Journal of Digital Forensics, Security and Law

Volume 16 Article 1

2-16-2021

Social Media User Relationship Framework (SMURF)

Anne David
Cranfield University, a.david@cranfield.ac.uk

Sarah Morris
Cranfield University, s.l.morris@cranfield.ac.uk

Gareth Appleby-Thomas
Cranfield University, g.thomas@cranfield.ac.uk

Recommended Citation

David, Anne; Morris, Sarah; and Appleby-Thomas, Gareth (2021) "Social Media User Relationship Framework (SMURF)," *Journal of Digital Forensics, Security and Law*: Vol. 16 , Article 1.
DOI: https://doi.org/10.15394/jdfsl.2021.1698
Available at: https://commons.erau.edu/jdfsl/vol16/iss1/1

SOCIAL MEDIA USER RELATIONSHIP FRAMEWORK (SMURF)

Anne David[1], Sarah Morris[2], Gareth Appleby-Thomas[3]
[1,2]Centre for Electronic Warfare, Information, and Cyber
[3]Centre for Defence Engineering
Cranfield University, Shrivenham, UK
{a.david,s.l.morris,g.thomas}@cranfield.ac.uk

ABSTRACT

The use of social media has spread through many aspects of society, allowing millions of individuals, corporate as well as government entities to leverage the opportunities it affords. These opportunities often end up being exploited by a small percentage of the user community who use it for objectionable or unlawful activities; for example, trolling, cyber bullying, grooming, luring. In some cases, these unlawful activities result in investigations where the swift retrieval of critical evidence is required in order to save a life. This paper presents a proof of concept (PoC) framework for social media user attribution. The framework aims to provide digital evidence that can be used to substantiate user activity in live triage investigations. This paper highlights the use of live triage as a viable technique for the investigation of social media activity, contextualizing user activity and attributing actions to users. It discusses the reliability of artefacts other than the communication content as a means of drawing inferences about user social media activity, taking into account.

Keywords: digital forensics, digital investigation, digital investigation framework, live triage, relationship attribution, social media

1. INTRODUCTION

Social media presents investigators with a plethora of useful digital evidence as shown by a range of criminal prosecutions and convictions as well as civil investigations with outcomes such as disciplinary measures. Thus, the timely recovery of these artefacts is considered of interest to investigators during a live triage.

Triage as a practice emerged in the field of medicine where it was used to assess hospital patients in order to establish the order of prioritization of treatment Robertson-Steel (2006). In digital forensics however, triage is a fast-growing practice with two streams; live and post-mortem triage.

In time sensitive investigations, the need to quickly obtain actionable intelligence, in a usable format from a live system Wiles and Reyes (2007) may take precedence over waiting to conduct a detailed forensic analysis. This paper proposes a proof of concept (PoC) framework for social media user attribution, to support investigators through the automated, targeted recovery artefacts from social media activity in a live triage.

This paper is focused on social media access through a web browser and aims to highlight the potential in using artefacts other than communication content to infer and contextualise user activity, and to attribute actions to a user. The proposed framework is intended to be applicable and generalisable to other triage investigations. The web browser(s) referenced in this paper is a demonstration for the framework.

1.1 Contribution

The key contributions of this paper are as follows:

- ⇒ the proposal of a PoC framework for live triage investigations involving social media
- ⇒ defines a baseline for the artefacts of interest as a method for grouping social media artefacts based on the perceived user activity. This can be used in the prioritization of live triage data analysis
- ⇒ a demonstration to test its applicability, generalisability and reliability through the identification of how SMURF can extend or complement existing triage tools

1.2 Paper Structure

The rest of this paper is structured as follows: related work is discussed in Section 2; SMURF is introduced in Section 3, discussing the problem definition, characteristics and development. Section 4 discusses the build process and how SMURF works. In Section 5, the PoC implementation is discussed including data generation, analysis and the PoC deployment. Section 6 presents a case study and reliability assessment. Section 7 contains the discussion, and Section 8 presents the conclusion and future work.

2. LITERATURE REVIEW

This section provides a background on related work in the investigation of social media activity, the rules and reliability of digital evidence, and live triage investigations. It discusses the challenges faced by practitioners in the course of investigating user social media activity, and acquiring digital evidence that can be used for the attribution of actions and relationships to a user.

2.1 Social Media and Digital Evidence

Social media provides an opportunity for users to connect with others of similar interests, form personal or business relationships, and to communicate in real time through sharing multimedia files, instant messaging, microblogging, advertising etc.

It also provides investigators with valuable evidence that can be used to connect a suspect (or suspects) to a criminal activity of interest or to other persons of interest.

Social media has been described by Zeng et al. (2010) as a distributed mode of generating, disseminating content and communication among communities. Its use as a source of digital evidence has been recognised as an area of interest Arshad et al. (2019) deserving of further research in the digital forensics community.

There have been a number of approaches taken by researchers and practitioners exploring social media activity. For example, Shaw et al. (2016) takes a network forensics approach, reviewing the vulnerabilities of social media platforms. Hubert (2014) presents techniques that can be applied to non-criminal investigations such as employee misconduct.

To understand social media communication within the context of this paper, it is necessary to define communication in a 'traditional' sense and how social media fits into the concept of communication. Communication, according to Dance (1967) has been described as **"the elicitation of a response from another**, without making a distinction between the interaction of animate and inanimate matter, or the interaction between humans and animals". Littlejohn (1992) suggests that communication can be defined by "looking at **the conceptual components of the act**, for example, speech, symbols, interaction, and intention".

These definitions outline, in a simplistic way, what communication entails i.e. "eliciting a response from another" through writing, speaking, the use of signs or symbols. It is also worthy to note that Littlejohn includes '**intention**' as one of the conceptual components of the act of communicating.

Viewing these from a digital forensics perspective, communication on social media occurs through interactions/exchanges (**action**) that 'elicit a response' from other users, and these interactions tend to involve the use of textual, graphic, audio content amongst others. These exchanges to a degree, often reflect the 'intent' of the user initiating the interaction and may be used to form associations (**relationships**) on social media platforms.

This could be considered as useful evidence to digital forensics investigators. The content shared, when recovered can be submitted as digital evidence while the nature of the interaction when contextualised may be used to infer the user's intent (**motive**) for example, the user may intend to provoke other users, commit a criminal act, or incite another user into committing a criminal act.

2.1.1 Challenges Posed by Social Media Communication

Although it can be a valuable source of digital evidence, as with every technology, social media also presents challenges to investigators. Some of which include the use of proprietary data formats across supported devices, the type of artefacts written to disk, and the ability of a portion of the platform's user community to engage in undesirable or unlawful activities Bello and DiBlasio (2013); Moore (2014); Select Committee on Communications (2014).

There has been several social media related prosecutions and convictions over the years, highlighting cases where social media has been used for activities such as trolling, murder, drug trafficking, and fraud Haroon and Carter (2010); Bello and DiBlasio (2013).

There is also the lack of standardisation Cusack and Son (2012) which often results in social media investigations being approached as a web browser focused investigation when the social media platform is accessed through a web browser. Self-contained social media applications may handle things differently but were not explored as they were deemed out of scope. It has been suggested that ACPO Guidelines can be used as a starting point when investigating social media activity in the UK.

Other challenges associated with the investigation of social media activity include communication content not being immediately available or accessible on disk David et al. (2020) without additional measures/efforts to retrieve it for example, the use of file carving scripts; there are also possible privacy implications as highlighted by Casey (2013) because some artefacts may contain information about unrelated users or activities.

2.1.2 Requirements of Admissibility of Digital Evidence

Digital evidence has been defined in different ways depending on the context of the investigation (i.e. criminal or non-criminal). Casey (2004) defines digital evidence as "*data stored or transmitted using a computer, which supports or refutes a theory on how an event occurred*"; thus addressing the critical elements of a crime such as determining the intent of a suspect.

Mukasey et al. (2008) defines digital evidence as "*information or data of investigative value that is stored on, received or transmitted by a computer*"; highlighting the role of digital evidence - its "**investigative value**" in criminal (or non-criminal) proceedings.

Digital evidence is known to exhibit a number of characteristics e.g.:

$\Rightarrow$ Latency: the evidence must be processed in order to derive meaningful and usable data from it.

$\Rightarrow$ Fidelity: the original evidence is not necessarily required. A copy of the original would suffice provided that the copying process does not invalidate the evidence.

$\Rightarrow$ Volatility: digital evidence is at risk of intentional or unintentional alteration or deletion. Intentional changes may be an attempt by an individual to corrupt and invalidate the evidence McKemmish (2008) however, this can be addressed by maintaining an audit trail and a comprehensive log of the evidence recovery process.

In order to be considered valuable to an investigation, and to be admissible in a court of law, digital evidence needs to meet certain requirements which include being relevant to the case, reliable, authentic, complete (i.e. paints an understandable, full picture), believable and the recovery method must be repeatable and should produce the same results when conducted by an independent third party ACPO (2012).

These requirements complement the characteristics of digital evidence and guide its admissibility in court. It is important to note that there may be circumstances where changes to digital evidence in the course of an investigation is unavoidable; this can be managed by maintaining a comprehensive record of the interactions between the investigator and the target device and also the possible implications of those interactions.

2.2 Live Triage Investigations

Triage, historically has been in practice in the medical field and has been defined as "*the process of quickly examining patients who are taken to a hospital in order to decide which ones are the most seriously ill and must be treated first*" Cambridge University Press (2019). In digital forensics however, this definition can be used to describe two scenarios: an on-site/live triage or an off-site/post-mortem triage, depending on the type of investigation. Post-mortem triage is out of scope for this paper, for more information, see Garfinkel (2013); Jusas et al. (2017); Parsonage (2009); Roussev and Quates (2012).

Live (on-site) triage is a technique used by investigators to quickly retrieve actionable intelligence at a crime scene. It is implemented in time sensitive investigations such as kidnapping, missing persons; where there is an imminent threat to life; the use of encryption on a device; the need to extract and preserve volatile artefacts Bashir and Khan (2013) or the practicality of seizing a device.

Live triage is appropriate when the requirement for actionable intelligence outweighs the need to perform a lab-based, detailed forensic analysis on all potential sources of digital evidence discovered at a crime scene Wiles and Reyes (2007). The data recovered during a live triage can be used to guide the investigation by highlighting possible areas or persons of interest prior to a full forensic analysis of the digital storage media.

Existing research has proposed a number of techniques in the use of live triage in digital forensics investigations Gielen and Bolzoni (2014); Montasari (2016). These highlight live triage as a viable means of recovering intelligence that can be used to generate new leads and to assess the severity of a crime. The Computer Forensic Field Triage Process Model (CFFTPM) developed by Rogers et al. (2006) can be applied to a wide spectrum of investigations and is considered a foundation for modelling the live triage process. It is focused on on-site analysis of digital devices, to obtain information that can be used during the "search and execution" stage of the investigation or during a suspect's interview. CFFTPM highlights the need to consider live triage as part of the digital forensics process.

The CFFTPM is not automated and is designed to be used by competent users. This requirement for examiner competency presents some challenges as additional resources may be expended on-site where the on-site investigation takes priority over existing cases Hitchcock et al. (2016), resulting in the creation of backlogs in the forensic lab as existing cases may need to be put on hold.

Hitchcock et al. however, in the Digital Field Triage Model (DFT) proposed that providing non-specialists with training on the skills required to supplement/support investigators and analysts on-site would help digital forensics units maximize resources and would also provide the information required during a live triage. This will ensure that analysts required in the lab are able to focus on lab-based work.

Cantrell et al. (2012); Cantrell and Dampier (2012) in the Digital Triage Process Model, also addressed the challenges of appropriately managing resources and the utilization of non-digital forensics specialists during live triage investigations by introducing and implementing semi-automated steps in the proposed process model. Semi-automation, it was argued, would make it easier for non-specialists to use; speed up the live triage process and make it possible for these processes to be scripted into a case-specific framework.

There are concerns around the use of live triage during investigations, specifically highlighting the possibility of data being written to the device being investigated. This is often dependent on the type of investigation. For example, during an incident response investigation, the objective is usually business continuity and ensuring that compromised systems are returned to a 'good working state' thus, creating files or writing to the target device is not often considered a problem Gielen and Bolzoni (2014).

During digital forensics investigations, the state of digital data is to remain unmodified where possible. Thus, in compliance with the relevant jurisdictional guidance and best practice, (e.g., ACPO Principle #2 as UK Guidance), all actions taken during a live triage including any possible impact must be documented and justified.

There are also concerns that a live triage may overlook pertinent evidence Casey (2013) however, this can be addressed by conducting a detailed forensic analysis of the device, building on the evidence recovered during the live triage. The detailed analysis may be conducted on-site or in the lab as appropriate.

In addition to the on-going research into the use of and applicability of live triage processes in digital forensics, further work is required to capture and address how triage tools can be used to:

⇒ supplement the limitations of existing tools and processes.
⇒ manage targeted capture of specific volatile or case-based artefacts that can provide investigators with actionable intelligence.

2.3 Digital Evidence: Challenges

Digital evidence aims to link an offender to a crime or to exonerate a suspect. A number of factors however, pose a challenge to the recovery and use of digital evidence. For example, differences in operating systems, applications and storage methods; the use of concealment and cryptographic techniques DFRWS (2001); the complexity of digital devices 7safe (2014); TWGECSI (2001); the volume of data to be analysed ACPO (2012); data compression and the storage capacity of the digital devices McKemmish (2008); Sommer (1999); spoliation techniques Marcella and Menendez (2007).

Evidence from user Internet activity may change with little or no notice for example, changes to the browser data storage format (e.g. proprietary compression format) and the type of data stored (e.g. cookies, history) could influence the type of data available for recovery. This often means that it takes longer to understand and decode the data in order to extract meaningful information from it.

Marcella and Menendez (2007) suggests reviewing and assessing new technologies from an offender's perspective in order to understand how it can be used to hinder an investigation. This is because technology often ends up being used in a manner that deviates from its original purpose. Encryption and evidence eliminating tools are widely available, often for free on the Internet and can be used to attempt to evade detection or prosecution.

2.4 The Reliability of Digital Evidence

In a generic sense, reliability has been defined as "*the quality of being able to be trusted or believed because of working or behaving well*" Cambridge University Press (2019).

Digital evidence could be user or system generated David et al. (2020) thus when determining its reliability, it is important to assess how trustworthy it is. This is because a number of system activities outside the control of the user may create artefacts that appear related to a case but may not be for example, background processes such as web browser caching.

Casey (2011) suggests that an effective way of assessing the reliability of digital evidence is to focus on the evidence itself (e.g. checking the evidence for signs of tampering/damage), rather than the process through which it was created.

Due to the nature of live triage investigations, where repeated access to a target device is not advisable, reliability can be assessed by:

⇒ logging and detailed documentation of the recovery process.
⇒ how believable the evidence is;
⇒ and how accurately such evidence can be correlated, corroborated and attributed to specific events. For example, artefact '*A*' infers that the user '*Bob*' added user '*Chuck*' as a friend and has visited '*Chuck's*' profile '*X*' times; this inference is supported by JSON and cache artefacts which provide context and imply that '*Bob*' and '*Chuck*' have a social relationship.

2.4.1 Relationship Attribution

Attribution in digital forensics is used to show a link between a suspect to the artefacts found on a digital device. In the context of this paper, relationship attribution is aimed at determining the degree to which a specific artefacts can be attributed to a specific event, using the artefacts recovered during a live triage.

This includes leveraging corroborating artefacts such as browser cached content, system generated artefacts etc. David et al. (2020) to show the connection between a user and social media activity or social relationships and associations. This would facilitate the prompt identification of persons of interest to an investigation prior to a detailed forensic analysis.

3. SMURF

SMURF is a framework built from a collection of components, and designed for live triage investigations focused on user activity on social media in digital forensics.

SMURF is a command-line interface (CLI) tool designed to automate the recovery of artefacts that can be used by investigators to infer and attribute actions to a user.

This section discusses the characteristics and development of SMURF. It also discusses how SMURF can be used by investigators to effectively and quickly build and deploy triage tools during an investigation.

3.1 Framework Problem Definition

David et al. (2020) presented results from a manual analysis of user activity on Twitter. This was done as a "dead disk" forensic analysis and provided some examples of the type of artefacts that an investigator may expect to find on digital storage media where a user has engaged in activity on Twitter.

In some time sensitive cases, digital forensic investigators are required to perform a live triage in order to obtain actionable intelligence that will help save the life of an individual (e.g. missing person). In such instances, this immediate need for actionable intelligence takes precedence over a full disk analysis which can be conducted later.

SMURF intends to address the need for a generalisable, reusable, and adaptable technique for automated evidence recovery from a variety of usage scenarios, providing a reusable design with multiple levels of functionality. To do this, SMURF is designed for:

I. use in a live triage investigation

II. the furtherance of the automated recovery, categorising recovered artefacts based on possible usage scenarios and the enabling the attribution of actions or relationships to a user.

III. the provision of quick and usable information that would enable investigators define the initial stages of an investigation, generate new leads, and highlight possible areas of interest to be looked at during a detailed forensic analysis.

3.2 Framework Characteristics

The development process for SMURF involved considering the characteristics that would define its functionality. This includes determining the availability and location of artefacts of interest, determining the types of artefacts that can be used to provide context to user activity.

This process also includes understanding that certain types of artefacts such as the contents of communication (e.g. text, multimedia files) may not be readily available to view or accessible on disk as they may be in unallocated space. Thus, it may be necessary to recover and use other artefacts that could be used in making inferences about user activity.

It is also important to understand how these artefacts can be used to attribute actions and relationships to a user based on the amount and usefulness of the actionable intelligence SMURF provides an investigator. This will enable an investigator to understand the context of the artefact, how it came to be, and what possible usage scenario could have created it. It will also assist the investigator when making a judgement on how to proceed.

3.3 Framework Development

The limitations observed with the ability to recover artefacts from social media activity (Section 3.2, i.e. communications content in unallocated space) highlights the need to identify and define the artefacts that may be of interest during a triage investigation.

Examples of these artefacts include browser history data such as presented in David et al. (2020) i.e. URLs, authentication credentials, session information, cached data etc. as well as system artefacts that provide context about the user accounts, programs and applications running on their digital storage media.

3.3.1 Components

This stage of the framework development process involved identifying and defining the major parts that constitute the framework. It involved evaluating the artefacts recovered during the manual investigation of the disk image and determining whether the artefacts would provide quick, easy to assimilate information for an investigator during a live triage.

The manual artefacts were subsequently categorised into context and case specific artefacts; then they were broken down into specific artefacts from which the framework components were derived.

Examples include:

I. a component that recovers artefacts containing user account information, such as the number of user accounts on the computer, the user Security Identifier (SID), and logon/logoff information
II. component for the recovery of user data from social media activity

As SMURF is intended for live triage investigations, it was determined that a component would be required for logging the interaction between the deployment device and the target device in compliance with ACPO #2.

3.3.2 Development Environment

This stage of the framework involved identifying a suitable programming language for the proof of concept (PoC). It also involved reviewing existing work with open source live triage tools to enable a gap analysis as part of this research. The gap analysis identified the capabilities of existing tools and how SMURF augments them.

The development environment setup is as follows:

⇒ Windows 10 Education (Local Machine)
⇒ Python 3
⇒ PyCharm Professional Edition with Anaconda Plugin

Scripting Individual Components Each component of the framework (as described in Section 3.3.1) was scripted to recover specific artefacts. In addition to the artefact-specific components, a 'core' component was also scripted to collate the recovered artefacts, and present them in a HTML report for the investigator. This approach can be described as a "*practitioner friendly*" approach to automated evidence recovery in a live triage.

This process is described further in Section 4.2 where a graphical illustration of the components is presented.

4. BUILDING THE SMURF FRAMEWORK

This section discusses at a high level, the theoretical aspects of SMURF. The intention was to build a linear, modular investigative tool, with the potential to extend and mature its functionality. SMURF's modularity allows for the addition or exclusion of components based on the use case.

Figure 1 is a graphical representation of how SMURF works. The components used to achieve these are discussed in Section 5.

4.1 Capture

This stage involves recovering data from a live machine. As this is the stage that initiates interaction between the triage device and the target device, it was necessary to first programmatically record the triage device details, the date and time it was connected to the target device before proceeding with the data collection.

The data collection at this stage focused on system generated and user generated artefacts (e.g. web browser artefacts).

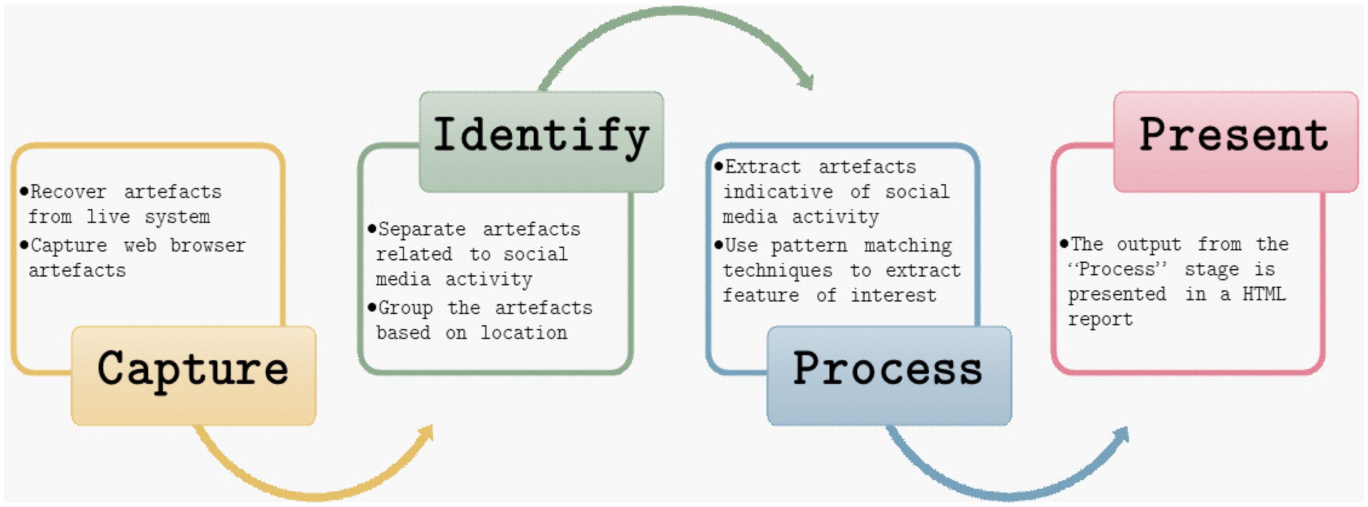

Capture
•Recover artefacts from live system
•Capture web browser artefacts
Identify
•Separate artefacts related to social media activity
•Group the artefacts based on location
Process
•Extract artefacts indicative of social media activity
•Use pattern matching techniques to extract feature of interest
Present
•The output from the "Process" stage is presented in a HTML report

Figure 1: An illustration of how SMURF works

4.2 Identify

This stage involves segregating the data collected during the capture. The artefacts are grouped based on type and the location found. For example, if the data contains the social media platform name or an associated name, and it is a URL (**type**), put it in the "visited places group" for further processing.

4.3 Process

The artefacts recovered and grouped at the "Identify" stage are processed to highlight the type of activity that could have created them. For example, user account registration, viewing and sharing content etc.

This stage also involves the use of pattern matching techniques as presented in David et al. (2020) to extract features of interest.

4.4 Present

The data from preceding stages feed into the "Present" stage. Here, the results from the processing stage are parsed and returned in a text based HTML report. Following the theme of a live triage, this report provides the investigator with a quick view of the data recovered, in a usable manner.

5. CREATING A POC SMURF IMPLEMENTATION

This section presents the data generation and analysis methodology for a PoC.

5.1 Data Generation

The data set used in this paper was generated from previous experiments conducted as part of the work presented in David et al. (2020). It involved the simulation of a range of normal user activity on Twitter, some of which include a variety of the activities listed below:

⇒ Power on and log in to the experimental VM
⇒ Launch the web browser (Google Chrome and Firefox were used)
⇒ Login to test user account on Twitter via the browser
⇒ Search for users to follow (publicly accessible accounts e.g. @bbcgoodfood)
⇒ Sending tweets (text, images, links)
⇒ Reply and retweet messages
⇒ Viewing and sending Direct Messages (DMs)
⇒ Sending photos via DM
⇒ Retweets via DM
⇒ Send follow requests to other test user account
⇒ Viewing and accepting follow requests
⇒ Updating the account privacy settings

⇒ Scrolling through the test user account timelines and that of its followers

5.2 Data Analysis

The data analysis for this paper was conducted in a Windows 10 desktop environment using the PoC framework - SMURF. This section discusses how individual components of the SMURF were used for data analysis; it includes code extracts with several lines removed for brevity.

A graphical illustration of SMURF is shown in Figures 2 and 3. It shows the components that make up the framework and how they feed data into the core component enabling the generation of a report with the relevant information required by an investigator.

5.2.1 Component: device_identifier

This component was used to retrieve the identifying information for the triage device on which SMURF was running. This includes the Vendor ID (VID) and Product ID (PID) of the device.

This information is written to the report to account for the interaction between the triage device and the target device being investigated.

Extracts from device_id.py

```
# Path to log file = "C:\Windows\INF"
windows_dir = os.path.abspath(os.sep)
path_to_logfile = pathlib.Path(windows_dir).joinpath('Windows', 'INF')

# Read data from setupapi.dev.log to get
# 'Device Install' Events for USB Devices
def get_devices():
       # Check that the setupapi.dev.log file exists on this machine
       if pathlib.Path(path_to_logfile).joinpath('setupapi.dev.log').is_file():

       # Read lines in setupapi file and search for device identifiers
       try:
           device_list = []

           with open(os.path.join(path_to_logfile,'setupapi.dev.log'),'r') as
                                              log_file:
            # Get the OS info from the first 6 lines of the log file (currently in
                                                 Win 10).
            # This may change in the future if it does, amend the number of lines
                                                 to read in.
                  for i in range(6):
                       os_info = str(log_file.__next__().strip(""))
                       device_list.append(os_info)
```

5.2.2 Component: predefined_paths

This component was written to support artefact recovery from Windows Registry. It contains known paths of interest in the SAM, SECURITY, SOFTWARE, SYSTEM, NTUSER.DAT and UsrClass.dat hives and was used as a global settings file for the registry parsers.

The paths held in this component can be used to provide context to user activity in a live triage. For example, identifying applications (e.g. browsers) installed by the user, mounted devices (e.g. shared network drives used for sharing files on social media). See Appendix A.1 for extracts.

5.2.3 Component: app_config

This component is made up of individual parsers for the registry hives. It is important to note that the SAM and SECURITY hives are locked and inaccessible when the device is powered on. This is because Windows locks the file and prevents read/write activity from any accounts other than the SYSTEM. To address this limitation, the parsers used the Python subprocess module to call the *reg.exe save hklm*\x command (e.g.

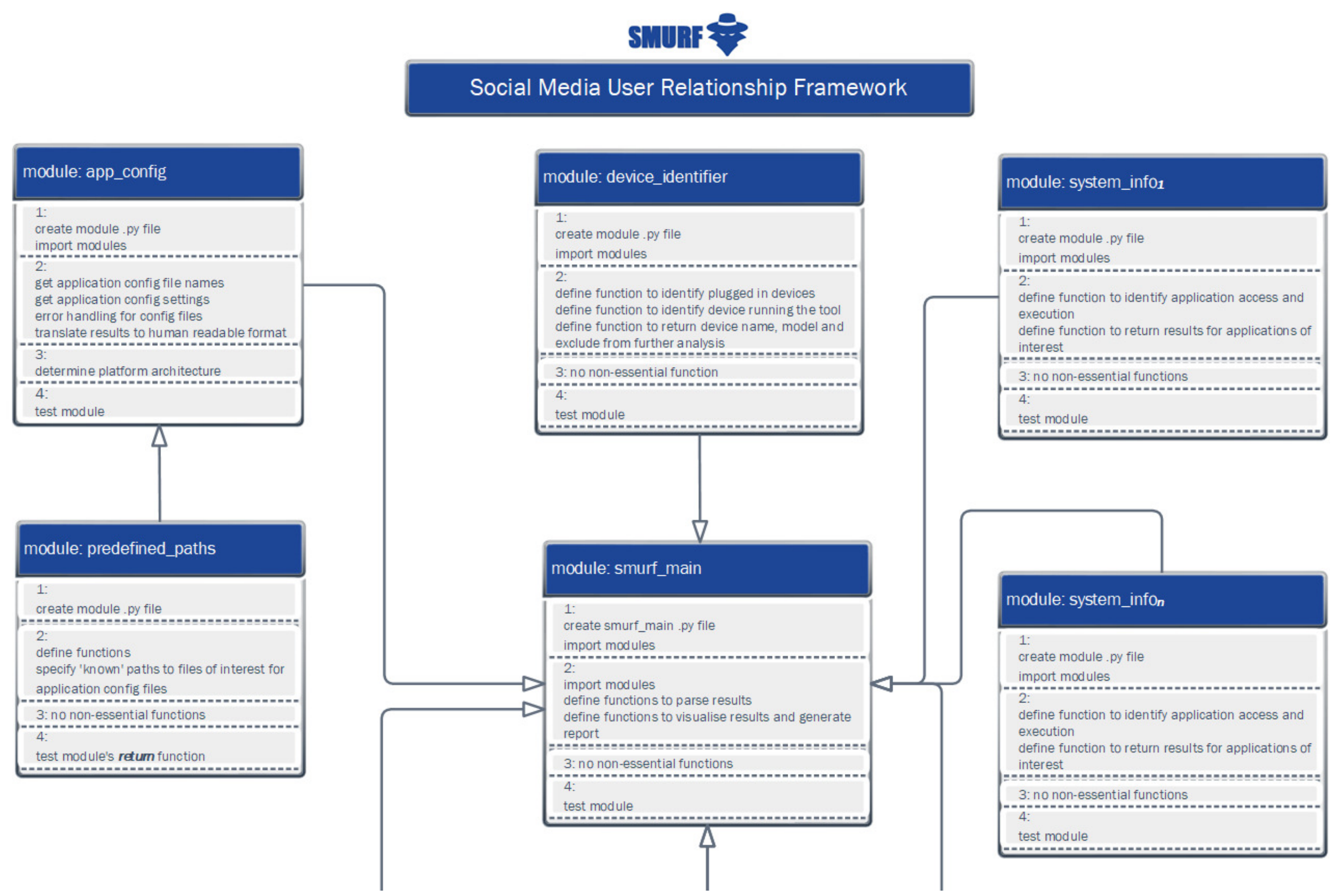

SMURF
Social Media User Relationship Framework
module: app_config
1: create module .py file import modules
2: get application config file names get application config settings error handling for config files translate results to human readable format
3: determine platform architecture
4: test module
module: device_identifier
1: create module .py file import modules
2: define function to identify plugged in devices define function to identify device running the tool define function to return device name, model and exclude from further analysis
3: no non-essential function
4: test module
module: system_info₁
1: create module .py file import modules
2: define function to identify application access and execution define function to return results for applications of interest
3: no non-essential functions
4: test module
module: predefined_paths
1: create module .py file
2: define functions specify 'known' paths to files of interest for application config files
3: no non-essential functions
4: test module's return function
module: smurf_main
1: create smurf_main .py file import modules
2: import modules define functions to parse results define functions to visualise results and generate report
3: no non-essential functions
4: test module
module: system_infoₙ
1: create module .py file import modules
2: define function to identify application access and execution define function to return results for applications of interest
3: no non-essential functions
4: test module

Figure 2: Part 1
A "practitioner friendly" illustration of SMURF components and how data is fed into the main component

reg.exe save hklm\sam). This was used to dump the SAM and SECURITY hives. The HARDWARE hive was also dumped to grab the current state of the system hardware information as this hive is volatile and is lost when the device is powered off.

The winreg module was used to access the Windows registry API in read-only mode to minimise any possible impact on the target device. See Appendix A.2 for extracts.

5.2.4 Component: system_info

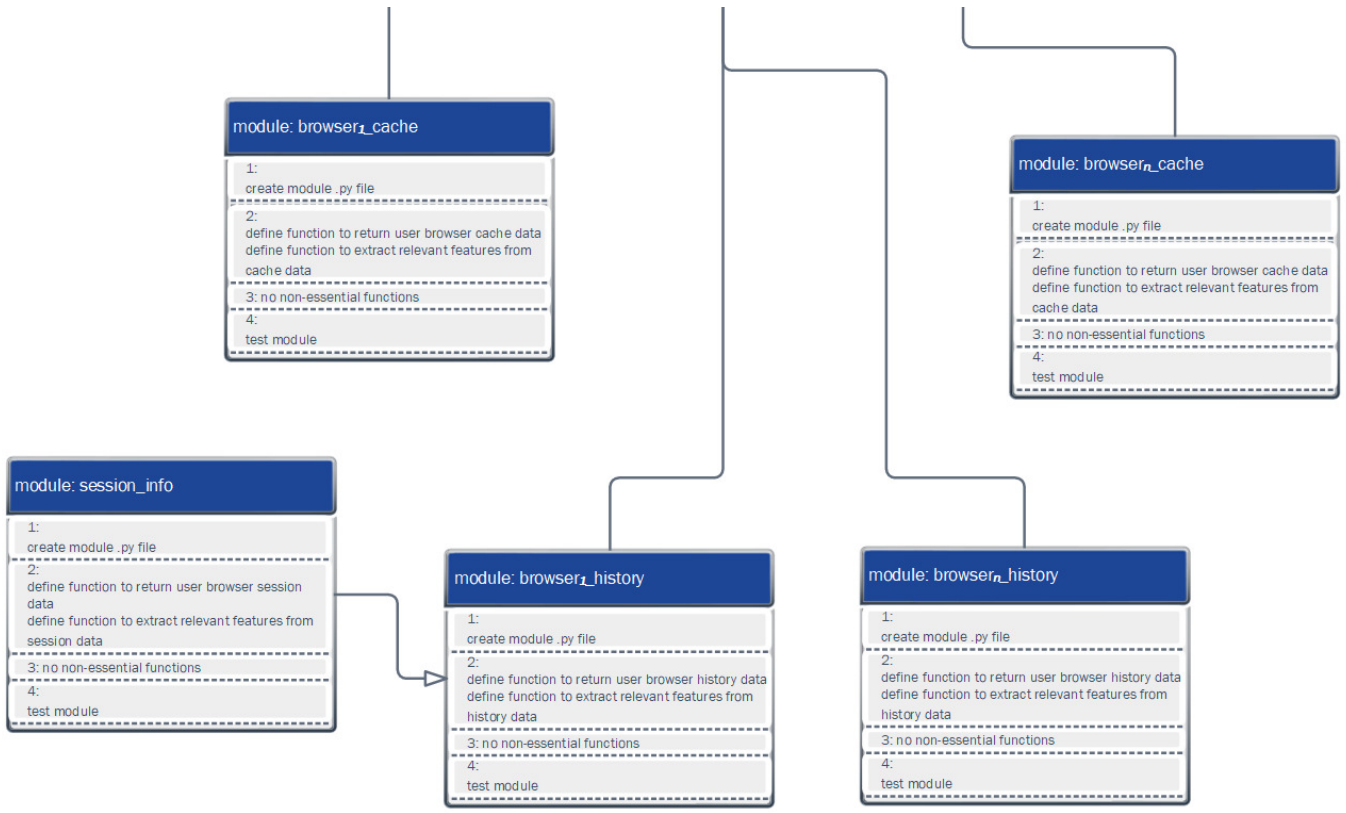
This component was used for the recovery of system generated artefacts from Prefetch and Event Logs. This was used to provide corroborating information on application installation and access times; and also to provide context to user activity.

This component was used to demonstrate the ability to use third party command-line interface (CLI) tools within SMURF. For example:

⇒ PECmd Zimmerman (2019) is a CLI tool that was used to process the Prefetch files and return the contents in JSON format.

⇒ python-evt Ballenthin (2019), a Python parser for Windows event logs (post-Windows XP), was used to process the Application and Security event logs. The output was saved as an XML file and a function was written to parse the XML file to display the content in an "easy to digest" format in the report.

⇒ SysInternals tools Russinovich (2016, 2018) were used to recover additional/corroborating system related and generated artefacts


module: browser$_1$_cache
1: create module .py file
2: define function to return user browser cache data define function to extract relevant features from cache data
3: no non-essential functions
4: test module
module: browser$_n$_cache
1: create module .py file
2: define function to return user browser cache data define function to extract relevant features from cache data
3: no non-essential functions
4: test module
module: session_info
1: create module .py file
2: define function to return user browser session data define function to extract relevant features from session data
3: no non-essential functions
4: test module
module: browser$_1$_history
1: create module .py file
2: define function to return user browser history data define function to extract relevant features from history data
3: no non-essential functions
4: test module
module: browser$_n$_history
1: create module .py file
2: define function to return user browser history data define function to extract relevant features from history data
3: no non-essential functions
4: test module

Key	Description
1	start_module
2	core_function
3	extra_function
4	testing

Figure 3: Part 2
A "practitioner friendly" illustration of SMURF components and how data is fed into the main component

such as processes, registered owner, system uptime etc.

The subprocess module was used to call these third party tools from the *system_info* component. See Appendix A.3 for extracts.

5.2.5 Component: session_info

This component was used to recover artefacts from the browser session files. For this paper, two types of session related artefacts were recovered. Extracts can be found in Appendix A.4.

I. sessionstore.mozlz4: this is used by Firefox to store session data. Artefacts from this file were recovered using a Python script derived from Blumenbach (2015)'s mozlz4a script.

II. Current/Last Session; Current/Last Tabs: these are used by Google Chrome, and as implied by the file names, the window and tab activities of the user are stored here. When Chrome is running, these files are locked by the host and are inaccessible, so it was necessary to use the psutil module to programmatically terminate chrome.exe.

The output was saved in text files for subsequent processing using RegEx.

5.2.6 Component: browser_history

This component was used to recover artefacts related to user browser activity. It is also designed to be extensible, allowing for the recovery of artefacts from multiple browsers and for session information to be fed into the framework to support browser history artefacts. It calls third party tools such as *chromagnon* Bancel, Jean-Rémy (2015) which was used to parse the Chrome History file, and to facilitate the recovery of session artefacts.

For the purposes of this paper, artefacts from Firefox and Chrome were analysed. See Appendix A.5 for extracts.

5.2.7 Component: browser_cache

This component was used to recover data in the browser cache that can be used for the corroboration of other browser artefacts.

The manual analysis conducted by David et al. (2020), observed and highlighted that content may be cached independent of user interaction. The cache returns a sizeable volume of data thus, for a live triage use case, this component focuses on the recovery of URL related cached content to reduce the amount of time and effort required to process cache information.

This component also demonstrates the extensibility and flexibility of SMURF. It calls third party tools such as *Hindsight* by Obsidian Forensics Benson (2019), which was used to facilitate the parsing of the browser cache artefacts. See Appendix A.6 for extracts.

5.2.8 Component: smurf_main

This is the main component of the framework. It was used to collate, manage the data ingested from the artefacts from the other components.

It was designed to format the data recovered and to generate a HTML report that was used to display the results in an 'easy to assimilate' format for the investigators (see Appendix A.7 for the HTML template extracts).

Extracts from smurf_main.py

```
# SMURF Main

def smurf():
# read content of output files from framework components
# write data of interest to SMURF Report.html

# Content to be published
   title = "Triage Report"
   devices = "SMURF deployed from: "

[...]
# Rename existing template and load the new report template
sys.stdout.write("Creating report template... \n")
suffix = "smurf_report/" + str(dt.datetime.now().strftime("%Y%m%d-%H%M%S")) + "-
                                          smurf_report.html"
html_template = "smurf_report/report_template.html"
if os.path.exists(html_template):
   os.rename("smurf_report/smurf_report.html", suffix)
   shutil.copy2(html_template, "smurf_report/smurf_report.html")

[...]
with open(url_feats, 'r') as uf:
    twitter_url = []
    for line in uf:
         twitter_feats = uf.readlines()
        for item in twitter_feats:
         if "twitter" in item:
      url_data = twitter_url.append(item)
```

```
        twitter_url_data = twitter_url[-50:] # negative indexing, enumerate from
                                                             tail
                else:
           if len(twitter_feats) == 0:
            url_data = twitter_url.append("No Twitter URLs found...")
             twitter_url_data = twitter_url[:1]
[...]
```

5.3 Method of Deployment

Being a collection of components, some of which call third party tools/utilities, a number of options were considered as possible methods of deployment. This was done to ensure:

⇒ good ethical conduct and compliance with the legal use of the third party tools (i.e. adherence to potential licence ramifications or legal implications);

⇒ respect for the authors and their licences by using the tool in the way author consent implies.

Some of the utilities explored as a method of deployment can be used to bundle a python script and its dependencies into an executable file; however these were deemed impractical for the purposes of demonstrating the extensibility and adaptability of SMURF and ensuring that third party tools are used as intended as mentioned above.

WinPython WinPython (2019) a free, portable scientific distribution of Python for Windows was deployed to a USB device and used to test the PoC in a live Windows 10 VM.

6. CASE STUDY: SMURF ON TWITTER

This section discusses the framework testing process, using Twitter as a case study. The results discussed in this section are grouped based on potential user activity. The weighting scale used to determine the feasibility of SMURF is also discussed in this section.

6.1 Twitter as a Case Study

Twitter is a very popular, real-time information network with a global user base. It has been described as *"a service for friends, family and coworkers to communicate and stay connected through the exchange of quick, frequent messages. People post Tweets, which may contain photos, videos, links and text. These messages are posted to your profile, sent to your followers, and are searchable on Twitter search*" Twitter Help Center (2019).

Twitter in its FAQs for new users, provides a description of how the platform can be used. This information was used as a guidance to extrapolate the basic activities that can be carried out on Twitter. These activities were placed into categories and the weighting scale shown in Figure 4 was derived from the analysis conducted in David et al. (2020).

In the context of a live triage investigation, the following is proposed:

⇒ **Strongly expect to find**: these are a range of artefacts that should generally be available to the investigator e.g. username, email address.

⇒ **May expect to find**: these are a range of artefacts that may often be available but not always e.g. login/logout activity, profile IDs of other users.

⇒ **Corroborates other artefacts**: these are artefacts that validates previously recovered artefacts e.g. search parameters in a URL.

⇒ **Needs to be contextualized**: this refers to any artefact that could be interpreted to mean different things and thus needs to be put in context of the previously recovered artefacts to verify its relevance to the case e.g. the user's email address, email addresses of other users.

⇒ **Requires corroboration**: this refers to any artefact that could mean something else and cannot be validated on its own. It needs to be supported by existing artefacts (e.g., communication content, text extracts, images).

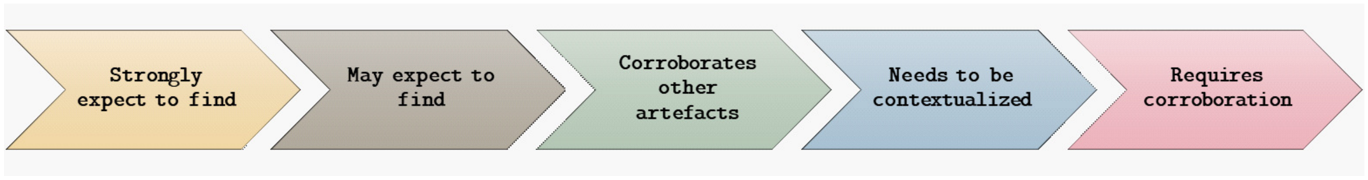


Figure 4: An illustration of the weighting scale

It is important to note that there may be overlaps when weighting the availability and relevance of artefacts recovered during a live triage investigation. For example, an investigator **may expect** to find the username of another user but would need to **corroborate** that information and put it in **context** to determine if it occurred as a result of direct user interaction or background processes on the social media platform.

6.1.1 Registration/Authentication Activities

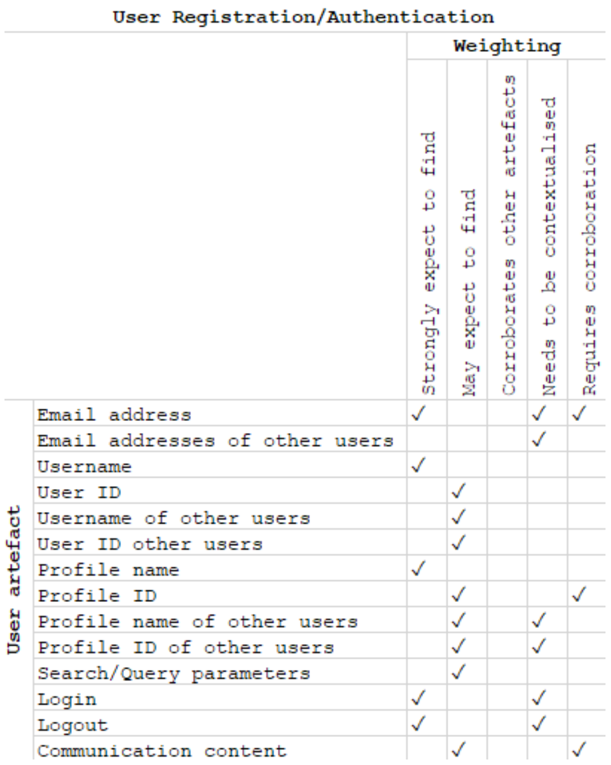
Users can access Twitter in a 'read-only' mode if they are not registered or logged in i.e. they are able to view content but cannot respond or interact with other users. In order to get the full benefits of the platform, a registered account is required.

Creating an account allows a user to personalise the account by amending the account settings to allow or restrict the public from viewing tweets; and adding profile photos and banners.

Figure 5 shows the range of artefacts associated with user registration/authentication on Twitter as recovered by SMURF as part of the first stage of the framework testing. The results indicate that there is a strong possibility of recovering artefacts such as the email address that was used for account registration.

Subsequent to that, URLs indicating login/logout activity can also be recovered and used to infer user account authentication activity.

There are other artefacts that may be used to support account activity. These include finding the usernames, profile names, user/profile IDs for other users. This may occur during the registration process where Twitter suggests people the user may like to follow.



User Registration/Authentication

		Weighting				
		Strongly expect to find	May expect to find	Corroborates other artefacts	Needs to be contextualised	Requires corroboration
User artefact	Email address	✓			✓	✓
	Email addresses of other users				✓	
	Username	✓				
	User ID		✓			
	Username of other users		✓			
	User ID other users		✓			
	Profile name	✓				
	Profile ID		✓			✓
	Profile name of other users		✓		✓	
	Profile ID of other users		✓		✓	
	Search/Query parameters		✓			
	Login	✓			✓	
	Logout	✓			✓	
	Communication content		✓			✓

Figure 5: User registration/authentication artefact weighting

In the context of user activity, to infer/confirm possible account ownership, '**strongly expect to find**' provides stronger evidence to validate the assumption. Table 1 shows the possible locations of these types of artefacts from Firefox and Google Chrome.

Table 1: Possible locations of registration/authentication artefacts

Chrome	Firefox
Last/Current Tabs	formhistory.sqlite
Last/Current Session	places.sqlite
Login Data	sessionstore.mozlz4
Web Data	login.json

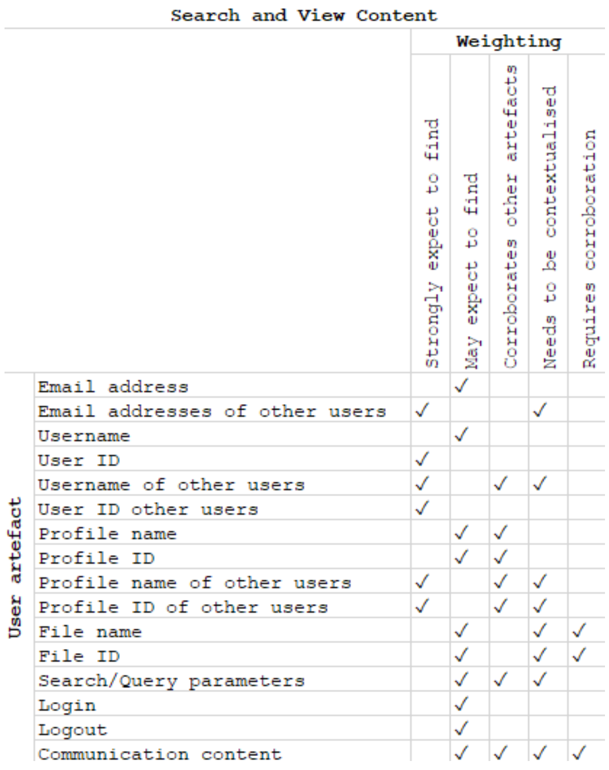
6.1.2 Search and View Contents

As mentioned in Section 6.1.1, it is possible for users who are not logged in or registered to search for and view content in Twitter. This may be through typing in a Twitter URL with the *@handle* of a Twitter user e.g. `twitter.com/joebloggs` or through a search engine e.g. "joe bloggs twitter".

The second stage of the framework testing involved the identification and recovery of artefacts that can be used to infer 'search and view' activities. This applies to both registered and unregistered users; the main difference being the availability of artefacts indicating that the user has a Twitter account for example, the presence of the artefacts described in Section 6.1.1.

Using the weighting matrix, the PoC was used to recover artefacts such as user/profile names and IDs in the '**strongly expect to find**' category. Other artefacts that can be used to build context were recovered in the '**may expect to find**' category. For example, where a Twitter user profile has been searched for by email address; the search parameters (**may expect to find**) can be used to contextualise the profile ID (**strongly expect to find**) and thus the profile visit/view.

Regular Expressions (RegEx) were used at this stage to parse the output files from the framework components to extract artefacts inferring a range of search and view activities that occurred on Twitter during the experiments. Table 2 shows the possible locations of these types of artefacts from Firefox and Google Chrome.



Search and View Content

		Weighting				
		Strongly expect to find	May expect to find	Corroborates other artefacts	Needs to be contextualised	Requires corroboration
User artefact	Email address		✓			
	Email addresses of other users	✓			✓	
	Username		✓			
	User ID	✓				
	Username of other users	✓		✓	✓	
	User ID other users	✓				
	Profile name		✓	✓		
	Profile ID		✓	✓		
	Profile name of other users	✓		✓	✓	
	Profile ID of other users	✓		✓	✓	
	File name		✓		✓	✓
	File ID		✓		✓	✓
	Search/Query parameters		✓	✓	✓	
	Login		✓			
	Logout		✓			
	Communication content		✓	✓	✓	✓

Figure 6: User search and view artefact weighting

Table 2: Possible locations of artefacts inferring search and view activity

Chrome	Firefox
Last/Current Tabs	places.sqlite
Last/Current Session	sessionstore.mozlz4
Cache	Cache
Cookies	Cookies

6.1.3 Share Content

Twitter is used for content dissemination through "Tweeting" or "Retweeting" content - images, videos, text, and/or audio. As part of content sharing, users may also choose to communicate privately by exchanging direct messages (DMs).

Content exchanged on Twitter (tweets, retweets, DMs) may be accompanied by hashtags to highlight trending i.e. currently popular topics; the keyword(s) referencing the topic is prefixed with the hash symbol (#) for example, #fitness, #freebies, #LatestNews.

Social media communication in the context of this paper has been described in Section 2.1 as the messages exchanged between two (or more) users, which may contain text, audio, video or image files.

As discussed in Section 2.1.1 some of the aforementioned may not be readily available or accessible on disk during a live triage investigation thus, it may be necessary to use other artefacts to build a picture of the user's activity.

SMURF was used to search for evidence of these activities to determine the reliability of the weighting matrix applied to the manual analysis of the artefacts presented in David et al. (2020).

Figure 7 shows the range of user artefacts that may be used to infer content sharing on Twitter. For example, user/profile names and IDs, in addition to URLs parameters such as file names can be used in conjunction with page titles and login/logout URLs to infer a user was logged and possibly shared the file.

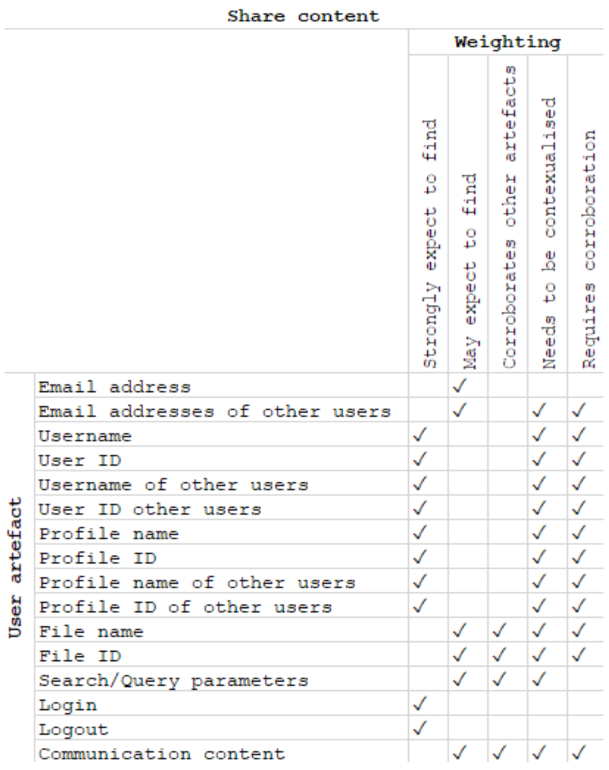
The framework did not focus on or attempt the recovery of the exact contents of communication as the goal of a live triage is the timely and efficient provision of reliable intelligence, for the intended audience (e.g. investigators), in a usable and easy to understand format. However, the recovered artefacts will need to be corroborated.

It is important to note that non-registered users cannot share but are able to view publicly available content (Section 6.1.2) so **context** is necessary in order to draw inferences.

Table 3 shows the possible locations of these types of artefacts from Firefox and Google Chrome.

Table 3: Possible locations of artefacts inferring content sharing activity

Chrome	Firefox
Last/Current Tabs	places.sqlite
Last/Current Session	sessionstore.mozlz4
Cache	Cache



Share content						
		Weighting				
		Strongly expect to find	May expect to find	Corroborates other artefacts	Needs to be contexualised	Requires corroboration
User artefact	Email address		✓			
	Email addresses of other users		✓		✓	✓
	Username	✓			✓	✓
	User ID	✓			✓	✓
	Username of other users	✓			✓	✓
	User ID other users	✓			✓	✓
	Profile name	✓			✓	✓
	Profile ID	✓			✓	✓
	Profile name of other users	✓			✓	✓
	Profile ID of other users	✓			✓	✓
	File name		✓	✓	✓	✓
	File ID		✓	✓	✓	✓
	Search/Query parameters		✓	✓	✓	
	Login	✓				
	Logout	✓				
	Communication content		✓	✓	✓	✓

Figure 7: User shared content artefact weighting

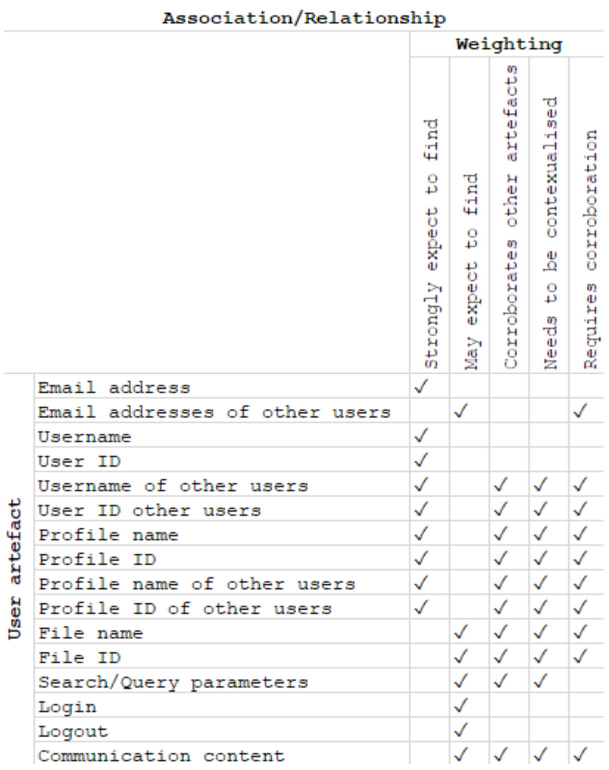
6.1.4 Associations/Relationships

On Twitter, users can choose to subscribe and be notified of another user's tweets. This is described as "following" Twitter Help Center (2019) and is designed as a two-way process where users follow each other. However, there are instances where one user does not reciprocate for example, a corporate entity would have several, perhaps thousands of followers but will not follow all its followers.

"Followership" may be used to determine a user's interests, infer associations and relationships.

Figure 8 shows a range of artefacts recovered by SMURF and their weighting. These artefacts can be used to draw inferences about a user's associations and by extension relationships on Twitter.

In order to determine the relationship(s) between users on Twitter, it is important to put the evidence inferring contact in context. This is because it is possible to form associations without a direct relationship. Examples include but are not limited to where a user has connections with or follows Government or Commercial entities with a Twitter profile; following celebrity



Association/Relationship

		Weighting				
		Strongly expect to find	May expect to find	Corroborates other artefacts	Needs to be contexualised	Requires corroboration
User artefact	Email address	✓				
	Email addresses of other users		✓			✓
	Username	✓				
	User ID	✓				
	Username of other users	✓		✓	✓	✓
	User ID other users	✓		✓	✓	✓
	Profile name	✓		✓	✓	✓
	Profile ID	✓		✓	✓	✓
	Profile name of other users	✓		✓	✓	✓
	Profile ID of other users	✓		✓	✓	✓
	File name		✓	✓	✓	✓
	File ID		✓	✓	✓	✓
	Search/Query parameters		✓	✓	✓	
	Login		✓			
	Logout		✓			
	Communication content		✓	✓	✓	✓

Figure 8: User association/relationship attribution

fan pages or profiles or following other users with similar interests (photography, hiking etc.).

It may also be possible to use Twitter search and query parameters to aid the attribution of associations and relationships to a user. This may be useful in highlighting where particular user profiles were searched for and then added as a social connection.

Table 4 shows the possible locations of these types of artefacts from Firefox and Google Chrome.

Table 4: Possible locations of artefacts inferring associations/relationships

Chrome	**Firefox**
Last/Current Tabs	formhistory.sqlite
Last/Current Session	places.sqlite
Login Data	sessionstore.mozlz4
Cache	Cache

6.2 SMURF: Assessing Artefact Reliability with AUTOPSY®

As part of assessing the reliability of the artefacts recovered by the framework, and to highlight its contributions to the digital forensics community, SMURF was tested against Autopsy® 4.15.0 Carrier (2020); an open source, peer reviewed, in-use digital forensics tool.

Autopsy® is a digital forensics platform and graphical interface to The Sleuth Kit® and other digital forensics tools Carrier (2020). It has a broad range of features including "Communications Visualization" which provides a consolidated view of events of interest and allows the investigator to visualize communication based on commonly used accounts and the timeframe of an event of interest.

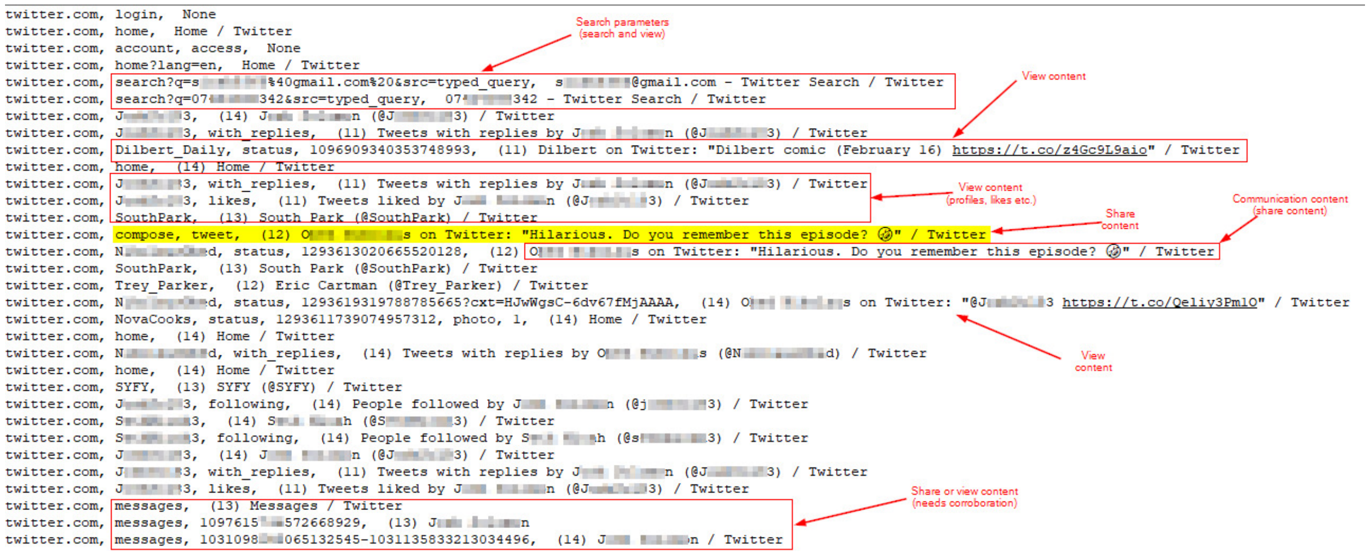
The process consists of the following activities:

⇒ Download Autopsy 4.15.0 from GitHub
⇒ Install Autopsy and create a new case
⇒ Add the experimental VM as a "Data Source"
⇒ Select and run relevant ingest modules ("Recent Activity", "Keyword Search")
⇒ Review results

A comparison of the artefacts inferring user activities as recovered by SMURF and the web browser activities recovered by Autopsy showed that SMURF was able to capture these activities as shown in Figures 9 and 10.

An attempt was made to visualise social media activities through the "Communications Visualization" feature, based on the Twitter URLs recovered by Autopsy however, it appears that web history visualisation is currently not supported although it works well with accounts found in the Data Source e.g. email accounts (PST, EML, MBOX), call logs and accounts from mobile devices. These were out of scope and not explored further.

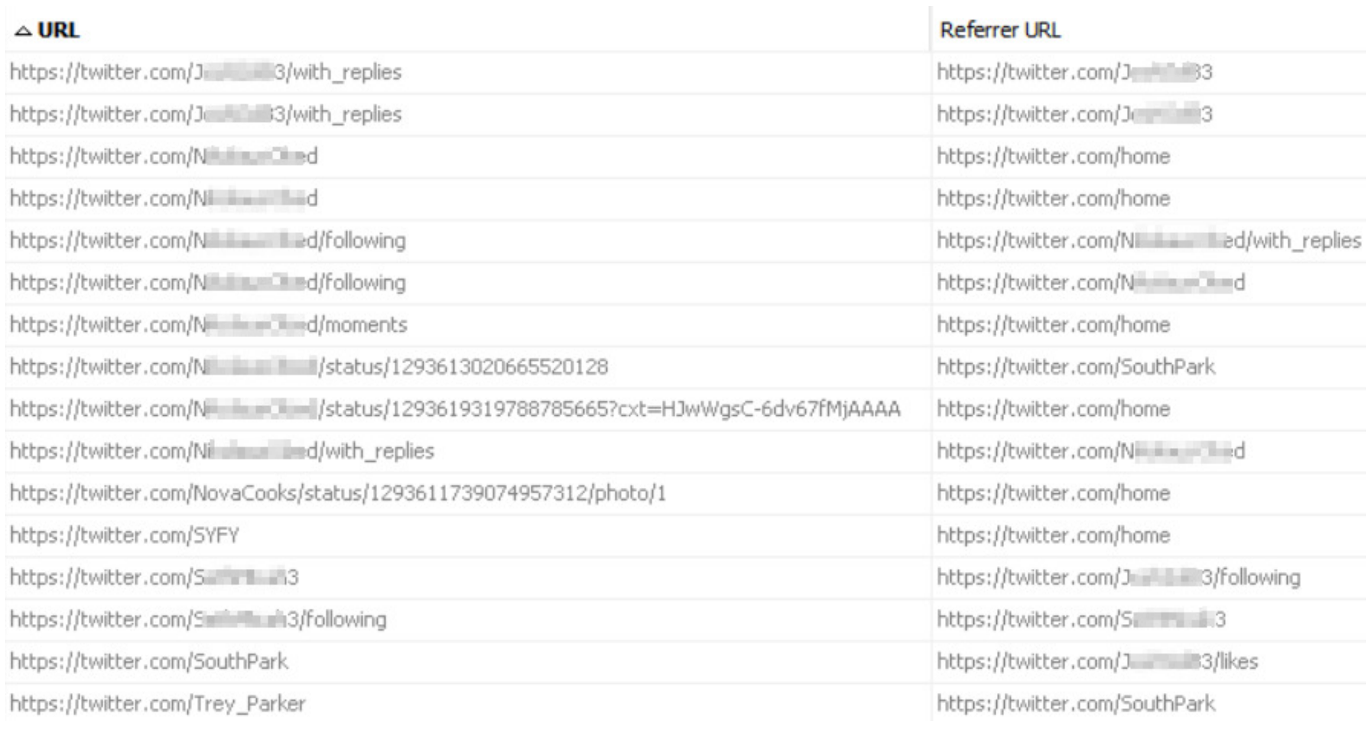
In May 2020, Brian Carrier through Basis Technology Basis Technology (2020) offered free Autopsy training. The feedback from the digital forensics community evidenced the value of the tool. It was suggested that users can leverage the extensibility of Autopsy by adding plugins to the tool.



```
twitter.com, login,  None
twitter.com, home,  Home / Twitter
twitter.com, account, access,  None
twitter.com, home?lang=en,  Home / Twitter
twitter.com, search?q=s[redacted]%40gmail.com%20&src=typed_query,  s[redacted]@gmail.com - Twitter Search / Twitter
twitter.com, search?q=07[redacted]342&src=typed_query,  07[redacted]342 - Twitter Search / Twitter
twitter.com, J[redacted]3,  (14) J[redacted]n (@J[redacted]3) / Twitter
twitter.com, J[redacted]3, with_replies,  (11) Tweets with replies by J[redacted]n (@J[redacted]3) / Twitter
twitter.com, Dilbert_Daily, status, 1096909340353748993,  (11) Dilbert on Twitter: "Dilbert comic (February 16) https://t.co/z4Gc9L9aio" / Twitter
twitter.com, home,  (14) Home / Twitter
twitter.com, J[redacted]3, with_replies,  (11) Tweets with replies by J[redacted]n (@J[redacted]3) / Twitter
twitter.com, J[redacted]3, likes,  (11) Tweets liked by J[redacted]n (@J[redacted]3) / Twitter
twitter.com, SouthPark,  (13) South Park (@SouthPark) / Twitter
twitter.com, compose, tweet,  (12) O[redacted]s on Twitter: "Hilarious. Do you remember this episode? 😂" / Twitter
twitter.com, N[redacted]d, status, 1293613020665520128,  (12) O[redacted]s on Twitter: "Hilarious. Do you remember this episode? 😂" / Twitter
twitter.com, SouthPark,  (13) South Park (@SouthPark) / Twitter
twitter.com, Trey_Parker,  (12) Eric Cartman (@Trey_Parker) / Twitter
twitter.com, N[redacted]d, status, 1293619319788785665?cxt=HJwWgsC-6dv67fMjAAAA,  (14) O[redacted]s on Twitter: "@J[redacted]3 https://t.co/Qeliv3Pml0" / Twitter
twitter.com, NovaCooks, status, 1293611739074957312, photo, 1,  (14) Home / Twitter
twitter.com, home,  (14) Home / Twitter
twitter.com, N[redacted]d, with_replies,  (14) Tweets with replies by O[redacted]s (@N[redacted]d) / Twitter
twitter.com, home,  (14) Home / Twitter
twitter.com, SYFY,  (13) SYFY (@SYFY) / Twitter
twitter.com, J[redacted]3, following,  (14) People followed by J[redacted]n (@j[redacted]3) / Twitter
twitter.com, S[redacted]3,  (14) S[redacted]h (@S[redacted]3) / Twitter
twitter.com, S[redacted]3, following,  (14) People followed by S[redacted]h (@s[redacted]3) / Twitter
twitter.com, J[redacted]3,  (14) J[redacted]n (@J[redacted]3) / Twitter
twitter.com, J[redacted]3, with_replies,  (11) Tweets with replies by J[redacted]n (@J[redacted]3) / Twitter
twitter.com, J[redacted]3, likes,  (11) Tweets liked by J[redacted]n (@J[redacted]3) / Twitter
twitter.com, messages,  (13) Messages / Twitter
twitter.com, messages, 1097615[redacted]572668929,  (13) J[redacted]n
twitter.com, messages, 1031098[redacted]065132545-1031135833213034496,  (14) J[redacted]n / Twitter
```

Search parameters (search and view)
View content
View content (profiles, likes etc.)
Share content
Communication content (share content)
View content
Share or view content (needs corroboration)

Figure 9: Social media URL artefacts recovered by SMURF



△ URL	Referrer URL
https://twitter.com/J[redacted]3/with_replies	https://twitter.com/J[redacted]3
https://twitter.com/J[redacted]3/with_replies	https://twitter.com/J[redacted]3
https://twitter.com/N[redacted]d	https://twitter.com/home
https://twitter.com/N[redacted]d	https://twitter.com/home
https://twitter.com/N[redacted]d/following	https://twitter.com/N[redacted]d/with_replies
https://twitter.com/N[redacted]d/following	https://twitter.com/N[redacted]d
https://twitter.com/N[redacted]d/moments	https://twitter.com/home
https://twitter.com/N[redacted]/status/1293613020665520128	https://twitter.com/SouthPark
https://twitter.com/N[redacted]/status/1293619319788785665?cxt=HJwWgsC-6dv67fMjAAAA	https://twitter.com/home
https://twitter.com/N[redacted]d/with_replies	https://twitter.com/N[redacted]d
https://twitter.com/NovaCooks/status/1293611739074957312/photo/1	https://twitter.com/home
https://twitter.com/SYFY	https://twitter.com/home
https://twitter.com/S[redacted]3	https://twitter.com/J[redacted]3/following
https://twitter.com/S[redacted]3/following	https://twitter.com/S[redacted]3
https://twitter.com/SouthPark	https://twitter.com/J[redacted]3/likes
https://twitter.com/Trey_Parker	https://twitter.com/SouthPark

Figure 10: Social media URL artefacts recovered by Autopsy

This research has identified the possibility of enhancing the "Communications Visualization" feature in Autopsy to allow the identification and inclusion of social media artefacts from web browsers as discussed in Section 8.

7. DISCUSSION

This paper was focused on developing and demonstrating a framework that can be used for the recovery of artefacts from social media activity during a live triage investigation and Twitter has been used as a case study to demonstrate the feasibility of deploying SMURF as a live triage investigative tool, as shown by the results presented in this paper.

Communication was defined in Section 2, providing context to the type of artefacts that can be created and their recoverability. This was used as a means of defining what 'communication content' is in the context of this paper; thus creating a relationship between an action and the resultant artefact.

A baseline for defining the framework functionality based on perceived user activity and artefacts created was introduced as it was necessary to establish constraints for the purposes of testing the PoC.

Using the weighting matrix described in Section 6, this paper has highlighted some possible use cases for Twitter activities and the type of artefacts that can be recovered and has shown that context is important when attributing actions to a user. For example, distinguishing use cases where:

⇒ a user logs in and interacts with others
⇒ a user has an account but doesn't login (at least not on the device being investigated)
⇒ a user doesn't have an account but frequents Twitter

Although the data set used in the creation of the framework can be said to be a subset of real world user activity, this paper demonstrates that it is possible to recover a wider range of artefacts created as a result of social media activity. This was confirmed by the results from Autopsy showing that the same category of artefacts were recoverable in both Autopsy and SMURF.

While discussing context, when it comes to the availability or non-availability of communication content, this paper demonstrated the possibility of recovering enough information to allow inferences to be drawn about the communication (see Figure 9); thus, allowing the investigator to identify and prioritize areas for further analysis.

There are certain caveats to be aware of during the investigation of social media activity. For example, artefacts may exist as a result of background processes on the social media platform; a web browser's caching mechanism and/or web storage of content the user has not interacted with; hence the need for context and corroboration when assessing relevance and reliability.

It is important to highlight that evidence from live triage may not necessarily be court bound but may end up being used as supporting evidence; thus, it is important to adhere to the relevant jurisdictional guidelines for handling evidence in a live triage.

8. CONCLUSION AND FUTURE WORK

This paper has demonstrated a PoC for a Social Media User Relationship Framework (SMURF). It has demonstrated through the results recovered the possibility of using SMURF in a live triage, to obtain quick, actionable intelligence that can be used to guide or progress an investigation.

It has been shown that communication content can be inferred from other sources such as the page title. This is useful in a live triage where it may not be possible to conduct an extensive keyword search or file carving.

For attribution, a user account can be linked to specific searches enabling an investigator to build a timeline of contact and subsequent activity with a person of interest or a victim.

This paper has presented a PoC with the potential for improvement and the extensibility of its existing functions. A number of opportunities for future work have been identified. These include implementing an automated process for the categorisation of potential artefacts of interest for example the categories discussed in Section 6 and extending the current functionality for other triage investigations.

It is important to note that additional cache data such as image files were not recovered as part of the framework however, in the event that image files are required as part of a live triage investigation, further work would be required to implement the recovery of image and video files using a range of techniques including known file hashes and skin tone detection.

No machine learning (ML) or artificial intelligence (AI) techniques were implemented. This is an avenue for future work. With ML, the components can be trained to adapt to specific scenarios using pre-set case information thus improving and extending the current functionality.

As mentioned in Section 6.2, SMURF (or elements of it) could be added as a plugin to Autopsy or other live triage tools. This will extend its current functionality and allow for the visualization of social media activity from browsers.

As this demonstration was carried out on Windows 10, there is the potential for testing SMURF on other operating systems and on a variety of web browsers.

REFERENCES

7safe. (2014). *The ACPO Good Practice Guide for Managers of e-Crime investigation* (Tech. Rep.). Retrieved from `www.7safe.com`

ACPO. (2012, March). *Good Practice Guide for Digital Evidence.* Retrieved from `http://library.college.police.uk/docs/acpo/digital-evidence-2012.pdf` (Version: 5.0)

Arshad, H., Jantan, A., & Omolara, E. (2019). Evidence collection and forensics on social networks: Research challenges and directions. *Digital Investigation*, *28*, 126–138.

Ballenthin, W. (2019). *python-evtx.* Retrieved from `https://github.com/williballenthin/python-evtx`

Bancel, Jean-Rémy. (2015). *Chromagnon (SNSS Branch).* GitHub. Retrieved from `https://github.com/JRBANCEL/Chromagnon/tree/SNSS` (Latest commit 2cbecb1 on 28 Mar 2018)

Bashir, M. S., & Khan, M. N. A. (2013). Triage in Live Digital Forensic Analysis. *The International Journal of Forensic Computer Science*, *1*, 35–44. doi: 10.5769/J201301005

Basis Technology. (2020). *Free Autopsy Training.* Retrieved from `https://www.autopsy.com/support/training/covid-19-free-autopsy-training/`

Bello, M., & DiBlasio, N. (2013, sep). *Twitter: The new face of crime.* USA Today. Retrieved from `http://www.usatoday.com/story/news/nation/2013/09/29/twitter-crime-dark-side/2875745/`

Benson, R. (2019). *Hindsight.* Retrieved from `https://github.com/obsidianforensics/hindsight`

Blumenbach, T. (2015). *mozlz4a.py.* Retrieved from `https://gist.github.com/Tblue/62ff47bef7f894e92ed5`

Cambridge University Press. (2019). *Cambridge Dictionary [Online].* Retrieved from `http://dictionary.cambridge.org/`

Cantrell, G., Dampier, D., Dandass, Y. S., Niu, N., & Bogen, C. (2012). Research toward a Partially-Automated, and Crime Specific Digital Triage Process Model. *Computer and Information Science*, *5*(2). Retrieved from `www.ccsenet.org/cisURL:http://dx.doi.org/10.5539/cis.v5n2p29` doi: 10.5539/cis.v5n2p29

Cantrell, G., & Dampier, D. A. (2012). Implementing the Automated Phases of the Partially-automated Digital Triage Process Model. *Journal of Digital Forensics, Security and Law*, *7*(4). Retrieved from `https://commons.erau.edu/jdfsl/vol7/iss4/5/`

Carrier, B. (2020). *Autopsy 4.15.0.* GitHub. Retrieved from `https://github.com/sleuthkit/autopsy/releases/`

Casey, E. (2004, jan). Digital evidence and computer crime: Forensic Science, Computers and the Internet. *Elsevier Academic Press*, 215.

Casey, E. (2011). Digital Evidence and Computer Crime, Forensic Science, Computers and the Internet. In (Third Edition ed., chap. 1: Foundations of Digital Forensics). Elsevier Inc.

Casey, E. (2013). Triage in digital forensics. *Digital Investigation*, *10*, 85–86.

Cusack, B., & Son, J. (2012). Evidence Examination Tools for Social Networks. In *10th australian digital forensics conference* (pp. 33–40). SRI Security Research Institute, Edith Cowan University, Perth, Western Australia. doi: 10.4225/75/57b3afc1fb861

Dance, F. E. X. (1967). *Towards a Theory of Human Communication (In Human Communication Theory: Original Essays)*. Holt, Rinehart and Winston, New York.

David, A., Morris, S., & Appleby-Thomas, G. (2020). A Two-Stage Model for Social Network Investigations in Digital Forensics. *Journal of Digital Forensics, Security and Law*, *15*(1). Retrieved from `https://commons.erau.edu/jdfsl/vol15/iss2/1`

DFRWS. (2001). *A Road Map for Digital Forensic Research: DFRWS Technical Report* (Tech. Rep. No. DTR - T001-01). DFRWS: Digital Forensic Research Workshop.

Garfinkel, S. L. (2013, feb). Digital media triage with bulk data analysis and bulk_extractor. *Computers & Security*, *32*, 56–72. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0167404812001472` doi: 10.1016/J.COSE.2012.09.011

Gielen, M., & Bolzoni, D. (2014). *Prioritizing Computer Forensics Using Triage Techniques* (Tech. Rep.). Retrieved from `https://essay.utwente.nl/65671/1/Gielen_MA_EWI.pdf`

Haroon, S., & Carter, H. (2010, mar). *Facebook security measures criticised after Ashleigh Hall murder.* The Guardian. Retrieved from `http://www.theguardian.com/uk/2010/mar/09/ukcrime-facebook`

Hitchcock, B., Le-Khac, N.-A., & Scanlon, M. (2016). Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists. *Digital Investigation*, *16*(Supplement), S75–S85. Retrieved from `http://dx.doi.org/10.1016/j.diin.2016.01.010` doi: 10.1016/j.diin.2016.01.010

Hubert, K. (2014). *Evidence Collection From Social Media Sites.* SANS Institute Information Security Reading Room. Retrieved from `https://www.sans.org/reading-room/whitepapers/legal/evidence-collection-social-media-sites-35647`

Jusas, V., Birvinskas, D., & Gahramanov, E. (2017, mar). Methods and Tools of Digital Triage in Forensic Context: Survey and Future Directions. *Multidisciplinary Digital Publishing Institute (MDPI)*, *9*(4), 49. Retrieved from `http://www.mdpi.com/2073-8994/9/4/49` doi: 10.3390/sym9040049

Littlejohn, S. W. (1992). *Theories of Human Communication.* Belmont, Calif.: Wadsworth Pub. Co.

Marcella, A. J., & Menendez, D. (2007). *Cyber Forensics: A Field Manual for Collecting, Examining, and Preserving Evidence of Computer Crime* (Second Edition ed.). NEW YORK: CRC PRESS - TAYLOR AND FRANCIS.

McKemmish, R. (2008). When is Digital Evidence Forensically Sound? *Advances in Digital Forensics*, *IV*, 3–15.

Montasari, R. (2016, jun). Formal Two Stage Triage Process Model (FTSTPM) for Digital Forensic Practice. *International Journal of Computer Science and Security (IJCSS)*, *10*(2), 69–87. Retrieved from `https://pure.hud.ac.uk/en/publications/formal-two-stage-triage-process-model-ftstpm-for-digital-forensic`

Moore, K. (2014, June). *Social media 'at least half' of calls passed to front-line police.* BBC News. Retrieved from `https://`

`www.bbc.co.uk/news/uk-27949674`

Mukasey, M. B., Sedgwick, J. L., & Hagy, D. W. (2008, April). *Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition.* U.S. Department of Justice, Office of Justice Programs, National Institute of Justice. Retrieved from `https://www.ncjrs.gov/pdffiles1/nij/187736.pdf`

Parsonage, H. (2009). *Computer Forensics Case Assessment and Triage.* (`http://computerforensics.parsonage.co.uk/triage/ComputerForensicsCaseAssessmentAndTriageDiscussionPaper.pdf`)

Robertson-Steel, I. (2006). Evolution of triage systems. *Emergency medicine journal*, *23*(2), 154–155. doi: doi:10.1136/emj.2005.030270

Rogers, M. K., Goldman, J., Mislan, R., Wedge, T., & Debrota, S. (2006). Computer Forensics Field Triage Process Model. *Journal of Digital Forensics, Security and Law*, *1*(2). Retrieved from `https://commons.erau.edu/cgi/viewcontent.cgi?article=1004&context=jdfsl` doi: 10.15394/jdfsl.2006.1004

Roussev, V., & Quates, C. (2012). Content triage with similarity digests: The M57 case study. *Digital Investigation*, *9*, S60–S68. doi: 10.1016/j.diin.2012.05.012

Russinovich, M. (2016). *PsTools Suite Windows Sysinternals | Microsoft Docs.* Retrieved from `https://docs.microsoft.com/en-us/sysinternals/downloads/pstools`

Russinovich, M. (2018). *Process Monitor - Windows Sysinternals | Microsoft Docs.* Retrieved from `https://docs.microsoft.com/en-us/sysinternals/downloads/procmon`

Select Committee on Communications. (2014, July). *CHAPTER 2: SOCIAL MEDIA AND THE LAW* . Retrieved from `https://publications.parliament.uk/pa/ld201415/ldselect/ldcomuni/37/3702.htm`

Shaw, U., Das, D., & Mehdi, S. P. (2016). Social Network Forensics: Survey and Challenges. *International Journal of Computer Science and Information Security (IJCSIS)*, *14*(11), 310–316.

Sommer, P. (1999). Intrusion Detection Systems as Evidence. *Computer Networks*, *31*(23–24), 2477–2487.

TWGECSI. (2001). *Technical Working Group Electronic Crime Scene Investigation - Electronic Crime Scene Investigation: A Guide for First Responders.*

Twitter Help Center. (2019). *New user FAQs.* Retrieved from `https://help.twitter.com/en/new-user-faq`

Wiles, J., & Reyes, A. (2007). Incident Response: Live Forensics and Investigations. In (pp. 89–109). Syngress.

WinPython. (2019). *winpython.* Retrieved from `https://github.com/winpython/winpython`

Zeng, D., Chen, H., Lusch, R., & Li, S. (2010, Nov). Social Media Analytics and Intelligence. *IEEE Intelligent Systems*, *25*(6), 13-16. doi: 10.1109/MIS.2010.151

Zimmerman, E. (2019). *Prefetch Explorer Command Line - PECmd version 1.3.4.5.* Retrieved from `https://github.com/EricZimmerman/PECmd`

A.

A.1 Extracts from predefined_paths.py

```
def current_user_keys():
# keys of interest in the current user hive
# HKCU\Software\Microsoft\Windows\CurrentVersion
# HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer
# HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32
# HKCU\Software\Microsoft\Windows\Shell\BagMRU
# HKCU\Software\Clients

   comdlg_set = r"Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\
                                            CIDSizeMRU", \
                r"Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\
                                                         OpenSavePidlMRU", \
                r"Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\
                                                         LastVisitedPidlMRU", \
                r"Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs",
                                                         \
                r"Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU", \
                r"Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2"
                                                         , \
                r"Software\Microsoft\Windows\CurrentVersion\Explorer\Taskband", \
                r"Software\Microsoft\Windows\CurrentVersion\Run", \
                r"Software\Microsoft\Windows\CurrentVersion\RunOnce", \
                r"Software\Microsoft\Windows\Shell\Bags\1\Desktop", \
                r"Software\Clients\StartMenuInternet", \
                r"Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist",

   return comdlg_set
```

A.2 Extracts from app_config.py

```
# predefined registry keys. These can be updated in 'predefined_hives.py'
prog_run = predefined_hives.software_keys()


def local_software():
   for path in prog_run:
       try:
           i = 0
           while True:
               get_keys = OpenKeyEx(HKEY_LOCAL_MACHINE, path, 0, KEY_READ |
                                                   reg_view_flag)
               name, value, data_type = EnumValue(get_keys, i)
[...]
               file_exists = pathlib.Path(sw_csv_file).is_file()

               with open(sw_csv_file, 'a+') as skv:
                   fields = ["path", "name", "value", "data_type"]
```

```
                    csvwriter = csv.DictWriter(skv, delimiter=',', lineterminator=
                                                              '\n',
                                                   fieldnames=fields)

                    if not file_exists:
                        csvwriter.writeheader()  # if file doesn't exist, write a
                                                                header

                    skv.write("{0}, {1}, {2}, {3} ".format(path, name, value,
                                                              data_type)  + '\n')
                i += 1
[...]
```

A.3 Extracts from system_info.py

```
# Prefetch
pref_out_file = "output_files/"

# set %winroot% as the OS partition then join %winroot% to path tho evtx logs
win_root = os.environ['WINDIR']
prefetch_dir = os.path.join(win_root, 'Prefetch\\')


def parse_pf():
   # if str(path).endswith(".pf"):
   [...]
   logging.info("Parsing {}".format(prefetch_dir))

   subprocess.call([r"PECmd\PECmd.exe", "-d", prefetch_dir, "--json",
                                            pref_out_file],
                    shell=True)
```

```
# System events
# set %winroot% as the OS partition may not be C: by default.
# join %winroot% to path tho evtx logs
win_root = pathlib.Path(os.environ['WINDIR']).drive
evtx_dir = pathlib.Path(win_root).joinpath('\\Windows', 'System32', 'winevt', '
                                                Logs')


def parse_logs():
   for each in evtx_dir.iterdir():
       i = str(each)

       e_logs = ["Application.evtx", "Security.evtx"]

       for log in e_logs:

           if i.endswith("Application.evtx"):
               evtx_file = i
```

```
                logging.info("Parsing {}: ".format(evtx_file))
                subprocess.run([r"python", r"third_party\bin\python_evtx\scripts\
                                                        evtx_dump.py", evtx_file,
                                                        ">>", app_out_file], shell
                                                        =True)
            elif  ...
            [...]
            else:
                 continue

    subprocess.run([r"third_party\bin\sysinternals\psloglist64", "-accepteula", ">"
                                                , process_loglist], shell=True)

    [...]
```

```
# Parse XML

# Parse the python_evtx output file
tree = etr.iterparse(dest, remove_pis=True, huge_tree=True, remove_blank_text=True
                                          , resolve_entities=True, recover=True)

[...]

for event, elem in tree:
    for child in elem:
        results = child.tag, child.text

        with open(evt_results, 'a', encoding="utf-8") as ft:
            # ft.write(child.tag, child.text)
            ft.write("{0}{1}{2}{3}{4}".format(child.tag, ': ', child.text, ' ', '\n
                                                               '))
        sys.stdout.write("{0}{1}{2}{3}{4}".format(child.tag, ': ', child.text, ' ',
                                                            '\n'))
```

A.4 Extracts from session_info.py

```
# sessionstore
def sessionstore_info():
    try:
        import lz4.block as lz4
    except ImportError:
        import lz4

    for each_dir in profile_dir.iterdir():
        for i in each_dir.iterdir():
            i = str(i)

     if  'sessionstore' in i and 'lz4' in i:
         file_name = i
      get_data = open(file_name, "rb")
      read_bytes = get_data.read()
       get_sig = read_bytes[:8]  # verify header
      if bytearray(get_sig).startswith(b"mozLz40\0"):
    json_data = json.loads(lz4.decompress(read_bytes[8:]).decode("utf-8"))

      with open(json_data_sstore, 'a+') as jd:
     jd.write(str(json_data))

       for win in json_data.get("windows"):
         for tab in win.get("tabs"):
        i = int(tab.get("index")) - 1
      urls = tab.get("entries")[i].get("url")
       [...]
       last_accessed = win.get("tabs")[i].get("lastAccessed")  tab_last_accessed =
                                                 dt.datetime.utcfromtimestamp(
                                                 last_accessed//1000.0)         [...
                                                 ]
```

```
# the path to profile folder is '\\AppData\\Local\\Google\\Chrome\\User Data\\
                                          Default'
# pathlib.Path.home() joins the user's home directory path to the profile path
chrome_dir = pathlib.Path.home().joinpath('AppData', 'Local', 'Google', 'Chrome',
                                          'User Data', 'Default')

user_profile_dir = pathlib.Path(chrome_dir).glob('*')


def chrome_data():
   # Check if chrome.exe is running and terminate process before parsing
   # Chrome places a lock on the browser files when  it's running
   for proc in psutil.process_iter():
       try:
           # process_name = proc.name()
           # check if chrome.exe is running
           if 'chrome.exe' in proc.name():
               proc.kill()
       except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
           continue

   time.sleep(5)

   # Parse current session
   c_file = str(chrome_dir) + r"\Current Session"
   logging.info("Parsing: {}".format(c_file))
   subprocess.run(["python", r"chromagnonSession.py", c_file, ">", current_session
                                             ],
                  shell=True)
[...]
```

A.5 Extracts from browser_history.py

```
# Firefox History
# the path to profile folder is '\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\'
# pathlib.Path.home() joins the user's home directory path to the profile path
home_dir = pathlib.Path.home().joinpath('AppData', 'Roaming', 'Mozilla', 'Firefox'
                                        , 'Profiles')

profile_dir = pathlib.Path(home_dir)


def user_places():
   for each_dir in profile_dir.iterdir():
       for i in each_dir.iterdir():
           i = str(i)

           if 'sqlite' in i:
               file_name = i

               # establish a connection to the database
               conn = sq3.connect(file_name)
               cursor = conn.cursor()

               # query for places.sqlite
               sql = "SELECT datetime(visit_date/1000000, 'unixepoch') as
                                                    visitDate, url, title " \
                     "FROM moz_places, moz_historyvisits " \
                     "WHERE moz_places.id = moz_historyvisits.place_id " \
                     "ORDER BY visitDate "


[...]
```

```
  with open(places_csv, 'a+', encoding='utf-8') as ud:
     fields = ["visit_datetime", "url", "title"]
     writer = csv.DictWriter(ud, delimiter=',', lineterminator='\n', fieldnames=
                                                fields)

      if not file_exists:
         # create file, write a header
              writer.writeheader()

      ud.write("{0}, {1}, {2}".format(each_record["visit_datetime"],
            each_record["url"],
             each_record["title"]) + '\n')

         with open(places_csv) as rd:
           readCSV = csv.reader(rd, delimiter=',')
             for row in readCSV:
            for item in row:
            ml = ['twitter', 'facebook', 'drive.google', 'docs.google']
           for j in ml:
          if j in item:
        patt = r'((?<=\/)[^:\/\s]*[?*\w\w]+)'
        regexp = re.compile(patt, re.IGNORECASE)
        matches = regexp.findall(item)
        with open(parsed_url, 'a+') as pu:
      pu.write("{0}, {1}".format(row[0], ', '.join(
        matches)) + '\n')
```

```
# Chrome History

# Parse history
h_file = str(chrome_dir) + r"\History"
logging.info("Parsing: {}".format(h_file))
subprocess.run(["python", r"chromagnonHistory.py", h_file, ">", chrome_hist_file],
              shell=True)

with open(chrome_hist_file, 'r') as ch:
   for line in ch:
       get_each_val = line.split(None, 3)
       cut_date_val = [i.split('.', 1)[0] for i in get_each_val[:-2]]

       ml = ['twitter', 'facebook', 'drive']
       for j in ml:
           if j in line:
               patt = r'((?<=\/)[^:\/\s]*[?*\w\w]+)'
               regexp = re.compile(patt, re.IGNORECASE)
               matches = regexp.findall(line)

               with open(chrome_parsed_url, 'a+') as pur:
                   pur.write("{0}, {1}".format(" ".join(cut_date_val), ', '.join(
                                                            matches)) + '\n')
```

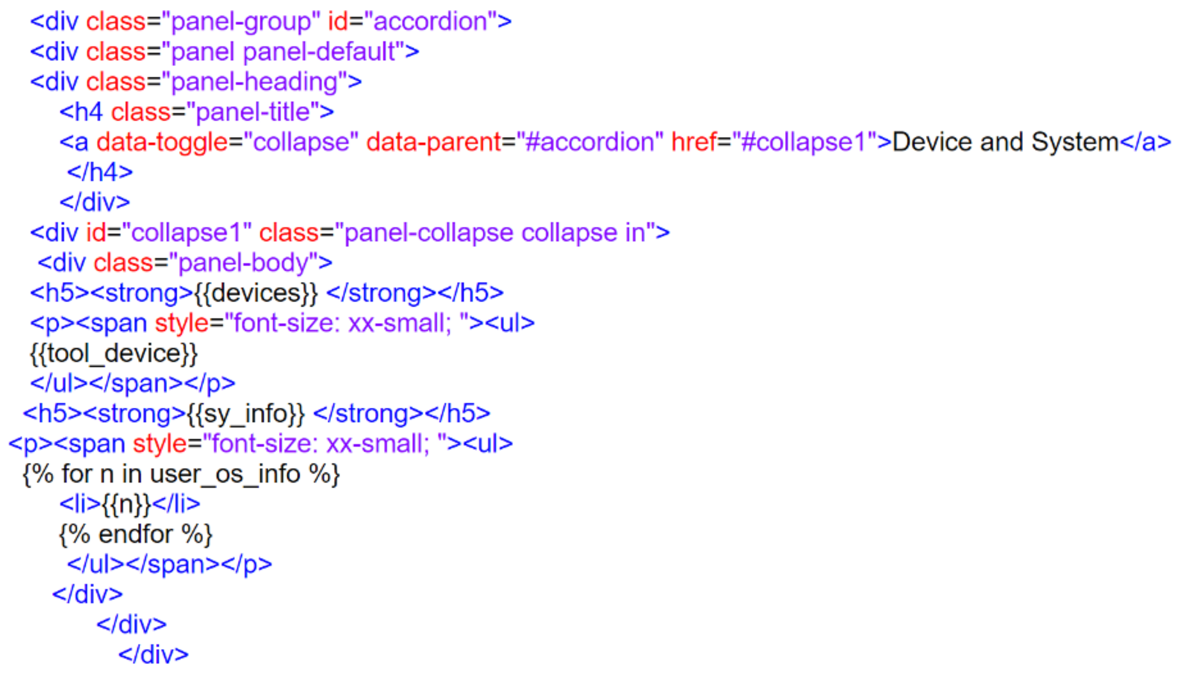
A.6 Extracts from browser_cache.py

```
# the path to the Firefox cache2 profile folder is '\\AppData\\Local\\Mozilla\\
                                          Firefox\\Profiles\\'
# pathlib.Path.home() joins the user's home directory path to the profile path
cache_profile_dir = pathlib.Path.home().joinpath('AppData', 'Local', 'Mozilla', '
                                          Firefox', 'Profiles')

cache_dir = pathlib.Path(cache_profile_dir)

# iterate through the cache profile directory
for each_dir in cache_dir.iterdir():
    for i in each_dir.iterdir():
        i = str(i)

        if i.endswith('cache2'):
            cache2_path = i

[...]

with open(cache_idx_out_file, 'a+') as cix:
    cix.write("Cache version: {}".format(cache_version) + '\n')
    cix.write("Index last written time: {}".format(unix_written_time) + '\n')
    [...]
    cix.write("Index_Hash is: {}".format(binascii.hexlify(index_hash).upper().
                                              decode("ascii")) + '\n')
    cix.write(" " + '\n')

    count += 1
    more_records = file_size - cif.tell()

[...]
```

```
# write to text file
        with open(cache_data_file, 'a+') as cdf:
            cdf.write("Parsing file: {0}".format(cache_entry_file) + '\n')
            cdf.write("Cache file is written in version: {0}".format(version) + '\
                                                            n')
            cdf.write("Cache entry file size: {0}".format(file_size) + '\n')
            cdf.write("Start of cache metadata: {0}".format(metadata_start) + '\n'
                                                            )
            [...]
            cdf.write("Key size: {}".format(key_size) + '\n')
            cdf.write("Key (URL): {}".format(key) + '\n')
            cdf.write("Key hash (SHA1 hash of url): {}".format(key_hash) + '\n')
            cdf.write(" " + '\n')
[...]
```

```
# Google Chrome

[...]
# Parse cache
chrome_d = str(chrome_dir)
chrome_cache = str(chrome_dir) + r"\Cache"
print("Parsing: {}".format(chrome_cache))
logging.info("Parsing: {}".format(chrome_cache))
subprocess.run(["python", r"hindsight\hindsight.py", "-i", chrome_d, "-o",
                                          chrome_cache_file,
               "-f", "jsonl"], shell=True)
```

A.7 Extracts from SMURF HTML Report template



```
  <div class="panel-group" id="accordion">
  <div class="panel panel-default">
  <div class="panel-heading">
    <h4 class="panel-title">
    <a data-toggle="collapse" data-parent="#accordion" href="#collapse1">Device and System</a>
     </h4>
    </div>
  <div id="collapse1" class="panel-collapse collapse in">
   <div class="panel-body">
  <h5><strong>{{devices}} </strong></h5>
  <p><span style="font-size: xx-small; "><ul>
  {{tool_device}}
  </ul></span></p>
 <h5><strong>{{sy_info}} </strong></h5>
<p><span style="font-size: xx-small; "><ul>
 {% for n in user_os_info %}
    <li>{{n}}</li>
    {% endfor %}
     </ul></span></p>
   </div>
       </div>
         </div>
```