
17:32
17:33
17:34
17:35
17:36
17:37
Answer / Send Skype Message

Figure 2: Exemplary Timeline of 20 Minutes Framework Activity

google.com, facebook.com as well as some Austrian news sites. The framework on the other hand used a preconfigured list of websites as well as randomized google queries. The five most visited websites however matched in 4 out of 5 cases between the framework and the users. Some time patterns that were extracted can be seen in Table 1. The test users did not receive any emails due to our setup, but were asked to send some. On average, one email was sent per user. The maximum number of sent emails by the users was 3, the minimum 0. The majority of test persons did not write an email. The words in an email vary in number between 6 and 51 words, the average number of words is 21. The time between sending emails varies between 1 minute and 15 minutes.

Nearly all test persons did chat during their session. There are between 7 and 46 outgoing messages and between 15 and 35 incoming messages. The

Browsing parameters	*Framework*	*Test Persons*
# Visited Websites	11	min: 1 max: 12 avg: 9
Time on Website (min)	8 sec	5 sec
Time on Website (max)	2 min 16 sec	13 min 05 sec
Time on Website (avg)	1 min 52 sec	2 min 50 sec

Table 1: Comparison of browsing patterns

shortest message is 1 word per message for each person. The highest number of words in a chat message is 23 words. The topics in chat messages depend strongly on the test person, there were topics such as health and small talk as well as football or movies. The response time of chat messages was at least 2 seconds and at most 8 minutes. The average response time is about 1 minute and 4 seconds. See Table 2 for details. The users furthermore edited or opened one document (.ods or .pdf) during the 30 minute timeframe, which is also consistent with the actions of the framework.

Chatting Parameters	*Framework*	*Test Persons*
Outgoing chat messages	22	(7/46/19)
Incoming chat messages	23	(15/35/21)
Length of chat messages	(1/18/8)	(1/23/4)
Response time	(2s/175s/45s)	(2s/480s/64s)

Table 2: Observed chat behavior (min/max/avg)

5 Discussion

Regarding the browsing habits of the framework we could show that the number of visited websites is reasonable compared to the results from the test persons. The time on each website is on average shorter than the time of the test, but this is a parameter which can be easily changed (just like the number of websites to visit). Some test persons stayed more than 10 minutes on one site, but in general the simulator fits into the timely patterns of the test persons. 4 of 5 of the most visited websites of the simulator are equal to the most visited websites of the test persons, which is a very good result as the websites were a-priori configured. However, the sites visited actually per person, depend strongly on user's preferences and have to be adapted. In summary we can say that the surfing feature is properly simulated by framework, but needs a user-specific configuration. Table 3 shows the overall comparison between the alibi framework and the values of the test persons.

Regarding chat and using the social interaction server, we were able to observe that the response time to answer chat messages fits the expected time frame. The framework does not response to every message, and the time it waits for sending a response is within the observations from the real users.

5.1 Limitations and Future Work

One limitation of our prototype is the lack of sophisticated contextual analysis of instant messages. While AIML can be used to generate genuine-looking conversations for short periods of time, a forensic analysis of extended conversations probably would reveal the chatbot. While this is definitely a problem in scenarios where the disk is analyzed forensically, generated alibis would

Feature	*Simulator*	*Test persons*		
visited homepages	11	min: 1	max: 12	✓
time on homepage	1m 52s	min: 5s	max: 2m 50s	✓
most visited homepages		matching in 4/5 sites		✓
emails (out)	1	min: 0	max: 3	✓
emails (in)	2	min: 0	max: 0	✗
length of emails (words)	6	min: 6	max: 51	~
content of emails		matching in 1/4 topics		✓
chat messages (out)	22	min: 7	max: 46	✓
chat messages (in)	23	min: 15	max: 35	✓
length in words	avg: 8	min: 1	max: 23	✓
response time	avg: 45s	min: 2s	max: 1m 4s	✓
content of conversation		matching in 2/5 topics		✓
opened documents	1	min: 0	max: 2	✓
document type	`.ods`	`.ods`, `.odt`, `.pdf`		✓

Table 3: Overall comparison framework vs. survey users

most likely withstand network forensic analysis because most protocols such as Skype implicitly encrypt messages anyway. For unencrypted email conversations, this limitation is more important. In future work, we thus aim at identifying methods for more content-dependent responses.

Another limitation is adaptivity. To forge a digital alibi reliably it is necessary to adapt the framework to the user's preferences. For comparison of efficiency of the framework, the values, ranges and preferences of results of test systems were taken for reference values. To fit the typical system usage of an individual, there are several possibilities to adapt the framework. Currently, most parameters of the framework are configured manually and have

to be adapted for each user. In the future the framework should be able to adapt those parameters automatically. This may be realized by either collecting the user's specific information from user data or by collecting it over a longer period as a learning phase. We would also like to compare long-term runs of the framework with real user data, as 30 minutes is not particularly long enough for all use cases where a digital alibi might be needed. Testing and supporting other operating systems as well as other browsers is also a goal for the future.
If the user has insufficient knowledge of the tools or the system it may happen that there is unwanted evidence left behind. It is essential to continuously upgrade and improve the framework, because operating systems and techniques are constantly changing as well. We did not implement any obfuscation or methods to hide the execution of the framework. Running it from external media as suggested in [5] should be for example just one item in the list of additional steps, as these techniques may strengthen the validity of a forged alibi.

6 Conclusion

In conclusion, we were able to show that it is possible to forge digital alibis with social interaction. We implemented a proof of concept and showed in the evaluation that the results of forensic analysis of our framework meet the range of values observed from real users. The framework has the ability - if it is configured correctly - to simulate browsing, write and respond to emails, chat with buddies, and opening and/or editing documents. The exact execution of the simulation depends on the configurations that should be adapted to user's preferences. In order to show that the behavior simulated by the framework differs from the actual user's behavior, a very intimate knowledge of the user's habits and usage patterns is necessary. In the future, we want to add a component for automatic configuration based on either existing log files, or use an additional learning phase to obtain an user-specific configuration that is even more indistinguishable with respect to a forensic investigator.

References

[1] P. Albano, A. Castiglione, G. Cattaneo, G. De Maio, and A. De Santis. On the construction of a false digital alibi on the android os. In *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on*, pages 685–690. IEEE, 2011.

[2] D. Brezinski and T. Killalea. Guidelines for evidence collection and archiving. *Request For Comments*, 3227, 2002.

[3] F. P. Buchholz and C. Falk. Design and implementation of zeitline: a forensic timeline editor. In *DFRWS*, 2005.

[4] B. Carrier. *File system forensic analysis*, volume 3. Addison-Wesley Boston, 2005.

[5] A. Castiglione, G. Cattaneo, G. De Maio, and A. De Santis. Automatic, selective and secure deletion of digital evidence. In *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 International Conference on*, pages 392–398. IEEE, 2011.

[6] A. Castiglione, G. Cattaneo, G. De Maio, A. De Santis, G. Costabile, and M. Epifani. The forensic analysis of a false digital alibi. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 114–121. IEEE, 2012.

[7] A. Castiglione, G. Cattaneo, R. De Prisco, A. De Santis, and K. Yim. How to forge a digital alibi on Mac OS X. *Multidisciplinary Research and Practice for Information Systems*, pages 430–444, 2012.

[8] CNN. Facebook status update provides alibi. (november 12, 2009), http://edition.cnn.com/2009/crime/11/12/facebook.alibi/index.html, June 2009.

[9] X. A. W. Community. Garlasco, alberto stasi acquitted (december 2009), http://richardseifer.xomba.com/garlasco_alberto_stasi_acquitted, June 2013.

[10] A. De Santis, A. Castiglione, G. Cattaneo, G. De Maio, and M. Ianulardo. Automated construction of a false digital alibi. *Availability, Reliability and Security for Business, Enterprise and Health Information Systems*, pages 359–373, 2011.

[11] S. L. Garfinkel. Digital forensics research: The next 10 years. *Digital Investigation*, 7:S64–S73, 2010.

[12] S. L. Garfinkel. Digital media triage with bulk data analysis and bulk_extractor. *Computers & Security*, 2012.

[13] A. Hoog. *Android forensics: investigation, analysis and mobile security for Google Android.* Access Online via Elsevier, 2011.

[14] A. Hoog and K. Strzempka. *iPhone and iOS Forensics: Investigation, Analysis and Mobile Security for Apple iPhone, iPad and iOS Devices.* Elsevier, 2011.

[15] M. Huber, M. Mulazzani, M. Leithner, S. Schrittwieser, G. Wondracek, and E. Weippl. Social snapshots: Digital forensics for online social networks. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 113–122. ACM, 2011.

[16] J. Olsson and M. Boldt. Computer forensic timeline visualization tool. *digital investigation*, 6:S78–S87, 2009.

[17] T. N. Y. Times. I'm innocent. Just check my status on facebook. (november 12, 2009),

http://www.nytimes.com/2009/11/12/nyregion/12facebook.html?_r=2&, June 2013.

[18] R. Wallace. The elements of aiml style. *Alice AI Foundation*, 2003.