Chapter 4

TOWARDS FULLY AUTOMATED DIGITAL ALIBIS WITH SOCIAL INTERACTION

Stefanie Beyer, Martin Mulazzani, Sebastian Schrittwieser, Markus Huber and Edgar Weippl

Abstract Digital traces found on local hard drives as a result of online activities have become very valuable in reconstructing events in digital forensic investigations. This paper demonstrates that forged alibis can be created for online activities and social interactions. In particular, a novel, automated framework is presented that uses social interactions to create false digital alibis. The framework simulates user activity and supports communications via email as well as instant messaging using a chatbot. The framework is evaluated by extracting forensic artifacts and comparing them with the results obtained from a human user study.

Keywords: Digital evidence, automated alibis

1. Introduction

Digital forensic techniques are increasingly applied in criminal investigations due to the widespread involvement of computers, smartphones and other modern technologies in crimes. Traces such as MAC timestamps and operating system specific log files left on hard drives and information transmitted over network connections are often combined to produce a holistic reconstruction of events for specific times of interest [4, 16]. The resulting digital alibis are routinely presented and examined during investigations and in court proceedings.

This paper describes a framework that fully simulates user interactions and implements the automated creation of digital alibis with special focus on online social interactions such as writing email and sending chat messages. The framework is unique because it engages social interactions that have been ignored in previous work. Additionally, the

framework is highly configurable and is available under an open source license (`github.com/mmulazzani/alibiFramework`).

This paper evaluates the framework by comparing it with usage patterns of real world users, demonstrating that digital forensic analysis methods are not reliable if they are specifically targeted. The goal is to raise awareness in the digital forensics community that digital alibis can be forged and, consequently, it is important to always question the reliability of digital alibis.

2. Background

Digital alibis have played an important role in numerous cases. In one case, Rodney Bradford was charged with armed robbery but was released because digital evidence demonstrated that he was actively using his Facebook account at the time of the crime [15]. His attorney noted that the digital evidence gave Bradford an "unbeatable alibi" [2]. In another case, a suspected murderer was acquitted because digital evidence revealed that he was working on his laptop when the murder took place [7, 17].

A digital forensic analyst is often confronted with multiple hard drives that have been imaged using hardware write blockers [3], and is asked specific questions about user actions that have been or have not been conducted on the associated computers [5]. Meanwhile, the widespread use of modern technologies and devices such as social networks [14] and smartphones [12, 13] dramatically increase the amount of digital traces that must be considered even in routine cases. The massive quantity of digital evidence and the accompanying case complexity render automated analysis crucial to extracting information of interest in a reasonable amount of time [10, 11].

Several researchers have focused on the creation digital alibis [8, 9]. However, their alibi generators often use proprietary languages such as AutoIt (Windows) or Applescript (OS X) and are, therefore, specific to the underlying operating system (e.g., Windows [8, 9], OS X [7] or Android [1]). Our framework, on the other hand, is not so operating system specific because it is implemented in Python. Although the implementation was developed for Linux systems because no Linux-specific solutions existed, it is a simple matter to port it to other platforms. Additionally, the configuration parameters were set based on extensive studies of real world users, rendering the alibi generation approach statistically more difficult to detect and the persistently-stored evidence more realistic compared with other approaches.

While other alibi creation approaches attempt to hide their programs using innocuous file names or separate storage devices, our framework is designed to leave no obvious traces. Also, the framework does not employ a file wiper during the post processing phase to remove suspicious traces. The rationale is that a forensic analyst should not be able to determine if the digital evidence artifacts originated from the framework or from a human user. This is accomplished by instrumenting keyboard signals, mouse clicks and other events within the alibi creation framework.

3. Alibi Creation Framework

The alibi creation framework is intended to simulate user activity as realistically and as thoroughly as possible. To this end, it is necessary to simulate standard user activities such as browsing the web, communicating via email, chatting with instant messaging software and editing documents of various types. The concrete actions performed by the framework should be neither scripted nor predictable; they should be randomized, but still realistic and convincing to digital forensic investigators.

A variety of word lists and online sources such as Google Trends are used as inputs to capture a snapshot of current online interests while simultaneously incorporating specific user preferences. Many facets of computer usage are highly user dependent. For the alibi creation framework to be as realistic as possible, factors such as the social interaction partners for email and chatting, the language of communication and the time delays between actions and usual concurrent actions must be configured in advance. Social interactions, in particular, are vulnerable to traffic analysis because the content of the messages as well as the identities of the communicating parties are of interest. Also, the response times are dependent on message length, and must be considered carefully when simulating social interactions.

The proof-of-concept system was implemented on Ubuntu 12.04 LTS using Python. The core features of the framework were chosen in a similar manner as other approaches in the literature [7, 8].

The implementation has three main components: (i) scheduler; (ii) program manager; and (iii) social interaction component. The scheduler is responsible for the overall management; it controls startup and triggers the shutdown of all the involved programs. Also, it decides which actions to perform (both local and online) and when to perform them. The program manager runs and manages all applications, including the browser and the email and chat software. The social interaction

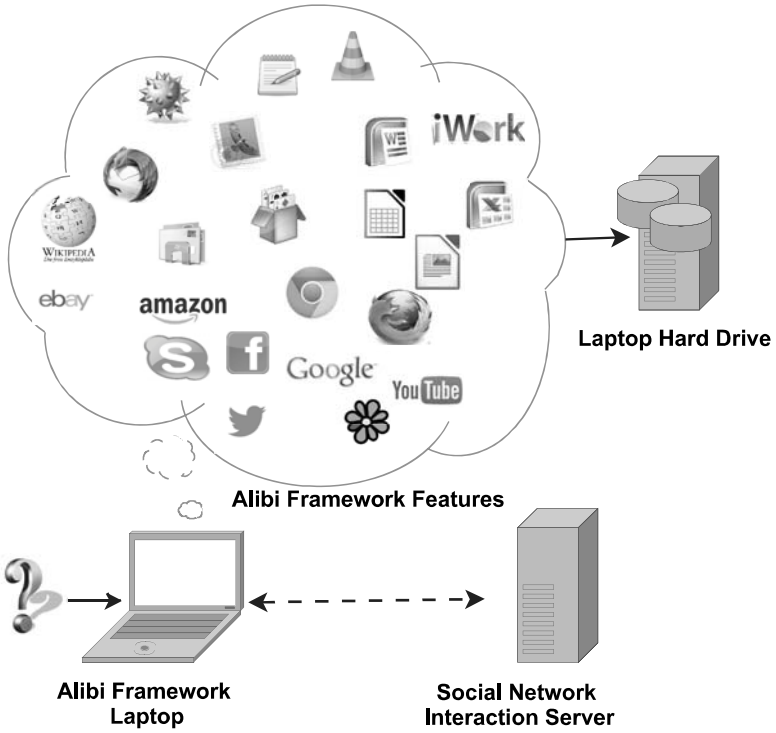

Figure 1. Conceptual view of the alibi creation framework.

component incorporates the email manager and a chatbot for instant messaging.

Figure 1 shows the main components of the alibi creation framework. The framework can launch and use local applications by sending key strokes and mouse clicks. It comes pre-configured for the use of several applications: Firefox, `gedit`, LibreOffice, Thunderbird, Skype and VLC. The Python libraries `xautomation`, `skype4py` and the chatbot implementation `PyAIML` are used for automation. Furthermore, `splinter` (based on Selenium) is used for the automation of Firefox. Thus, the implementation can browse the web, send and receive email, chat using Skype, open and edit documents (using LibreOffice and `gedit`) and launch programs such as music players and video players. The frequency and content of alibi events were derived from the forensic analysis of several typical office workstations at our university.

The framework can query Google and follow suggested links, tweet on Twitter and log into Facebook. Also, it can search for YouTube videos and browse websites with random mouse clicks and follow links. New email messages are drafted, received email messages are forwarded,

and email responses are sent after reasonable delays. Additionally, it is possible to mark new email messages as read and to delete email messages. The action to be performed is chosen at random; not every email that is read is answered. The subject and content of email can be predefined and stored in lists. The answering of instant messages is supported with the help of a chatbot. Reasonable time delays are implemented based on the response message size and by using random delays. Chat templates are easily adapted via AIML [18]. When the timer of the scheduler expires, the chatbot says goodbye and shuts down. The editing of local documents is implemented by deleting a random amount of content or by randomly inserting predefined text that fits the content of the document. The use of LibreOffice is implemented by simulating key strokes and mouse clicks because no Python bindings for LibreOffice are available.

However, one post-processing step is necessary: Splinter has to use a separate Firefox profile and cannot work directly with the user profile. Therefore, during startup, `splinter` copies the user profile to `/tmp` and the user profile is overwritten at shutdown by moving it back. Depending on the user threat model, additional steps might be appropriate.

4. Evaluation

The implementation was evaluated by comparing its behavior with that of human users. Since the usage of a computer depends heavily on the person using it, the evaluation of the implementation focused on demonstrating that the default configuration is reasonable.

Nine human test subjects were asked to use a virtual machine for 30 minutes just like they would use their computers, and the results were compared with a single run of the framework. The test subjects were asked to browse the web, send email, chat, edit documents and to do anything they would normally do.

Sleuth Kit and Autopsy were subsequently applied to the virtual machine images to extract data in a forensic manner. Timestamps were extracted and the files containing user data of interest were inspected manually. Manual analysis was conducted on the browser history *places.sqlite* of Firefox, the local mailbox files of Thunderbird, and the chat history *main.db* of Skype to extract timestamps, message content, conversation partners and the time intervals between messages, email and website visits. Network forensics, namely the inspection of network traffic, would have been an alternative approach for evaluation. However, hard drive analysis allows unencrypted information as well as additional information such as local timestamps to be extracted; this motivated the use of

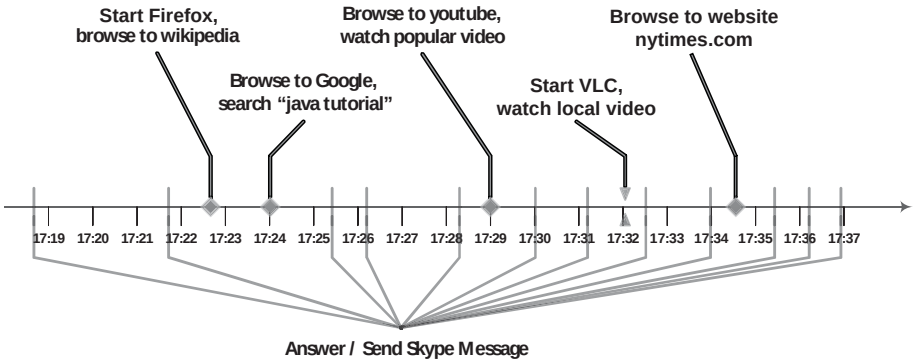

Figure 2. Example timeline of 20 minutes of framework activity.

the evaluation method. Various metrics were used to compare the framework behavior with that of real users during the 30 minute test period: number of visited websites, durations of website visits and numbers of chat messages sent and received.

Figure 2 shows an example timeline of 20 minutes of framework activity. The social interactions can be observed clearly. A total of twelve messages were sent by the social interaction component to various recipients, either as responses in ongoing conversations or as new messages to initiate conversations. The browser was directed to several websites and links were followed to simulate further browsing (not shown in the figure). The browsing activities accessed news sites such as `nytimes.com` and prominent news stories on the front page; and `youtube.com` and top videos from the "most popular" section based on random search queries. Local timestamps and history logs were written to disk in all cases. Furthermore, VLC was started and a local video from the hard drive was opened; this is reflected in the timestamps shown in Figure 2.

5. Test User Survey

Target websites and time patterns were extracted to capture the browsing behavior of the nine volunteers; the most popular websites visited were `google.com`, `facebook.com` and various Austrian news sites. On the other hand, the framework used a preconfigured list of websites and randomized Google queries. Nevertheless, four of the five most visited websites by the framework and the users matched. Table 1 shows three extracted time patterns. The test users did not receive any email as a result of the experimental setup, but they were asked to send email messages. On the average, one email message was sent per user. The maximum number of email messages sent by the users was three and the

Table 1. Website visit comparison.

	Framework	**Users**
Number of 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 5 sec
Time on Website (Avg)	1 min 52 sec	2 min 50 sec

minimum was zero. The number of words in an email varied between six and 51, with an average of 21. The time between sending email varied between one minute and fifteen minutes.

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

	Framework	**Users**
Chat Messages Sent	22	(7/46/19)
Chat Messages Received	23	(15/35/21)
Chat Message Length	(1/18/8)	(1/23/4)
Response Time (sec)	(2/175/45)	(2/480/64)

Table 2 provides details about the observed chat behavior. Nearly all the test users conducted chat conversations. There were between seven and 46 outgoing messages and between fifteen and 35 incoming messages. The shortest message was one word long for each user. The maximum number of words in a chat message was 23. The chat message topics depended strongly on the test users; they included health, football and movies, as well as small talk. The response time for chat messages was at least two seconds and at most eight minutes. The average response time was about one minute and four seconds. The users edited or opened one document (`.ods` or `.pdf`) during the 30 minute timeframe, which was consistent with the actions of the framework.

6. Discussion

Table 3 compares the behavior of the alibi framework with that of the test subjects. The browsing behavior of the framework in terms of the number of websites visited is comparable to that of the test users. The time spent on each website is on the average shorter than the time spent by the test users, but this is a parameter that is easily changed in the framework (just like the number of websites to be visited). Some test users spent more than ten minutes at a site but, in general, the time spent by framework on websites generally matches the time spent by the

Table 3. Overall comparison framework vs. survey users.

Feature	Framework	Users	
Websites Visited	11	Min: 1	Max: 12
Website Visit Time	1 min 52 sec	Min: 5 sec	Max: 2 min 50 sec
Most Visited Websites		Matching in 4/5 sites	
Email Messages (sent)	1	Min: 0	Max: 3
Email Messages (received)	2	Min: 0	Max: 0
Email Message Length (words)	6	Min: 6	Max: 51
Email Message Content		Matching in 1/4 topics	
Chat Messages (sent)	22	Min: 7	Max: 46
Chat Messages (received)	23	Min: 15	Max: 35
Chat Message Length (words)	Avg: 8	Min: 1	Max: 23
Response Time	Avg: 45 sec	Min: 2 sec	Max: 1 min 4 sec
Conversation Content		Matching in 2/5 topics	
Opened Documents	1	Min: 0	Max: 2
Document Types	`.ods`	`.ods`, `.odt`, `.pdf`	

test users. Four of the five most visited websites for the framework and the test users matched, which is a very good result because the websites visited by the framework were configured *a priori*. However, the actual sites visited by the test users depended strongly on user preferences; this feature will have to be adjusted in the framework. In summary, the framework adequately simulates web surfing activities, but it requires the implementation of a user-specific configuration feature.

With regard to chat conversations and the use of the social interaction server, the response time for chat messages matches the expected response time. The framework adequately models the behavior of test users in terms of the response time delays and the fact that not every message is answered.

7. Limitations and Future Work

One limitation of the 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 the conversations would likely reveal the use of a chatbot. While this is definitely a problem in scenarios where the disk is analyzed forensically, the generated alibis would pass network forensic analysis because most protocols (e.g., Skype) implicitly encrypt messages. This limitation is more significant for unencrypted email conversations. Our future work

will attempt to identify methods for creating better content-dependent responses.

Another limitation is adaptivity. To forge a digital alibi in a reliable manner, it is necessary to adapt the framework to user preferences. Currently, most of the framework parameters are configured manually and have to be adapted for each user. Ideally, the framework should be able to adapt the parameters automatically. This can be realized by collecting user-specific information from user data or by collecting it over a longer period during a learning phase. It would also be useful to compare long-term runs of the framework with real user data – 30 minutes is insufficient to cover all the use cases where digital alibis might be needed. Another future goal for the framework is supporting other operating systems and browsers.

An important point to note is that a user who has insufficient knowledge of the tools and the system could leave undesirable evidence. Therefore, it is essential to continuously update and refine the framework as operating systems and applications change. The current framework does not implement any obfuscation methods to hide its execution. Running the framework from external media as suggested in [6] could help strengthen the validity of a forged alibi.

8. Conclusions

The proof-of-concept alibi creation framework demonstrates that it is possible to forge digital alibis that adequately model real user behavior. The framework has the ability – if correctly configured – to simulate browsing, emailing, chatting and document editing behavior. The precise simulation that is obtained depends on the configurations that should, of course, be adapted to user preferences. This requires an intimate knowledge of user behavior and usage patterns. Our future research will focus on adding an automated configuration feature based on existing log files or learning user-specific behavior, with the goal of rendering the framework behavior more indistinguishable from normal human behavior, even after analysis by experienced forensic investigators.

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, *Proceedings of the Third International Conference on Intelligent Networking and Collaborative Systems*, pp. 685–690, 2011.

[2] D. Beltrami, I'm innocent. Just check my status on Facebook, *New York Times*, November 11, 2009.

[3] D. Brezinski and T. Killalea, Guidelines for Evidence Collection and Archiving, RFC 3227, 2002.

[4] F. Buchholz and C. Falk, Design and implementation of Zeitline: A forensic timeline editor *Proceedings of the Fifth Digital Forensic Research Workshop*, 2005.

[5] B. Carrier, *File System Forensic Analysis*, Addison-Wesley, Upper Saddle River, New Jersey, 2005.

[6] A. Castiglione, G. Cattaneo, G. De Maio and A. De Santis, Automatic, selective and secure deletion of digital evidence, *Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications*, pp. 392–398, 2011.

[7] A. Castiglione, G. Cattaneo, G. De Maio, A. De Santis, G. Costabile and M. Epifani, How to forge a digital alibi on Mac OS X, *Proceedings of the IFIP WG 8.4/8.9 International Cross Domain Conference and Workshop on Availability, Reliability and Security*, pp. 430–444, 2012.

[8] A. Castiglione, G. Cattaneo, G. De Maio, A. De Santis, G. Costabile and M. Epifani, The forensic analysis of a false digital alibi, *Proceedings of the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 114–121, 2012.

[9] A. De Santis, A. Castiglione, G. Cattaneo, G. De Maio and M. Ianulardo, Automated construction of a false digital alibi, *Proceedings of the IFIP WG 8.4/8.9 International Cross Domain Conference and Workshop on Availability, Reliability and Security*, pp. 359–373, 2011.

[10] S. Garfinkel, Digital forensics research: The next 10 years, *Digital Investigation*, vol. 7(S), pp. S64–S73, 2010.

[11] S. Garfinkel, Digital media triage with bulk data analysis and `bulk_extractor`, *Computers and Security*, vol. 32, pp. 56–72, 2012.

[12] A. Hoog, *Android Forensics: Investigation, Analysis and Mobile Security for Google Android*, Syngress, Waltham, Massachusetts, 2011.

[13] A. Hoog and K. Strzempka, *iPhone and iOS Forensics: Investigation, Analysis and Mobile Security for Apple iPhone, iPad and iOS Devices*, Syngress, Waltham, Massachusetts, 2011.

[14] M. Huber, M. Mulazzani, M. Leithner, S. Schrittwieser, G. Wondracek and E. Weippl, Social snapshots: Digital forensics for online social networks, *Proceedings of the Twenty-Seventh Annual Computer Security Applications Conference*, pp. 113–122, 2011.

[15] V. Juarez, Facebook status update provides alibi, *CNN*, November 13, 2009.

[16] J. Olsson and M. Boldt, Computer forensic timeline visualization tool, *Digital Investigation*, vol. 6(S), pp. S78–S87, 2009.

[17] R. Seifer, Garlasco, Alberto Stasi acquitted, *Xomba* (`richardseifer.xomba.com/garlasco_alberto_stasi_acquitted`), December 13, 2004.

[18] R. Wallace, The Elements of AIML Style, A.L.I.C.E. AI Foundation, 2003.