How long does it take to get owned?

David Wardle

Technical Report

RHUL–ISG–2019–4

27 March 2019

Information Security Group
Royal Holloway University of London
Egham, Surrey, TW20 0EX
United Kingdom

Student Number: 100862122

David Wardle

How long does it take to get owned?

Supervisor: Jorge Blasco Alís

Submitted as part of the requirements for the award of the
MSc in Information Security
at Royal Holloway, University of London.

I declare that this assignment is all my own work and that I have acknowledged all quotations from published or unpublished work of other people. I also declare that I have read the statements on plagiarism in Section 1 of the Regulations Governing Examination and Assessment Offences, and in accordance with these regulations I submit this project report as my own work.

Signature:

Date:

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr Jorge Blasco Alís. Not only did he come up with the fantastic idea for this project, he has provided a lot of support and guidance during the course of it. Thank you.

Thank you too to Liam, Marcos, Nick and my wife, Lucy, for taking the time to proofread this report and for all of their helpful feedback.

Finally, thank you to my family for their patience over these last few months.

This project is dedicated to the memory of my Father.

This page intentionally left blank

Executive summary

This report investigates the use of stolen credentials; more specifically measuring the time it takes for them to be used after they have been leaked. By way of a literature review and technical research, it begins by looking at the concept of digital identity and the use of honeypots in information security. It then presents the different approaches and techniques that can be used to monitor access to an online account. It also describes the methods adopted by cybercriminals to illegally share personal data including passwords.

The report continues by presenting a framework that was designed to create fake online identities along with an infrastructure to monitor their activity. The design was implemented using a combination of manual processes and software developed for this project. The implementation was tested by publishing the credentials for eleven fake identities on paste websites. Over the course of six weeks, five events of unauthorised access were recorded, with the fastest occurring just 34 minutes after the leak.

The report concludes by discussing the results of the experiment, recommending improvements that can be made to the framework and proposing opportunities for future work.

This page intentionally left blank

Table of contents

List of tables

List of figures

List of acronyms

2FA	Two factor authentication
API	Application programming interface
APT	Advanced Persistent Threat
AWS	Amazon Web Services
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CPU	Central processing unit
CSO	Chief Security Officer
DNS	Domain Name System
GDPR	General Data Privacy Regulation
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IMAP	Internet Message Access Protocol
IP	Internet Protocol
ISP	Internet Service Provider
JSON	JavaScript Object Notation
OSINT	Open-source intelligence
POP3	Post Office Protocol version 3
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
Tor	The Onion Router
UI	User interface
URL	Uniform Resource Locator
VPN	Virtual Private Network
VPS	Virtual Private Server

1 Introduction

News stories regarding major data breaches have become a regular occurrence in recent years. Last year saw over 53,000 security incidents leading to 2,216 confirmed data breaches according to a report conducted by Verizon [1]. Another recent survey found that 26% of Americans, an estimated 64 million adults, received a breach notification in 2016 [2]. Uber, Equifax, CarPhone Warehouse, Wonga, Ticketmaster, and Yahoo are just a few examples of well-known companies who have acknowledged a large data breach in the last two years [3]. The stolen datasets, many containing sensitive information such as passwords, are often subsequently made publicly available on the Internet.

A leaked password can have serious consequences if it is re-used across a number of different services. Studies suggest that users struggle to cope with the cognitive burden of remembering a large number of complex passwords and so re-use the same password across most of their accounts. Research, conducted in 2017, found that 38% of users re-used a password on at least two different services [4]. Another recent study put this figure at 67% [5], and the numbers are only increased once partial re-use is considered [4], [5].

Credential stuffing has become a serious threat. This attack automates the process of logging into another ("un-breached") website using stolen credentials and is one of the most common techniques used to a take control of a user account. An attacker can then syphon the compromised account of its stored value, financial information and other personal information [6]. Mitigation techniques (such as password managers and multi factor authentication) have failed to achieve mass adoption. It is no surprise that Alex Stamos, the outgoing Facebook CSO, believes that the re-use of passwords is the number one cause of harm on the internet [7].

In this project, the credentials for eleven fake identities were published online, mimicking the approach used by cybercriminals to share illicit data. The use of these credentials was then monitored for a period of six weeks to answer the question: *how long does it take to get owned?*[1]

[1] *Owned* is a slang term for being hacked.

1.1 Objectives and scope

The ultimate goal of this project was to measure how long it takes for stolen credentials to be used once leaked. To achieve this goal, the following objectives were set:

1. Understand the concept and make up of a digital identity.
2. Analyse the different techniques available to monitor access to a web-based account.
3. Propose a framework to:
    a. Generate fake digital identities ("honey identities").
    b. Monitor access and usage of these honey identities.
4. Conduct an experiment using a prototype implementation of the framework to:
    a. Generate a number of honey identities.
    b. Publish their credentials online.
    c. Monitor access over a period of time.

By completing these objectives, it was hoped to gain a better understanding of how cybercriminals make use of stolen credentials and the potential damage that can be caused by password re-use.

The scope for this project was defined by several constraints. The web services chosen to make up the honey identity were limited due to various ethical, practical and technical reasons. The credentials for the accounts were leaked rather than waiting for them to be exposed in a data breach. Furthermore, only one avenue of publishing the credentials (paste websites) was used in the experiment phase. Finally, due to the time restrictions of the MSc, the observation period for monitoring access was limited to just over six weeks.

1.2 Methodology

The nature of this project needed an approach that used several different methods to fulfil the objectives. These included:

- **Literature review.** It was important to gain a better understanding of several background topics (*Chapter 2*). It was also vital to look at related work and see how others had approached similar experiments (*Chapter 4*).
- **Implementation research.** This covered research into techniques that could be used to monitor access of a web account (*Chapter 2*), selecting appropriate web services based on a defined criterion (*Chapter 3*), software tools to be used during the project and other design considerations (*Chapter 4)*.
- **Implementation.** The most time-consuming part of the project was the development of a prototype framework and monitoring infrastructure designed in this project (*Chapter 4*).
- **Experiment.** The developed framework was used to create a number of identities, publish the credentials and monitor access over a period of six weeks. The experiment is presented in *Chapter 5* and the findings are discussed in *Chapter 6*.

This page intentionally left blank

2 Background

This section introduces four key areas of this project: the building blocks of (digital) identity; the use of honeypots in information security; the various methods that can be used to monitor unauthorised access to online accounts; and the illegal sharing of stolen credentials.

2.1 Identity

In order to create a fake identity, a basic understanding of "Identity" in the digital age was first required. In other words, what attributes and components are needed to make a fake identity?

The majority of research involving the concept of "digital identity" falls into two distinct topics: (social) psychology and privacy. Surprisingly, there is no common terminology when discussing personal data; even the phrase "digital identity" has several contrasting definitions. Indeed, there is research that discusses the different terms used and proposes new models [8]. Whilst it is out of the scope of the project to come up with a new meaning, it is worthwhile to briefly look at some prominent publications from both categories.

2.1.1 Psychology

From the psychological point of view, identity is the basis for an old philosophical question: "Who am I?". Rodogno looked at this question in his paper discussing the effect of the Internet on Personal Identity [9]. He identified six overlapping types of identity; shown in Table 2.1.

When it comes to online activities, the same person may have distinct and conflicting online and offline identities. It is not the case of a single identity having different attributes but of complete identities in the plural. However, the fake identities created for this project exist purely in databases; there is no real person behind them. For that reason, the definition of "Passport Identity" is the most relevant one for our purposes.

Type	Definition
Passport Identity	A Customs Officer at Immigration will want to know who I am. Presenting my passport provides information about my appearance, name, sex, date of birth, nationality, and place of residence. This personal information is sufficient for our common purposes.
Numerical Identity	Using the example of a talented musician and professor, who after having a devastating stroke, can no longer play an instrument and struggles to remember even his closest friends. Is this pre-stroke person the same as the post-stroke one? This is another long-established philosophical question referred to as the re-identification or persistence question.
Attribution Identity	What are the conditions under which various psychological characteristics, experiences, and actions are properly attributable to some person? This is called the characterization question.
Social Function Identity	Someone introducing them self as "the repairman" when you are expecting someone to fix your washing machine would be sufficient information about that person's identity for your common purposes.
Attachment Identity	More in line with what people consider to be a "deeper" understanding of personal identity. It may have little in common with one's social role and points to things to which we are attached – things that we care about or matter to us.

Table 2.1: Different types of Identity defined in [9]

2.1.2 Privacy

Psychological reports can help one understand the deeper meaning of "Identity", but they do not provide guidance to the attributes that make up an identity. For the purpose of this project it is, therefore, better to look at the topic from a Privacy point of view.

The concept of a "digital persona" was introduced by Roger Clarke in 1994 [10]. He proposed the following definition:

> *a model of an individual's public personality based on data and maintained by transactions, and intended for use as a proxy for the individual.*

Clarke identified a number of key characteristics of digital personae leading to several different categories. Like Rodogno, Clarke states that one person may have many digital personae in order to present themselves in a different way to different people or the same person at different times. These definitions are summarised in Table 2.2.

Category	Description
Informal persona	A persona based on human perception
Formal persona	A persona constructed on the basis of accumulations of structured data
Projected persona	A persona controlled by the individual it is associated with
Imposed persona	A persona controlled by someone other than the individual it is associated with
Passive persona	A persona that comprises of data alone
Active persona	A persona that has some capacity to act, on behalf of, or in substitution for, the individual it is associated with, cf. a software agent
Public persona	A persona that presents a commonly-held, composite image of a person who is presumed to be well-known (e.g. Marilyn Monroe, Marshall McLuhan) or of an archetype (e.g. "action man", "drug mule", "psychopath")

Table 2.2: Categories of digital personae from [10]

Using Clarke's definitions, it may be more appropriate to describe the fake identities in this project as "passive digital personae". However, this does not go any further in providing the building blocks than the psychological research.

The Internet Society [11] is an international campaign group working towards the goal of an open, secure and trustworthy Internet. The terminology in their literature (Table 2.3) is rather different from that of Clarke's.

Concept	Definition
Identity	Complete set of characteristics that define you. E.g. Name, nicknames, birth date, and any unique characteristics that combine to make you who you are.
Identifier	A way of referring to set of characteristics. E.g. Your email address (myID@me.com) or username (RaulB) or an account number. Usually an 'index' to other data held about you.
Partial identity	A subset of the characteristics that make up your identity. E.g. Demographic information about you or any purchase history is stored in your account at a website
Profile	Information collected by others about your actions and characteristics. E.g. A search your conducted for "discount shoes" or a list of websites visited. Your profile may also be based on inference data. For instance, a service provider has a certain number of data points, they will use those as the basis to infer other things about you.
Persona	A partial identity created by you to represent yourself in specific situation. E.g. a social network account or your online blog

Table 2.3: Terminology used by the Internet Society [11]

The International Telecommunications Union (ITU) is the United Nations agency for information and communication technologies. They published a report in 2006 called "digital.life" in which they looked at the "digital individual" from a security and privacy point of view [12]. Again, the terminology that the ITU use differs from that discussed above.

> *Digital Identity refers to the online representation of identity. More specifically, it refers to the set of claims (in their digital form) made about a user or another digital subject.*

Digital claims are sets of data, also known as attributes or identifiers. Attributes can include a name, phone number, bank balance, but also past purchases or employment records (Figure 2.1). They can be static (such as place of birth) or dynamic (such as employer's name).

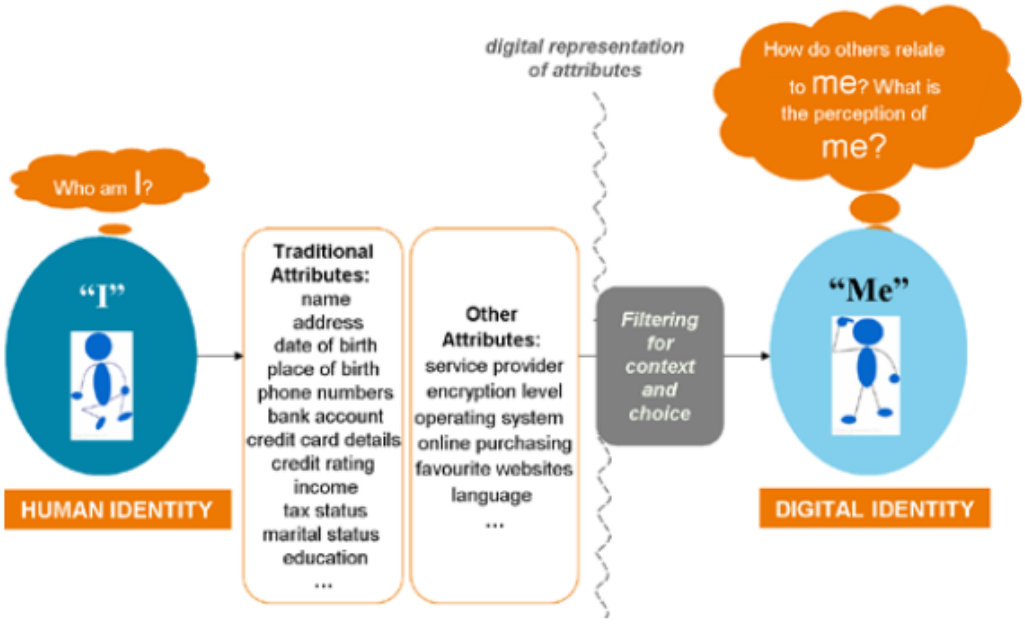


Figure 2.1. From "I" to "Me" [12]

Different digital identities exist in a specific context and the context will determine which subset of attributes are required. This is known as a "partial identity" and is close to the Internet Society's definition of the same term. Figure 2.2 shows examples of the many partial identities of Alice. She may share her name and address with both her health care provider and employer, but only the former will know her blood group and health status.

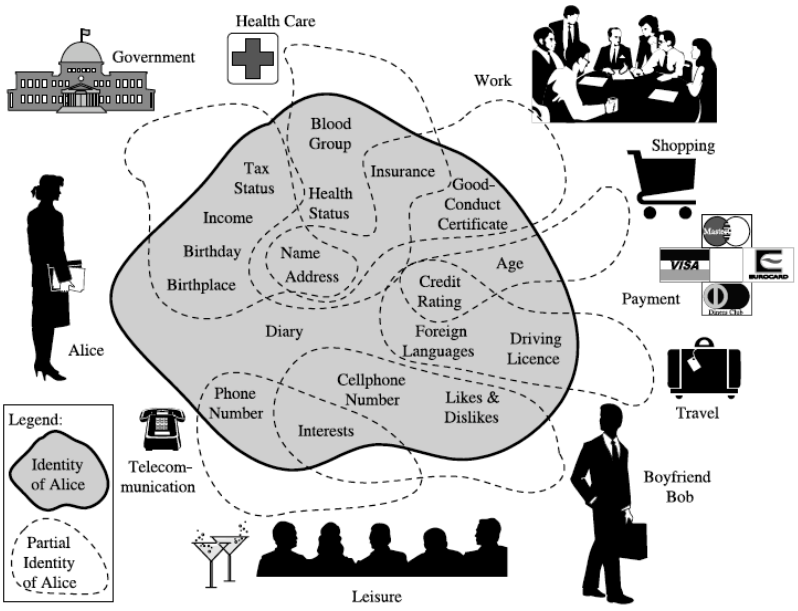


Figure 2.2. Partial identities of Alice [13]

The ITU terminology offered the most helpful guidance for this project, providing examples of the different attributes that can be used as digital claims. However, in the Internet age, it is almost impossible to quantify or define the huge number of data points that comprise personal information [14].

2.1.3 The terminology used in this project

It is important not to get side-tracked about the differences in terminology used across the different research in different contexts. Since there are no universally accepted definitions, it is often simply the personal preference of the author.

A more simplistic definition, offered by the authors of a patent for managing digital identity information, may be the most appropriate for this project [15]:

The summation of a person's personal data.

This can be complemented with the term "digital footprint" [8]:

Data descriptive of an individual, laid down by that individual as a result of using, or being observed by, computing devices.

In other words, the personal data that comprises one's digital identity leaves a digital footprint when it is used on a computer (or, in this case, a website).

The design of a fake digital identity and its encompassed digital footprint will be discussed in detail in the next chapter.

2.2 Honeypots

The fake identities served as a trap aimed at detecting unauthorised access to various web services; in other words, they were a honeypot. Such a concept has been widely used as a network defence since the early 1990s. However, it wasn't until 2003 when Spitzner [16] introduced the widely accepted formal definition:

> *A honeypot is a security resource whose value lies in being probed, attacked or compromised.*

Today, there are a vast number of honeypot tools that offer a variety of different services such as detecting malware and thwarting spam. They are often used to complement other security resources such as Intrusion Detection Systems and firewalls [17].

The use of honeypot accounts was in the news recently when, during the 2016 French presidential elections, a 9 GB collection of emails from Emmanuel Macron's party were posted online. However, it quickly turned out that the majority of this information was fake, and part of a strategy to protect against an Advanced Persistent Threat (APT). To slow down the attack, Macron's team flooded phishing attacks with fake data to hide the credentials of any users that may have been tricked. Additionally, they also planted the credentials for a number of honeypots accounts. These accounts contained a number of fake documents to not only create a large amount of disinformation when they were eventually leaked but also to help identify the hackers. [18]–[20]

Spitzner also proposed the term "honeytoken" for a honeypot which is not a computer. They can take the form of any digital entity such as a credit card number, Word document, URL or a special login. The different formats all have the same concept: their value lies in the unauthorised use of that resource [21].

2.3 Monitoring unauthorised access to online accounts

In order to construct a honeypot account, it is crucial that the website offers functionality that can be used to monitor the access and activity of that account. In this section, a number of different possible techniques are discussed independent of any actual web service. It is often the case that a single method will only provide a limited amount of information regarding any unauthorised access. However, a combination of several could provide enough detail to formulate a picture of the intruder[1] and their motives.

Two Factor Authentication – SMS Verification

Two Factor Authentication (2FA) is an authentication mechanism that has been adopted by many websites to increase login security. SMS verification is the most common, albeit least secure, form of 2FA [22]. At its most basic, a text message containing a verification code will be sent to the user when they try to log into the site. Knowledge of this code proves possession of their mobile phone and, along with the valid password, thus greater confidence in the user's identity. An example of this kind of text message is shown in Figure 2.3.

Figure 2.3: An example of a 2FA verification SMS from Airbnb

Receiving one of these notifications will demonstrate that valid credentials have been used which would be sufficient for the purposes of this project. However, there are several limitations to this approach. Firstly, since an intruder would not be able to receive the text message then they would not gain full access to the account. This means that it would not be possible to gain an insight into the intruder's motives through their activity within the

[1] A number of different terms were considered to describe those who use stolen credentials, such as *cybercriminal*, *miscreant*, *attacker* and *credential seeker*, but it was felt that *intruder* was most appropriate.

account. Additionally, the actual contents of the message would not provide any useful information, such as an IP address, that could be used to link the attacker to other incidents.

New login notification

When a user logs into their account for the first time from a new device and/or location, an email is sent to the user containing information regarding this login. The amount of information in the email will vary according to the service but often contains a minimum of the time and device details. Whilst this alert can be useful for monitoring access to an account, it does not provide any indication of further activity other than the validation of the credentials. Figure 2.4 displays a screenshot of this type of email.

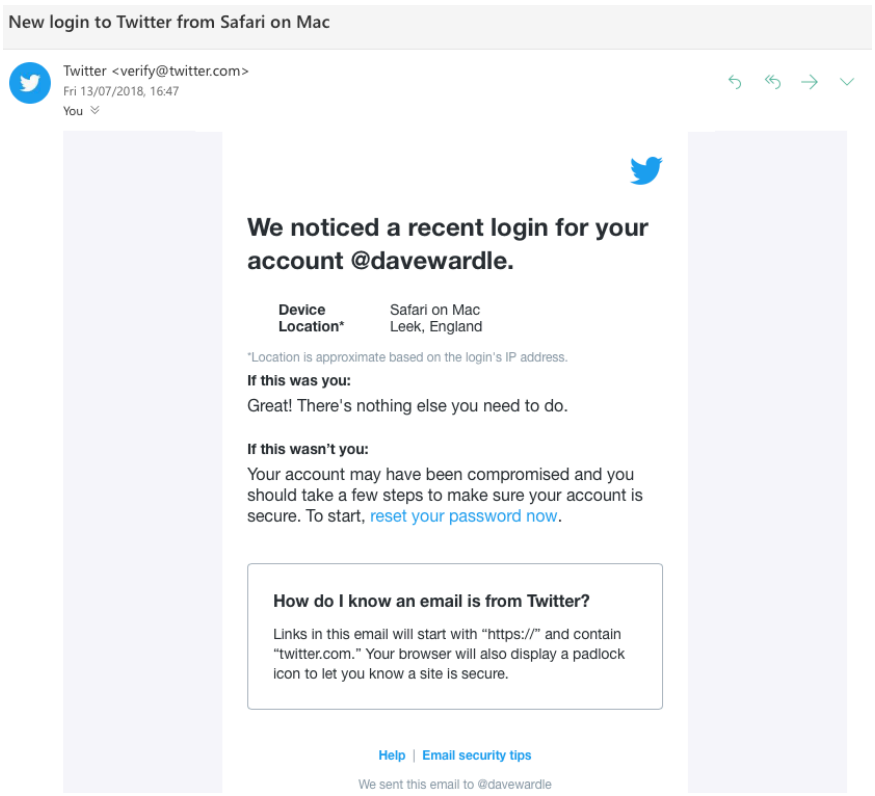


Figure 2.4: An example of a new login notification email from Twitter

Login history/recent activity

Similarly, a web service will often have a special page that allows the user to audit recent logins and active sessions. Initial tests showed that the information provided was more detailed than that in a new login email. For example, Dropbox displayed the IP address of the user on this page but did not include it in the email. However, like the notification emails, the amount of information varied between services. One major downside of using these pages for monitoring is that one would need to log in on a regular basis, or after a separate

alert, and manually check for unauthorised activity. An example of the login history on a Gmail account is shown in Figure 2.5 and further examples can be found in Appendix A.

Recent activity:

Access Type [?] (Browser, mobile, POP3, etc.)	**Location (IP address)** [?]	**Date/Time** (Displayed in your time zone)
Browser (Chrome) Hide details *"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36,gzip(gfe),gzip(gfe)"*	* United Kingdom (217.155.33.122)	16:54 (0 minutes ago)
IMAP () Show details	United Kingdom (217.155.33.122)	16:54 (0 minutes ago)
Authorised Application (450232826690-0rm6bs9d2fps9tifvk2oodh3tasd7vl7.apps.googleusercontent.com) Hide details *"name: iPhone Mail"* *"os: iOS"* *"os-version: 11.4 (15F79)"* *"version: 15F79"* OAuth Domain Name: *450232826690-0rm6bs9d2fps9tifvk2oodh3tasd7vl7.apps.googleusercontent.com* Manage Account Access	United Kingdom (94.197.120.124)	16:42 (12 minutes ago)
Authorised Application (450232826690-0rm6bs9d2fps9tifvk2oodh3tasd7vl7.apps.googleusercontent.com) Show details	United Kingdom (217.155.33.122)	11 Jul (21 hours ago)
Authorised Application (450232826690-0rm6bs9d2fps9tifvk2oodh3tasd7vl7.apps.googleusercontent.com) Show details	United Kingdom (217.155.33.122)	10 Jul (2 days ago)
IMAP () Show details	Denmark (109.238.48.132)	10 Jul (2 days ago)
IMAP () Show details	Denmark (109.238.48.151)	10 Jul (2 days ago)
Authorised Application (450232826690-0rm6bs9d2fps9tifvk2oodh3tasd7vl7.apps.googleusercontent.com) Show details	United Kingdom (217.155.33.122)	9 Jul (3 days ago)
SMTP	United Kingdom (217.155.33.122)	9 Jul (3 days ago)
SMTP	United Kingdom (217.155.33.122)	9 Jul (3 days ago)

Figure 2.5: A screenshot showing recent login activity on a Gmail account

Credential check

A study looking at the effect of leaked credentials for Gmail accounts identified account hijacking as a common behaviour of intruders. They describe this as the process of changing the account's password to lock the legitimate owner out of the account [23].

If an account is hijacked, it will prevent the use of some monitoring techniques discussed in this section and others could be disabled by the intruder. Even so, it is in itself a sign of unauthorised access. Sometimes the service provider will email the account owner whenever the password is changed however this is not always the case. Furthermore, this email cannot be relied upon as the email account may have also been compromised by the same intruder. A better method would be to check that the credentials are still valid on a regular basis.

A big drawback of this technique is that it would provide no information about the intruder and *only* then if they change the password. For this reason, it should only be used as a last resort when it is not possible to use any other monitoring technique.

Account activity

Due to the nature of a honeypot account, any activity within it can be classified as unauthorised. Messaging services such as email and Twitter, have a natural log of activity in the form of sent/received messages. Financial services will store a record of all transactions. Other services, such as media streaming and e-commerce, keep a log of activity to improve their service and personalise recommendations. Whilst account activity may not provide IP/location information, it can provide a crucial insight into the motives of the cybercriminals.

In addition to "Hijackers", Onaolapo et al devised three other categories of attackers: Curious, Gold diggers and Spammers [23]. By analysing account activity, it may be possible to expand on these categories and even propose new ones for different types of services. It could well be the fact that different types of services may have different trends. For example, one would expect that on a paid-for streaming service, such as Netflix, an intruder would hope that their activity went unnoticed rather than hijacking the account. Figure 2.6 displays an example of activity history on this service.

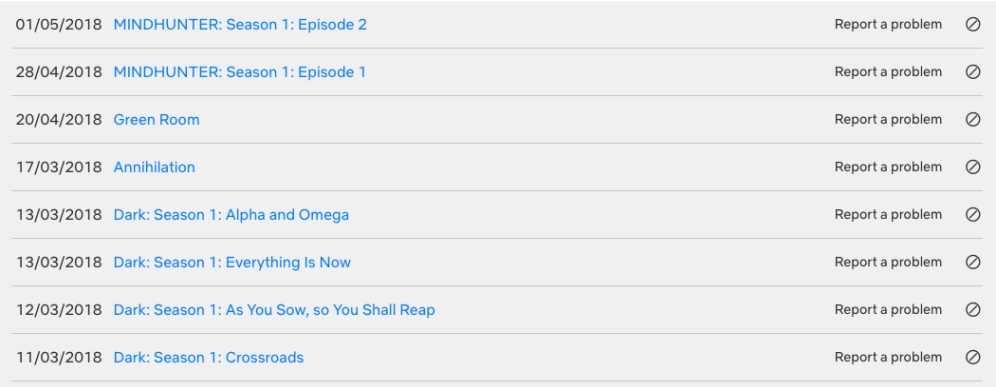


Figure 2.6: A screenshot showing recent activity on Netflix

An aspect that is unique to a compromised email account is the possibility to gain access to other services without knowledge of a valid password by resetting the password. Not only would the presence of such an email indicate access to that account, which could be considered a separate way of monitoring access, it also presents a clear picture of the intruder's intentions.

Data request

Under the General Data Privacy Regulation (GDPR), individuals have the right to request access to all data that a business may have relating to that individual. That data could include useful information relating to logins and activity. However, since the fake identities are not real entities, there would be ethical concerns in making a request of this nature. For that reason, a manual data request was not considered for the purposes of this project.

In some cases, a service provider may offer a "data download" functionality that allows a user to download a copy of all their personal data stored by that service. Given that this is a feature of the website rather than the use of an individual's legal rights, there would be no ethical concerns to using it. However, this would be a reactive measure taken to gain further information of a breach and could not be used as the sole technique.

Honeytokens

The concept of honeytokens was introduced in *Chapter 2.2*. Honeytokens can be used as a monitoring technique as long as a service has the functionality to store data that can only be viewed by the logged-in user. This is a big advantage above previously mentioned techniques as it does not rely on the infrastructure provided by the service. Furthermore, any honeytoken would remain active even if an account is hijacked.

A basic honeytoken could be the credentials for another monitored account hidden within messages. Access to this second honeypot account would signify that the honeytoken, assuming the credentials cannot be obtained elsewhere, had been used and in turn that the honey identity's account had been accessed.

The main shortcoming of this approach is that it requires the intruder to "trigger" the honeytoken (i.e. make use of it). If they do not find it, or have no interest in it, then the access to the account could go unnoticed.

Custom scripts

It may be possible to develop custom scripts that can run on the web services to monitor any activity. In the aforementioned study focussed on Gmail accounts [23], the researchers created a tool that would track emails being viewed, starred or sent. The script, embedded in a hidden Spreadsheet, marked all emails as unread and unstarred when it initialised. It would then run once an hour and look for any emails that were not in this baseline state.

However, the researchers noted that whilst quite powerful, Google App Script is fairly limited and cannot provide location information.

During the investigation phase of this project, it was rare to find a service that allowed custom scripts to be executed. It is often an unnecessary feature and would present a clear security risk to the service provider.

Partnership

One final option would be to form a partnership with a service. In fact, this may be the only option when it is simply not feasible to create a honeypot account, for example an online bank account. This would lead to several possibilities such as:

- Bulk creation of honeypot accounts
- Additional monitoring techniques
- More detailed logging information

Prior to the commencement of the project, several prominent websites and online financial institutions were contacted. Although several expressed interest in the project, none were willing to form a partnership.

Nine different methods for monitoring access to an account have been presented in this section. It is often the case that a website will only offer one or two of these options. Any of the techniques could be used as the basis of a honeypot account. Nonetheless, a minimum of login notification email or the presence of a login history page was required for the honeypot accounts created in this project. The full criteria for selecting suitable websites, along with the chosen services, are discussed in the next chapter.

2.4 Illegal sharing of personal data

The final topic area to introduce is the illegal sharing of stolen personal data including credentials. Over the past decade, the releases of usernames and passwords (*credential dumps*) have become a popular shared commodity, especially within underground communities. Cybercriminals share credential dumps in order to prove technical capability, enhance their reputation, and demonstrate legitimacy with criminal groups. They use three

main techniques to collect these credentials: phishing, malware, and database compromises [24].

The "darknet" refers to the anonymous communication provided by crypto-networks such as "The Onion Router (Tor)". Tor is free software that effectively allows for anonymised browsing and thus protects the user's privacy. This is in contrast to the "deepnet" which refers to portions of the open Internet (the "Clearnet") that are not indexed by search engines [25]. The darknet has become widely used for illegal activity leading to it being discussed in many news stories. However, Nunes et al [25] state that there is actually little hacking activity on the darknet. On darknet markets, only a small fraction of products (13%) are related to malicious hacking.

Butler et al [24] state that as stolen credentials are published, they are slowly distributed across the Internet, usually to hacking forums before appearing on public paste sites. Troy Hunt [26], who operates a data breach notification website, agrees that few data breaches are released on the darknet and any that are will eventually end up on the Clearnet.

Forums are user-centric platforms with the sole purpose of enabling discussion with like-minded individuals. Underground forums (also known as blackhat, hacker, or carding forums) are forums that focus on illegal activity and operate on both the darknet and Clearnet. Shakarian et al [27] observed that the majority of English-language forums are only accessible through the Tor-network whereas forums for Russian speakers are more often hosted on the surface layer Internet. Access to these forums can vary heavily – some allow anyone to register whereas others will require an invitation code, payment, or even an interview. English speaking forums contain feature boards concerned with financial fraud, hacking, information security, and the release of credential dumps and personal data.

In the first longitudinal measurement study of the underground market for stolen credentials, Thomas et al [28] assess the risk it poses to millions of users. The authors looked into forums that trade credentials exposed via data breaches along with phishing kits and keyloggers. They developed an automated framework to monitor blackmarket actors and stolen credentials. Over the period of a year from March 2016, they managed to identify over 1.9 billion usernames and passwords exposed by data breaches however they emphasise that this is just sample of underground activity.

Shulman [29] discovered that stolen webmail accounts were more valuable than those for other services. This is because they may allow further compromises through password recovery features. Of these, Gmail accounts were the most expensive, fetching up to $80 in comparison to $1.50 for a Hotmail account. Whilst those figures may have changed since 2010, the high value of webmail credentials are echoed by DeBlasio et al [30].

Thomas et al [28] created a framework to identify and source credential leaks. A breakdown of their results can be seen in Table 2.4. Most public leaks are small with 48% containing fewer than 1,000 credentials.

Source	Candidate documents	Confirmed leaks	Credentials extracted
Paste sites	3,317	1,666	4,855,780
Search index	26,208	1,304	10,856,227
Public forums	1,921	557	107,343,690
Private forums	-	258	1,799,553,568

Table 2.4: Breakdown of source of credential leaks [28]

Malderle et al refer to a service or storage location where credential dumps are traded as a *data sink*. They state that a data sink can be public, semi-public or closed for a special group of users. Paste sites are an example of a public data sink and used by cybercriminals to either distribute their stolen data leaks or advertise them by sharing a sample of the leak to prove their value [31]. These sites are services that allow users to store and share plain text (such as snippets of code). Most allow users to post anonymously which has given rise to the sharing of credential dumps. There are some common patterns of pastes that may appear on these kinds of sites [32].

Database dumps normally take the form of scripts that can be used to restore the entire database structure. They will often contain the password which may be secured using a cryptographic hash. An example is displayed in Figure 2.7.

```
(`id`, `team_id`, `email`, `name`, `password`, `league`, `active`, `regdate`, `lan`, `lastlogin`, `birthdate`, `favclub`,
`favmanager`, `description`, `pers_email`, `mess_id`, `iso`)

(14, 568, 'vcpd_@hotmail.com', 'Flavio00', '059b4db7cdb1cbddc3f0e5d95c881597', 1, 1, 1224313200, 0, 0, 0, '', '', '', '', '',
''),
(4, 1, 'levi@medeeaweb.com', 'Slash', 'c57aeddaffce62fead6be61022eb1340', 1, 1, 1224313200, 0, 1235380637, 483260400, 'FC
Juventus Torino', 'Carlo Ancelotti', 'I''m the admin of this site :D', 'slash@manager-arena.com', 'slashwebdesign', ''),
(96, 241, 'bobbydick1983@yahoo.com', 'bobbydick1983', '48238b7f2aa5f76a1d1e119f8942ebe7', 2, 1, 1224491297, 0, 0, 0, '', '', '',
'', '', ''),
(68, 77, 'azrants@yahoo.com', 'billyboy', 'bee783ee2974595487357e195ef38ca2', 1, 1, 1224313200, 0, 0, 0, '', '', '', '', '', ''),
(16, 21, 'valdas.jancauskas@gmail.com', 'webuxxx', '1aa87e76902e6df9042d17a642d04181', 1, 1, 1224313200, 0, 1234686482, 0, '',
'', '', '', '', ''),
```

Figure 2.7: Example of a database dump paste [32]

Email and password pairs are simple lists containing credentials consisting of a username (or email address) along with a plain text password. Figure 2.8 shows an example of email and password pairs.

```
mekim45@hotmail.com:mullins74
jjenkins19@yahoo.com:faster
r_m_vincent@yahoo.com:soling12
deadally@hotmail.com:balajaga1
Choas23@gmail.com:8KlAs432
jefftrey@yahoo.com:tanqueray
syntex05@mac.com:kobela87
michaelbarbra@hotmail.com:2177
mcginnis_98@yahoo.com:bentley
majikcityqban82@gmail.com:tinpen39
```

Figure 2.8: Example of an email and password pair paste [32]

Logs and code blocks take a variety of different forms and may be anything from compromised system logs to code.

```
array("/upload/iblock/ed0/--.jpg","vitaly.cherkasov@autohansa.ru"),
array("/upload/iblock/562/--.jpg","andrey.mastakov@autohansa.ru"),
array("/upload/iblock/ed2/---.jpg","sergey.smirnov@autohansa.ru"),
```

Figure 2.9: Example of a log paste [32]

The pastes could also be a seemingly random list of email addresses without any context. These could be completely innocent but may also indicate a serious data breach.

In the experiment phase of this project, paste websites were used to publish the credentials for various honeypot accounts belonging to the fake identities. The contents of the pastes were based on ones that can easily be found on those sites and are described in more detail in *Chapter 5*.

This page intentionally left blank

3 Design

This chapter describes the design of the honey identities and monitoring infrastructure using knowledge gained from the previous chapter. The implementation of a framework to create honey identities and the monitoring infrastructure is discussed in *Chapter 4*.

3.1 Honey identity

A new term was coined for this project: *honey identity*. This can be described as a collection of personal data used to create a number of accounts (or a digital footprint) whose value lies in being attacked or compromised. Therefore, a honey identity should have two key properties:

- A potential intruder should not be able to identify it as a honeypot and thus avoid it;
- Equally, it should entice any potential intruders.

To gain the first property, it is necessary to use realistic attributes to make up the identity. The attributes should be different for each honey identity to ensure that they cannot be identified if there is more than one in the same dataset. A simple way to achieve this would be to develop a database of possible values and then randomly select from it. Given the nature of randomness, some form of quality control may be required to ensure that the combination of attributes still looks "real", for example a character with the same first and last name may stand out.

The second property is more difficult to achieve. Indeed, future use of honey identities could help to identify contributing factors. As stated in *Chapter 2*, stolen credentials are traded on underground markets and forums. It was speculated that creating a digital footprint consisting of accounts for several high-value websites would make the honey identity "desirable" to cybercriminals. However, there is limited research available on the price of stolen credentials and the data that is available is often out of date.

The most recent figures available have been collated by Top10vpn.com. The website, who review VPNs and other Privacy tools, has created a "Dark Market Index"; showing the price of stolen credentials for many popular websites. The prices were calculated by taking an average of the advertised sale price for hacked accounts across three big darknet

marketplaces over the period of a week [33]. However, it did not consider the validity of the credentials. A shortlist of suitable websites was created by selecting those with the highest prices in each category.

Whilst the figures provided by Top10vpn.com were a suitable starting point, it would be inadequate to rely solely on their research, so it was decided to also consider other popular websites. The shortlist was supplemented with several English language websites that had a high Alexa Rank (`www.alexa.com`). Alexa is a web traffic analytics company which publishes a daily list of the top million websites. The final shortlist of 30 websites is displayed in Table 3.1.

The design for creating a new honey identity is presented in Figure 3.1. Another aspect that would aid both properties would have been for the honey identity to have an active online presence. Several social media networks were on the shortlist, but it would not have been feasible to regularly post new content due to the amount of time available for this project.

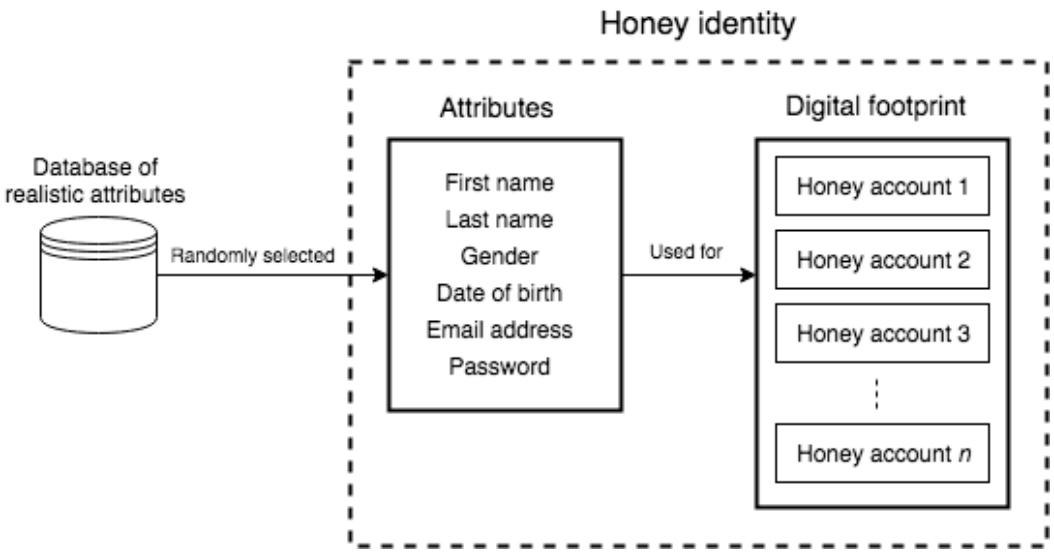


Figure 3.1: Design for the creation of a honey identity

Service	Domain	Category	Price (£) [33]	Rank[1]
Paypal	paypal.com	Finance	279.94	68
eBay	ebay.com	Shopping	26.2	37
Apple ID	apple.com	Entertainment	10.98	74
Amazon	amazon.com	Shopping	6.78	11
British Airways	britishairways.com	Travel	6.73	4531
Netflix	netflix.com	Entertainment	5.99	34
Spotify	spotify.com	Entertainment	5.69	123
Airbnb	airbnb.com	Travel	5.66	364
Uber	uber.com	Travel	5.02	1315
Facebook	facebook.com	Social network	3.74	3
Deliveroo	deliveroo.com	Food delivery	3.74	18821
Skype	skype.com	Communication	3	505
Groupon	groupon.com	Shopping	2.99	633
Outlook	live.com	Email	2.37	17
Booking.com	booking.com	Travel	2.25	101
Plenty of Fish	pof.com	Dating	2.24	1004
Match.com	match.com	Dating	2.24	1759
TripAdvisor	tripadvisor.com	Travel	1.87	204
LinkedIn	linkedin.com	Social network	1.49	52
Yahoo	yahoo.com	Email	1.2	6
Twitter	twitter.com	Social network	1.2	10
Instagram	instagram.com	Social network	0.92	14
Google	google.com	Email	0.75	1
YouTube	youtube.com	Entertainment	-	2
Reddit	reddit.com	Social network	-	16
GitHub	github.com	Software	-	73
Pinterest	pinterest.com	Social network	-	76
Adobe	adobe.com	Software	-	87
Soundcloud	soundcloud.com	Social network	-	110
Dropbox	dropbox.com	Software	-	132

Table 3.1: The shortlisted web services

[1] on July 2nd 2018: https://toplists.net.in.tum.de/archive/alexa/alexa-top1m-2009-07-02.csv.xz
In cases of multiple domains (e.g. .com and .co.uk), the rank of the highest placed domain was used.

3.2 Selecting web services for a digital footprint

The final websites were chosen from the shortlist based on the following criteria:

- Ethical and practicability constraints. It would be against ethical guidelines to break the terms and conditions of a service. Some services state that they must be informed if an account is compromised and many of them specifically forbid the creation of fake/false accounts altogether. There may also be certain other restrictions that mean the service would be impractical, such as requiring valid billing details.
- The availability of the monitoring techniques discussed in *Chapter 2*. At a minimum, login notification emails or a login history page should be present.
- The possibility to automate registration and certain monitoring techniques. In this case, the presence of a CAPTCHA was noted however the absence of one did not necessarily mean that it was possible to automate.

A summary of the findings can be seen in Table 3.2. Please note that monitoring techniques were not investigated for some services that forbid fake accounts. Furthermore, services with obvious practical constraints, such as banking, were not even considered.

3.2.1 Chosen web services

Eight of the shortlisted websites met the defined criteria, from which six were chosen to form the digital footprint. Yahoo and Pinterest were not selected as preference was given to similar websites. This section provides a brief introduction to each chosen service along with details of any reported data breaches and the available monitoring techniques.

Gmail (`www.google.com/gmail/`)

Google is one of the most popular email providers with over 1.4 billion accounts [34]. Whilst the project is focussed on the email service, a Google account offers a whole suite of applications including word processing, spreadsheets, video chat and social network. A Google account can also be used on YouTube, the video sharing platform.

Gmail was chosen over Yahoo email for this project as their platform offers the chance to test almost every monitoring technique mentioned in *Chapter 2* including the use of custom scripts. Whenever there is a login from a new location or device, an email is sent to the

Service	Allow fakes	2FA SMS	Login email	Login history	CAPTHCA free
Paypal	✗	-	-	-	-
eBay	✗	✓	✗	✗	✗
Apple ID [1]	✓	✓	✓	✗	✗
Amazon [1]	✓	✓	✗	✗	✓
British Airways	✗	-	-		-
Netflix [2]	✗	✗	✗	✓	-
Spotify	✗	✗	✗	✗	✗
Airbnb	✗	✓	✗	✓	-
Uber [2]	✗	-	-	-	-
Facebook	✗	✓	✓	✓	✗
Deliveroo [2]	✗	-	-	-	-
Skype [3]	N/A	N/A	N/A	N/A	N/A
Groupon	✗	-	-	-	-
Outlook	✗	✓	✓	✓	✓
Booking.com	✓	✓	✗	✗	✓
Plenty of Fish	✓	✗	✗	✗	✓
Match.com	✗	-	-	-	-
TripAdvisor	✗	-	-	-	-
LinkedIn	✗	✓	✗	✓	✓
Yahoo	✓	✓	✓	✓	✓
Twitter	✓	✗	✓	✓	✓
Instagram	✓	✓	✗	✓	✓
Google	✓	✓	✓	✓	✗
YouTube [3]	N/A	N/A	N/A	N/A	N/A
Reddit [4]	✓	✓	✗	✗	✓
GitHub	✓	✓	✓	✓	✓
Pinterest	✓	✗	✓	✓	✓
Adobe [1]	✓	✓	✗	✗	✓
Soundcloud	✓	✗	✗	✗	✗
Dropbox	✓	✓	✓	✓	✗

Table 3.2: Review of the shortlisted web services

[1]Terms state that the service provider must be immediately notified in the account is compromised

[2] Registration requires valid payment details

[3] Registration is for an account with parent service provide (Microsoft and Google, respectively)

[4] 2FA verification code is sent by email rather than SMS

user for further review. This email cannot be disabled, and more information about any login can be gathered from a recent activity page. There is an additional page showing a log of certain other types of activity involving other Google tools (e.g. search). Google also offers the ability to increase login security with 2FA through SMS, a smartphone app or a physical token (security key).

It is possible to easily monitor the email activity by configuring the account to forward all incoming messages to a separate email account. At the same time, outgoing email can be set up to send using a different SMTP server. This means that even if the messages are deleted then there will still be a record of them. It is harder to check whether emails have been viewed since it is possible to mark as unread. One possible option would be to hide honeytokens within select emails. If they are subsequently triggered, then this would indicate the message had been read. However, this relies on the intruder finding these emails and then using the tokens.

Unique to Google is the ability to use custom scripts for additional monitoring as mentioned in *2.3*. Google Apps Script is a scripting language based on JavaScript that can be used to automate and enhance functionality across the Google Apps suite. In combination with the login notification and history page, this can be used to provide further information about any account activity such as viewed emails [23].

Dropbox (`www.dropbox.com`)

Dropbox provides cloud storage and file sharing services to over 500 million users [35]. Dropbox offers a free basic package with paid for options available to increase storage size and extra functionality.

The company was the victim of a significant hack in 2012. It was initially reported that only email addresses were stolen. However, almost four years later, it was revealed to have been a lot more serious and included password hashes for over 68 million users. It was of interest that the hack is rumoured to have been due to the re-use of a password that had been leaked via a data breach from LinkedIn, a social network platform [36].

Dropbox will email the user whenever a there is a login from a new location or the account is synced to a new device. This notification is optional but enabled by default. Further information can be viewed on a "Security" page that shows current web sessions and synced

devices. There is also an option to enable 2FA using text messages or the mobile app. Whilst they do not provide a full log of user activity, one can plant honeytokens into uploaded documents and track new uploads.

Dropbox's paid-for plan allows for the batch creation of accounts and provides a full audit log. However, since the cost is per user, this approach would not scale well for a large experiment.

Twitter (`www.twitter.com`)

Founded in 2006, Twitter is a social networking platform with over 335 million active users worldwide [37].

Earlier this year, Twitter disclosed that there was a software bug that exposed plain text passwords. The company stated that there was no indication of a breach or misuse [38]. In 2013, Twitter acknowledged that hackers may have gained access to around 250,000 accounts including usernames, email addresses and password hashes [39].

As with Dropbox, an email is sent whenever there is a login from a new location by default. There is also an option to enable SMS 2FA. Whilst there is no page displaying login history[1], it is possible to obtain this information through an automated data request feature. It would be easy to track the account's activity by simply subscribing to their Twitter feed. Additionally, honeytokens can be embedded in private messages.

Apple ID (`appleid.apple.com`)

Apple ID is the single sign-on service used by Apple to authenticate devices, e-commerce and other services. Their terms and conditions state that any unauthorised activity must be immediately reported to Apple. Despite this, Apple ID was chosen due to its high value in Top10vpn.com's Dark Web Market Index [33]. 2FA was enabled on any account created thereby preventing full access to it. Any text message will indicate a breach, and despite the absence of location information, this could be used as the basis of a honeytoken.

There have been no reports of Apple ID having suffered a data breach. However, in 2014, over 100 celebrities had private photos stolen from the platform. The thefts were made possible through targeted phishing attacks rather than any breach of Apple's systems. In the

[1] There is a page showing logins from apps but this does not include web browsers.

subsequent aftermath, Apple strengthened the security of their systems and enabled certain security features by default [40].

Instagram (`www.instagram.com`)

Instagram is a photo and video sharing social network. It was launched in 2010 and bought by Facebook two years later for approximately $1 billion. It is reported to have over 800 million active users [41].

Last year, a bug allowed email addresses and contact details for millions of users to be revealed. This bug was then used to extract the personal data for thousands of celebrities which were subsequently listed for sale on a special website [42].

Like Twitter, it is possible to follow account activity through a public feed. There is a page displaying login history, however the information is extremely limited; showing just the time and date of login. The automated data download does not provide any further details on the logins. Instagram provides the option of 2FA through SMS only.

Despite the limited monitoring options, Instagram was chosen as it was a good proof-of-concept for automated registration and credential checking. Any credentials for this service could also be used as a honeytoken.

GitHub (`www.github.com`)

GitHub, also chosen as a good example of automation, is a web-based hosting service for software version control. It was recently announced that Microsoft had reached an agreement to acquire GitHub for $7.5 billion [43].

The company seems to take a proactive approach to security: monitoring user accounts, identifying new attack vectors and offering a bug bounty [44]. Whilst there have been no reports of a data breach for GitHub, it has been the source of several high-profile incidents. This was not due to any failings by the company but rather customers storing secret information in their code repositories. In the high-profile case of Uber, their GitHub account was hacked allowing the hackers to view AWS credentials contained within a private repository which, in turn, led to a large data breach [45].

There is a page displaying active sessions and a history of security events. It is also possible to enable 2FA through SMS or a smartphone app. Fake source code could be created containing honeytokens or as a means to leak credentials for other honey identities.

3.2.2 Private server

In addition to the above web services, it was decided to set up a private server for a website and email. With full control over the server, including source code and access logs, the generic monitoring techniques discussed previously were not as critical.

A website was developed for a fictional financial company called "ISG Project" (`www.isgproject.org`). The main purpose of the website was to provide a level of authenticity to the honey identities and to frame the leaking of credentials as the result of a hack on the website. Each honey identity would have a public profile page as well as their own account on the website.

An email server was also configured, and each honey identity was given an @isgproject.org email address. In addition to IMAP and SMTP, a webmail system was installed for easier access. A link to the webmail was added to the website.

The configuration of the server and a look at the risks associated are discussed in the next chapter.

3.2.3 Types of honey identity

Some of the services (Apple ID, Instagram and GitHub) were chosen despite their limitations as they offered a proof-of-concept monitoring technique. For this reason, it was not necessary to create accounts on these services for all of the generated honey identities. The different levels of honey identity are briefly discussed in this section.

Full and Partial

A honey identity that had online accounts with all six of the chosen web services was referred to as a **full honey identity**. A **partial honey identity** had (at least) accounts for the "ISG Project" website and email, and Dropbox.

Validation

One honey identity was needed to validate the framework and ensure that the monitoring

techniques were operating as expected. This identity's credentials would not be published but instead they would be used on a regular basis to trigger the various alerts.

Control

Another identity was required as a control measure; each account was created with a unique and strong password. Like the validation user, its credentials were not to be published. Any activity from this honey identity would indicate a flaw in the framework or that one of the services had separately been compromised.

3.3 Monitoring infrastructure

The various monitoring techniques used for each of the chosen web services have already been discussed, however only the 2FA SMS verification message and new login email would send an actual alert upon unauthorised access. It is important that any alerts were immediately viewed and acted upon to gain as much information about the intruder as possible, as displayed in Figure 3.2. Additionally, there is a risk that, if an email account is compromised, then any security-related emails could be deleted by the intruder. To mitigate this risk and make it quicker to check the email accounts there were two options:

- Use a single email client to fetch the email for every honey identity;
- Or more simply, configure each account to forward all new email to a single email address.

Furthermore, the credential checking and login activity page methods would be time-consuming if performed regularly and/or on a large scale. Tasks like these can be automated through the use of special software.

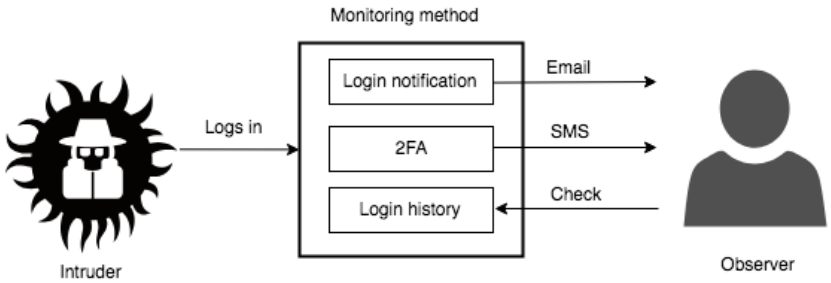


Figure 3.2: Basic design of login monitoring infrastructure

4 Implementation

This chapter presents the related work that has guided some of the implementation choices. It then goes on to describe the development of the honey identity framework and the monitoring infrastructure. It also details the architecture used during this phase of the project.

4.1 Related work

There were two projects, found during the literature search, that influenced decisions made during the implementation of the framework. These studies are summarised in this section.

4.1.1 What Happens After You Are Pwnd : Understanding The Use Of Leaked Account Credentials In The Wild [23]

In the research that most closely resembles this one, Onaolapo, Mariconti and Stringhini [23] conducted a study to gain an understanding of how cybercriminals make use of stolen webmail credentials. They designed a system to monitor the activity of Gmail accounts, manually created 100 accounts and published the credentials online using three different methods. Over the period of seven months, they logged and analysed the activity of their honey accounts. There was access to 90 of the honey accounts during the time period and they identified four types of activity. They categorised these activities as “Curious”, “Gold diggers”, “Spammers” and “Hijackers”. With the exception of credentials leaked via malware, the majority of initial access to the honey accounts was within two months.

Honey account setup

The researchers had to manually create all of their honeypot accounts. They commented that Google rate-limits the creation of new accounts from the same IP address by requesting phone verification and that this limited the number of accounts that they could create to 100. They used random combinations of popular first and last names when creating the accounts.

The accounts were populated with emails from a public dataset to give the impression that they were corporate users. To make the contents of the emails more believable, some aspects such as recipients, timestamps and company names were changed. The accounts were set up so that all outgoing email would be sent to a special “sinkhole” SMTP server, ensuring

that all outgoing email would go undelivered. This mitigated the risk of the accounts being used to send spam emails and subsequently being disabled by Google.

Leaking credentials

The credentials were leaked with the following split: 50 on paste websites (`pastebin.com`, `pastie.org`, `p.for-us.nl` and `paste.org.ru`); 30 on underground forums (`offensivecommunity.net`, `bestblackhatforums.eu`, `hackforums.net` and `blackhatworld.com`); and 20 using virtual machines infected with malware.

Monitoring infrastructure

In order to monitor activity on the honey accounts, they wrote basic software using Google Apps Script. The scripts would send notifications any time an email was opened, sent or "starred". Whilst powerful, they found that Google Apps Script did not provide enough information (such as IP addresses). They thus created external scripts that would log into each honey account and parse information from a recent activity page. In order to further similar research, they released the source code[1] for their system.

Differences

The main difference between this project and their research was the number of honey accounts for each identity. Onaolapo et al limited their research to a single email provider. By having more accounts, this provided the opportunity to investigate the effects of password re-use. As such, in this project it was necessary to develop a larger infrastructure to monitor all of the different web services. Whereas Onaolapo et al conducted their experiment over a longer time period and investigated other methods for leaking the credentials.

4.1.2 Tripwire: Inferring Internet Site Compromise [30]

DeBlasio et al [30] described a prototype system, called Tripwire, where they created special email accounts to use when they registered for a website account. Their idea was that any access to these email accounts could indicate a data breach from that website. In a year-long study, monitoring over 2,300 websites, they detected 19 compromises. Whilst these accounts

[1] `https://bitbucket.org/gianluca_students/gmail-honeypot`

were created to detect a breach rather than operate as a honeypot, there was a lot to be learnt from their methodology and framework for creating fake user profiles.

Identity creation

The researchers wanted to ensure that their accounts were not easily distinguishable from organic ones. All of their identities had full names, addresses, phone numbers, dates of birth, employers, etc. Most of the details were created using a third-party service[1]. The phone numbers were ones that were under their control and not used more than once on each particular website. They also used a methodical approach for creating usernames and passwords. The researchers conducted the experiment in partnership with an unnamed email provider, this allowed them to easily create thousands of email accounts with their generated credentials. Like the above project, all of the accounts were configured so that any outgoing email would go undelivered.

Automated registration

To automate account registration, they developed a web crawler using PhantomJS, a scriptable, headless web browser based on the WebKit engine. The crawler located registration forms, filled them out and submitted them. In order to avoid bot detection, they utilised a CAPTCHA solving service and a small network of web proxies. Finally, any incoming email was evaluated for verification links. The source code for the crawler has been published along with anonymised login data[2].

Detecting compromise

The email provider reported all successful login activity for their honey accounts on a regular basis throughout their study. The researchers used the assumption that any successful login would have been the result of an attacker having stolen credentials from the registered website.

Differences

There were several differences between this project and their research. Most importantly the goals of each were very different. The goal of Tripwire was to detect breaches in third party websites whereas the main objective of the framework in this project was to monitor access

[1] https://www.fakenamegenerator.com/

[2] https://github.com/ccied/tripwire

to honey accounts. Tripwire took great care in protecting the credentials and waited for a website to be hacked. They formed a partnership with an email provider who provided a report of any unauthorised access. This meant that they did not need to create a monitoring infrastructure. Finally, given the number of honey accounts that they created, they had to rely on automation a lot more than this project.

4.2 Creating the honey identity

The two main processes in the framework for creating a honey identity are:

- Generating the appropriate digital claims.
- Creating the digital footprint by registering this new identity on select web services.

An overview of the implemented framework is shown in Figure 4.1

4.2.1 Generating digital claims

As mentioned in *Chapter 2*, digital claims are sets of attributes that help to make up the digital identity. It would be nearly impossible, and also unnecessary, to create every attribute for the honey identity. It was sufficient to generate only the metadata that was needed for the chosen web services. Only the following were required by all of the registration forms:

- First name
- Last name
- Password
- Email address

Additionally, some web services asked for gender, date of birth and a profile photo after the account was created. It was decided to generate these as well to enhance the authenticity of the honey account.

A simple program was developed to produce all of these attributes. The various steps of this tool are described in this section. The source code can be seen in Appendix B along with all other code written for this project.

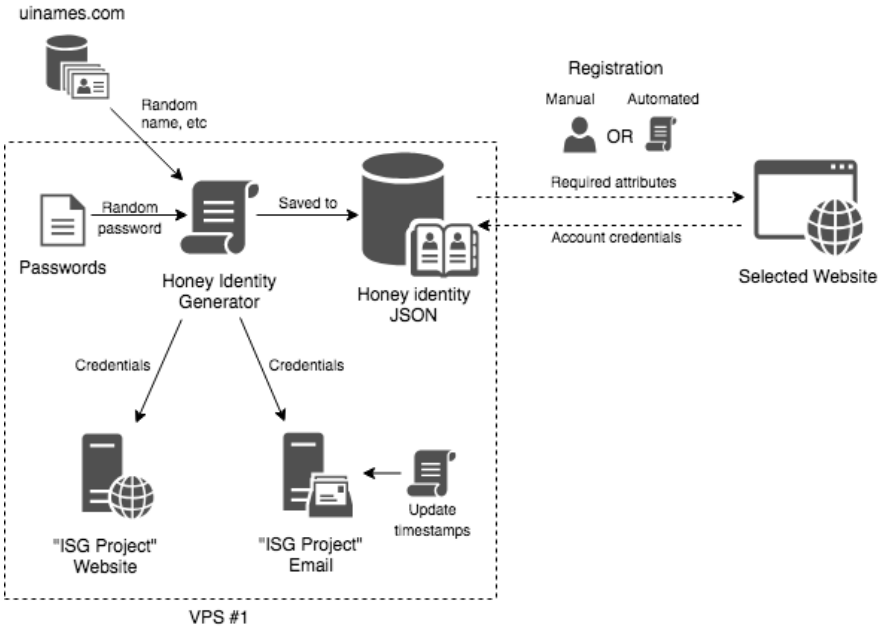


Figure 4.1: Overview of the implemented honey identity framework

First name and last name

Whilst it would be straightforward to create a database containing a set of popular names and randomly select from it, there are several third-party services available that could be used to save time. uinames.com was chosen as the API was easy to use and the quality of the data seemed to be most appropriate for the purposes of this project. One advantage that uinames.com had over similar tools was the ability to select a region. This meant that it was simple to generate names that seem "normal" in England. It also provided the opportunity to quickly change the country which would be advantageous for future experiments. In addition to a name, the API also returned the gender, date of birth and a profile photo.

After the attributes have been fetched from the uinames.com API, the program displays them to the user along with an acceptance prompt. If the name does not look authentic, or it was too similar to other names, then it can be re-generated before proceeding to the next step.

Password

The value of the honey identity lies in it being attacked or compromised, therefore, the password should not be a strong one. However, the password cannot be too weak as it risks being rejected during registration. A weak password may also look suspicious to any potential intruder.

A file was created containing passwords that, it was believed, would be sufficient for most websites' requirements, but any corresponding hash value would still be vulnerable to an offline dictionary attack. To make this file, a popular password list was first downloaded[1]. This list consisted of the 10,000 most common passwords found in a collection of credential dumps. The top 1,000 passwords were discarded and only those that were exactly nine characters long were kept from the remaining as a study stated that it was the average password length [5]. The resulting file contained a list of 898 candidate passwords. The honey identity generator loaded this file and randomly selected one of the passwords.

Email

Every honey identity required an email account for registration on most websites, the exception being those that provided an email service. The creation of a Gmail account could have been the first step in the next stage however it was decided to use the "ISG Project" email as the primary email. There were three big advantages for this approach: the accounts could be created programmatically; more information could be gathered about any unauthorised activity (by analysing log files); and, collectively, it provided a level of credibility to the credential leaks.

The program created an email account using the data from the previous steps. The username was a combination of the first and last name. To give the impression that the email account was an active one, it was populated with the same public dataset[2] that Onaolapo et al used in their study [23]. Likewise, the content of the emails was modified to replace names, dates and locations. An additional script was used to update the timestamps of the emails every day so that they always had a recent date.

As well as the email, an account was created on the "ISG Project" website for the honey identity. A profile page was manually created for it in the next stage.

Output

Once the honey identity had been created, the various attributes were saved to a JSON file to aid the next phase.

[1] https://github.com/danielmiessler/SecLists/blob/master/Passwords/darkweb2017-top10000.txt
[2] https://www.cs.cmu.edu/~./enron/

4.2.2 Creating the digital footprint

After the honey identity was generated, its details could be used for the registration of several websites and thus build a small digital footprint. The project's aim was to automate as much of this stage as possible. The Tripwire web crawler was not used as it was based on deprecated tools and, in fact, the authors posted a message discouraging use on the source code repository. Therefore, a modern suite of software tools for browser automation, Selenium[1], was used as the basis of a simple tool.

The tool was coded with a generic approach in mind but did not use heuristics to automatically determine the various registration fields. Instead it loaded two JSON files containing the honey identity attributes from the previous stage and pre-defined website configuration. The latter file was used to map the correct metadata to its respective field on the registration form. The program would then load a web browser in the background, visit the page containing the registration form, complete and submit this form, and report if there were any errors. A screenshot was taken of the final view so that a successful registration could easily be confirmed. The source code for this program can be seen in Appendix B along with the configuration files for Instagram and GitHub. The script did not try to resolve any errors that could be encountered when trying to register (e.g. unavailable username or invalid password) however this could be added in the future.

Unfortunately, most of the selected websites used anti-bot technology such as Google's ReCAPTHA to prevent automated registration. Whilst there are tools[2] and methods available to bypass these measures, this was outside the scope of the project. Therefore, it was necessary to manually create accounts for our honey identities on Gmail, Dropbox, Apple ID and Twitter.

Finally, many websites require the email address used during registration to be validated. They do this by sending an email containing a unique URL to visit. DeBlasio et al created a tool that would automate this process but noted that it proved problematic in practice [30]. In this project, all emails were forwarded to a single mailbox and so it was easy to manually verify the registrations.

[1] https://www.seleniumhq.org/

[2] http://www.deathbycaptcha.com/ or https://de-captcher.com/

4.3 Monitoring infrastructure

The various monitoring techniques used for each of the chosen websites was discussed in *Chapter 3*. This section describes further configuration and tools that were used to enhance the monitoring infrastructure. Figure 4.2 shows how the various services interact within the infrastructure.

4.3.1 Email

All "ISG Project" and Gmail email accounts were configured so that incoming email was automatically forwarded to the same bespoke email account. This email account was created solely for the purpose of receiving these emails and is referred to as the "alert" address. This meant that any login notification email could be viewed without the need to regularly check the individual accounts.

Additionally, email accounts were set up so that all outgoing mail was sent to a Mailtrap mailbox rather than the designated recipient. This, in turn, could be configured to forward all mail to a separate email address. The combination of both rules permits one to easily monitor all email activity.

4.3.2 2FA – SMS Verification

It was noticed that the SMS messages for 2FA did not contain any account identifier. Furthermore, some of the services would not allow the same mobile phone to be used on multiple accounts. For these reasons, it would be necessary for each honey identity to have their own phone number. On a small scale, it may be feasible to purchase a low-cost phone on a pay as you go contract however this would quickly become expensive and difficult to manage.

Twilio is a service that allows users to programmatically send and receive text messages. For a low monthly cost, it is possible to create phone numbers and, with a simple webhook, set them up to forward any received messages to the alert email. Unfortunately, during initial testing, the messages from several websites were not received. It could be that there was a fault on the service side but more likely the number was identified as not being "real". This meant that this technique was only available for Dropbox and Apple ID.

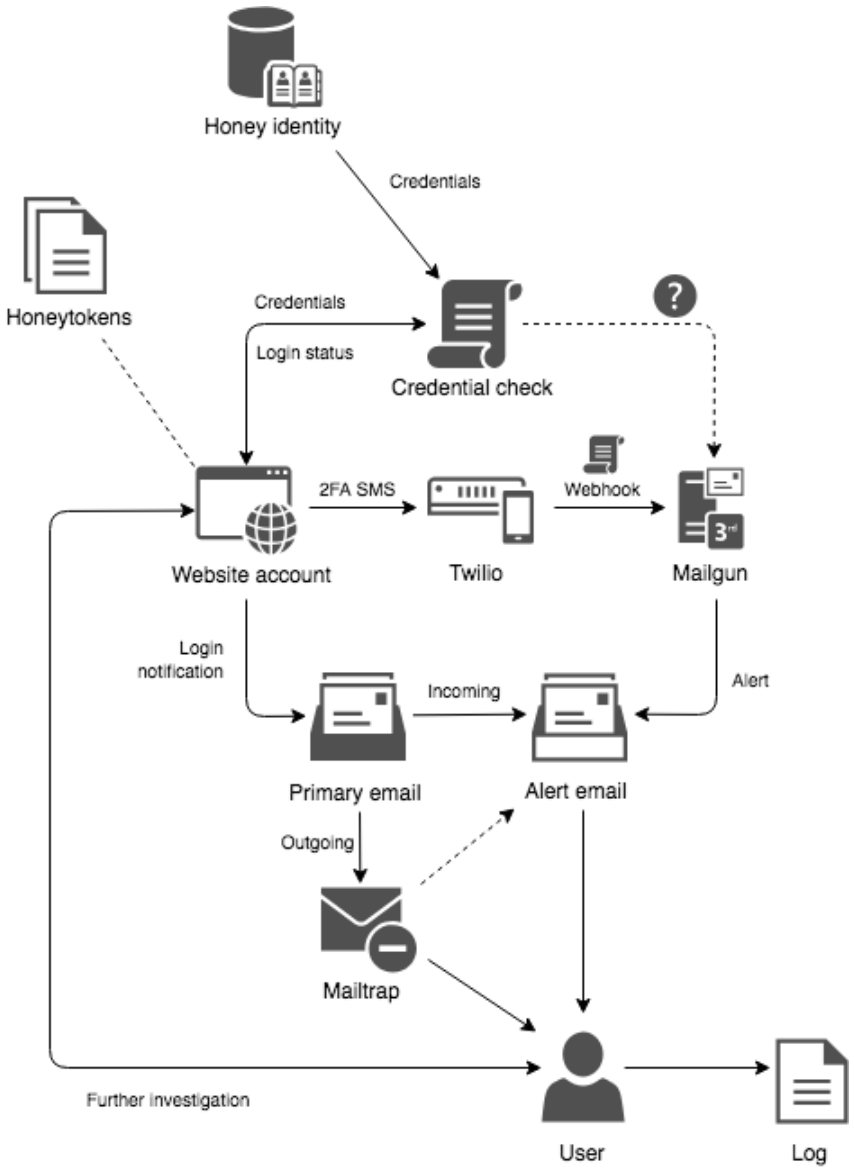


Figure 4.2: Overview of the monitoring infrastructure for new logins

4.3.3 Automation

In order to gain further information about the attacker's location, it would often be necessary to view the login/activity history pages whenever there was a new login. For a large-scale experiment, this could be time-consuming and would require the observer to react instantly. A better approach would be the one taken by Onaolapo et al [23]; develop a script to periodically login into the account the scrape and parse the information on these pages. Such a script was not developed given the size of the experiment in this project, but it would be necessary for any larger ones.

The script proposed above would perform a check of the credentials whenever it logged into the website. This could easily be adapted or separated so that it sent an email if the login failed. A prototype program was developed to perform this "heartbeat" functionality on Instagram; the source code for which is in Appendix B.

4.3.4 Honeytokens

Two types of honeytokens were designed for the use in the experiment. The first was simply the login credentials for a web service account that would trigger a notification, for example the 2FA verification SMS from Apple. The second was a PHP script hosted on the web server that would send an email whenever it was loaded before redirecting the user to an error page. The server was then configured so that any URL ending with .pdf or .doc would load the PHP script. This meant that it was easy to create new URLs that could easily be used to identify the correct honeytoken (e.g. `https://isgproject.org/this-is-in-the-report.pdf`).

Another possibility that was strongly considered, but ultimately discounted for practicability issues, was the use of credit card numbers. Through the use of virtual or pre-paid credit cards, it may be possible to safely hide the card details within a document or email. Using the app-based bank, Monzo (`www.monzo.com`), as an example, the user receives an in-app notification any time the card details are used. Crucially, the notification is still sent even if the transaction fails due to lack of funds. However, the account would require real details and so this approach could have a negative impact on the holder's credit rating.

Canarytokens (`www.canarytokens.org`)

Canarytokens is a tool, created by Thinkst, that simplifies the process of generating honeytokens as well as embedding them in files. The tool offers several different formats, relevant to this project are as follows:

- A simple web URL such as `http://canarytokens.com/szqma2v42usrxvqsc18vkm6zy/admin.php`;
- An image that can be embedded on a webpage or in an email;
- A Microsoft Word document;
- An Acrobat Reader PDF document;
- A QR code;

- Secret keys for Amazon Web Services; and
- A special URL that redirects a different (real) web page. During the redirect, JavaScript is used to probe the browser and gather more information about the visitor.

When creating a Canarytoken, one can set an email address for notification and a note to that can be used to identify the token. A token is triggered by visiting the URL (or opening a document which in turn requests the URL) and an email is then sent containing a minimum of the IP address and browser User Agent. Whilst similar basic functionality was created in just a few lines of PHP, Canarytokens has additional advantages in that it stores a history of all trigger events that can be viewed from a web console.

Another benefit of using Canarytokens over regular honey tokens is that some of the formats can trigger without any further interaction. However, if the attacker has a firewall enabled to monitor their outgoing traffic then this trigger could easily be prevented. Furthermore, in initial testing, it was noted that a warning was displayed when opening a Canarytoken contained within an Adobe PDF document. Likewise, a token embedded in an Adobe PDF or Microsoft Word document would not trigger unless the document was opened using Acrobat or Word.

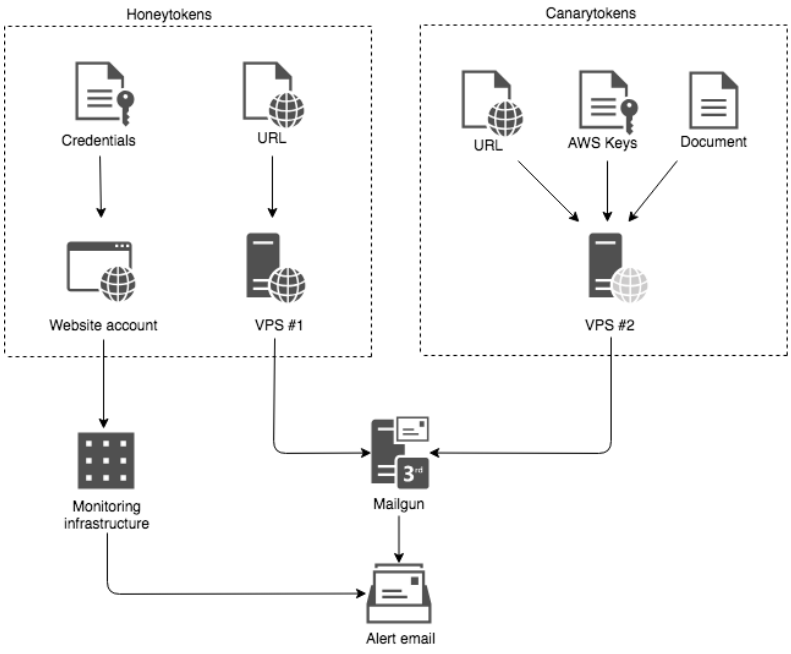


Figure 4.3: The honeytoken architecture

Figure 4.3 shows the architecture for the honeytokens and Canarytokens. The two formats were used equally where most appropriate. They were inserted into the contents of email, private messages and documents.

4.3.5 Server

Having complete control over the server hosting the website and email allows for a greater depth of monitoring tools and access to the log files.

On the "ISG Project" website, a security plugin[1] was installed. This plugin provided a full list of successful and failed logins along with other audit tools. This was enhanced using a hook to record the actual password used in cases of failed logins. A separate script was written to record all of the user's activity once they had logged into the website. In addition, Google Analytics and access logs were used to gain an insight into any impact that the publication of credentials may have had on the website.

The webmail was configured to log all user activity including failed logins. Like the website, this was enhanced to store the password used in unsuccessful logins. A plugin was also developed for the webmail system so that an alert email was sent whenever there was a successful login.

Since it was also possible to use the email account via POP3 and IMAP, a script was written to scan the log files for any successful logins. This script could be improved by keeping track of logins and sending an email whenever there was a new one. The source code for this script can be seen in Appendix B.

4.3.6 Recording activity

Finally, it was important to keep an accurate log of any unauthorised access and activity. In the event of a login notification, as much information as possible about the intruder was gathered. All of the details about the event were then stored in a spreadsheet.

[1] https://en-gb.wordpress.org/plugins/sucuri-scanner/

4.4 Architecture

Most of the software in this project was written using Python 3. This programming language was chosen as it is versatile and has a vast library of useful functions. PHP was used for the honeytokens, webhooks and WordPress plugins. All of the source code can be viewed in Appendix B.

In addition to a development machine, two virtual private servers (VPSs) were acquired for the period of the project[1]. The specifications for each VPS are shown in Table 4.1. The first VPS was used to host the website and email server. A script was used to speed up the set-up of the server[2]. This configured the email server and installed several open source tools such as webmail and an admin console. The nginx configuration was then modified to add an additional virtual host for the website. The website was powered using WordPress and a bespoke theme created for the fictional financial company, ISG Project.

Thinkst, the developers of Canarytokens, provide a free hosted solution however the generated URLs (`canarytokens.com`) give a clear indication as to their true nature. Fortunately, the source code is open source and a Docker configuration is available that makes it easy to set up the service on a private server[3]. Whilst this could have been achieved on the same VPS as the website and the email, it was decided that they should operate on separate servers. This meant that if there was any downtime, for whatever reason, on the web server, the Canarytokens were unaffected and vice versa.

	VPS #1	**VPS #2**
Purpose	Website and email	Canarytokens
OS	Ubuntu 16.04.4 x64	Ubuntu 16.04.4 x64
RAM	1 GB	1 GB
HDD	8 GB	25 GB
Host	Amazon Web Services	Digital Ocean

Table 4.1: VPS specifications

[1] To facilitate marking of this project, the servers will remain online until November 2018

[2] `https://docs.iredmail.org/install.iredmail.on.debian.ubuntu.html`

[3] `https://github.com/thinkst/canarytokens-docker`

A few third-party services were used as part of the framework:

- Mailgun (`www.mailgun.com`) is a transactional email service. This was used by Canarytokens and bespoke monitoring tools to send email alerts. For the project requirements, the free package was suitable, but all recipient email addresses needed to be previously verified.
- Mailtrap (`www.mailtrap.io`) is a tool designed to allow SMTP to be tested in a development environment without sending real emails. All honey identity email accounts were configured to send using Mailtrap to ensure that all outgoing emails would go undelivered and mitigate the risk of the accounts being used to send spam. Again, this is a free service, but a monthly payment would be required if there was a lot of outgoing email or additional features were required.
- Twilio (`www.twilio.com`) allows developers to programmatically send and receive text messages with phone numbers available for a number of countries. It was used to create additional mobile phone numbers that could be used to receive 2FA messages and forward these messages to an email account. Twilio costs $1/month for each phone number and $0.0075 for every received SMS. However, they offer a free trial that provides one free phone number and $20 credit.
- uinames.com (`www.uinames.com`) is a simple tool, aimed at UI designers, to generate fake names for use in their design mockups. This is a free service and was used to generate the names for the honey identities.

4.4.1 Risks

The presence of the two public facing servers invited a level of risk. A successful hack would not only impact the findings in this project but potentially endanger other individuals. Any server that is online is exposed to attack and the presence of leaked credentials may increase this (unwanted) attention. Best practice was followed in the configuration of the servers to minimise the risk of a successful breach including, but not limited to, auto updates, restricting traffic to HTTP(S) and email ports, and the use of intrusion prevention software.

5 The experiment

This chapter summarises the experiment that was conducted to test the implementation of the framework and the monitoring infrastructure. In total, seven different pastes were published on three separate occasions. Since the results from the initial pastes affected the design of the final two, the different stages and events of the experiment are discussed in chronological order. The dates of these, excluding any results, are displayed in Table 5.1.

Dates	Event
June 2018	Creation of the honey identities, pastes and other set up
July 3rd - 5th 2018	Publication of the first four pastes
July 3rd 2018	Start of the observation period
July 18th 2018	Publication of the fifth paste (purposefully delayed)
August 6th - 7th 2018	Creation and publication of two additional pastes
August 14th 2018	End of the observation period

Table 5.1: The timeline of the experiment

5.1 Setup

This section discusses the creation of the honey identities along with other necessary preparation. It also presents the various threats to the validity of the experiment that were recognised.

5.1.1 Honey identities

Initially, twelve honey identities were generated for use in this experiment. Due to time constraints and the manual process involved in registering for the majority of the chosen services, it was not feasible to create the full honey identity in all cases. As such, only one full honey identity was created as a proof-of-concept along with nine partial honey identities, one validation honey identity and one control honey identity. The attributes for all of these honey identities are displayed in Table 5.2.

#	Title	First name	Last name	Gender	Date of Birth	Password	Type
1	Ms	Emma	Fields	Female	06/06/1982	tacobell1	Full
2	Ms	Amelia	Coleman	Female	05/12/1981	love4life	Partial
3	Mr	Charles	Sutton	Male	20/02/1984	baseball2	Partial
4	Mr	Jamie	Baker	Male	28/01/1982	tacobell1	Partial
5	Mr	Brian	Peterson	Male	29/05/1986	mountain1	Partial
6	Ms	Gracie	Lewis	Female	19/12/1982	twilight1	Partial
7	Mrs	Charlotte	Baker	Female	15/10/1983	jeremiah1	Partial
8	Mr	Dylan	Watson	Male	26/10/1985	arsenal [1]	Partial
9	Mr	Alex	Baker	Male	12/01/1990	thomas123	Partial
10	Ms	Isabel	Griffiths	Female	13/03/1990	garfield1	Partial
11	Mr	Randy	Day	Male	02/04/1989	babyphat1	Validation
12	Mr	Luke	Scott	Male	01/05/1997	*<redacted>*	Control

Table 5.2: The initial honey identities created for the experiment

5.1.2 Further preparation

The honey identities were configured for monitoring as described in *Chapters 3* and *4*. All of the accounts had a Dropbox account. A shared folder was created using honey identity #1's account and all of the identities, with the exception of the control one, were given access to this shared folder. This folder contained a number of dummy files including two files containing Canarytokens. The Dropbox account for honey identity #1 was configured so that 2FA SMS verification was enabled. This was to prevent the contents of the shared folder from being edited.

Several of the partial honey identities had an account with one other web service with the aim that these logins could serve the purpose of honeytokens. These honeytokens were stored in documents on their Dropbox account. Additionally, an email containing a honeytoken or Canarytoken URL was sent to the primary email account of each honey identity.

Validation

A scheduled task was created to automatically send an email to the validation honey identity every day; this email should, in turn, be forwarded on to the alert email address. In addition,

[1] This password was weakened prior to its publication

once every week, a valid login was carried out on all of this identity's accounts and its honeytokens were triggered. A different Virtual Private Network (VPN) was used every time these checks were carried out to ensure that the login was from a new location.

Control

When registering for websites using the control honey identity, different strong passwords were used for each service. The passwords for the "ISG Project" website and email were manually changed since these were set by the honey identity generator program.

Breach notification

As an additional monitoring technique, the domain `isgproject.org` was registered on a breach notification website, *Have I Been Pwned?* (`www.haveibeenpwned.com`). This service would send an email whenever it discovered an email address belonging to the domain in a data breach, including those on paste websites.

5.1.3 Threats to validity

It was felt necessary to create a fictional company to provide the honey identities, and the publication of their credentials, a level of believability. However, it was acknowledged that a minimal amount of Open Source Intelligence (OSINT) would have revealed that the company did not exist. There was no listing of the company within Companies House, a WHOIS search divulged the domain name was less than a year old, the website source code showed that it actively discouraged search engines, and clearly the name of the company was a slight giveaway. This may have deterred some potential intruders from validating the leaked credentials.

It was also accepted that using the Enron dataset to populate the email accounts may have introduced bias into the results. Even with modifications, it would have been possible to locate the true source of the emails. However, there was not sufficient time to create a new email dataset and it would have only become apparent after access had been gained to the account.

On 29th June 2018, a week prior to the publication of the credentials, the website experienced an unsuccessful brute force attack. This is form of attack is a common occurrence on websites powered by WordPress and other content management systems [46]. Whilst this could be

an interesting topic for further research, it presented a threat to the project since most of the accounts had weak passwords. Access gained to the website through this method would not be relevant to the experiment. To reduce the number of these attacks, a WordPress plugin[1] was installed that would identify and block brute force attempts. Since the website was not indexed by any search engines, it is probable that the site was discovered by someone conducting a mass scan of the Internet. At that point in time, visiting the server's IP address in a web browser would have redirected the user to the domain name. The web server's configuration was changed so that any subsequent visitors to the IP address would see an error page instead.

5.2 Pastes

It was decided to publish the credentials for the honey identities using paste websites. Onaolapo et al had great success with leaking the credentials through these websites; 80% of all unique accesses were within 25 days and the majority were within a few days of the leak. In comparison, the figures drop to 60% and 40% for accounts leaked to underground forums and malware respectively [23].

Even so, the use of underground forums, malware and phishing were still considered as an additional means to leaking the credentials. However, the forums would require a level of interaction, and other social hurdles, to avoid suspicions; malware would require suitable samples and the use of multiple Virtual Machines; and phishing would also require sample emails for the appropriate websites. The amount of time required for each setup, along with the longer observation period necessary, would have been too much for the tight timescales set by the MSc project.

A preliminary trial of Paste websites was carried out by posting a Canarytoken, in the form of AWS keys, onto Pastebin (`www.pastebin.com`) without any explanation[2]. The Canarytoken was triggered after just two days and continued to be triggered on a regular

[1] `https://jetpack.com/support/security-features/#protect`

[2] `https://pastebin.com/ax50RTR4`

basis between December 2017 and May 2018[1]. This test reinforced the decision to use paste websites as the means for leaking the credentials.

5.2.1 Design

This section presents the different formats of paste used to leak the credentials. Several different types were used to test whether complexity of the format, placement of the credentials and the use of weaker password hashes would affect the use of the credentials. Each paste contained just one or two valid credentials. This meant that it was easy to link any event of unauthorised access to the specific source paste.

All of the pastes were based on ones that were easily found using Pastebin's search functionality and designed to frame a hack of `isgproject.org` as the source of the data breach. It was hoped that this would provide a level of authenticity to the pastes and also avoid the risk of falsely attributing the data breach to a service provider.

Plaintext passwords

Research into password dumps and leaked databases is a common, but controversial, practice amongst security professionals. Even though the datasets are publicly available, they often contain sensitive data and should be treated accordingly. It was necessary to investigate the structure of real password dumps in order to construct fake ones for the experiment. These pastes were easily found using the search functionality on the Pastebin website however no data from any of them was used. Figure 5.1 shows a partially redacted example of plaintext passwords that can be found using the search term "Spotify" on Pastebin.

However, it was decided to disregard the most common type of paste: an email and plaintext password pair. To construct an authentic looking paste, hundreds of pairs would be needed. It would be conceivable to generate random email addresses but, if realistic, this too could actually result in legitimate ones. Even though the password would be invalid, there is still a risk that these addresses could be spammed or targeted for phishing attacks. Furthermore, the presence of a plaintext password means that it would be easy for a potential intruder to

[1] `http://canarytokens.org/history?token=h77sepcrv158gju2pqktcx3i7&auth=8679dab6e54e3ccd6f415219eddca82f`

validate the data. Whilst a high failure rate may be common for these kinds of pastes, a single valid pair amongst lots of failures would be suspicious.

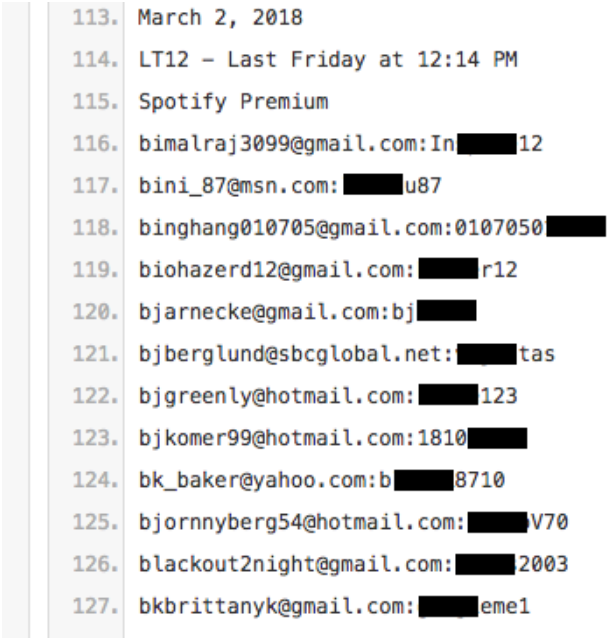


Figure 5.1: Email and password pairs easily found on Pastebin

It is important to explicitly state that none of the credentials found during the research were used in this experiment, nor were any validated.

5.2.2 Paste content

Six different pastes were initially created for the experiment. All of these files can be seen in full in Appendix C. The real pastes that formed the basis for each is displayed in Table 5.3.

Paste	Format	Example URL
1	Database dump	`https://pastebin.com/08e9jtPF`
2	Database query	`https://pastebin.com/w6Xufi7B`
3	SQL injection	`https://pastebin.com/8Hm99dKu`
4	Slexy	`https://slexy.org/view/s2uh5WjJnT`
5	Vulnerability scanner	`https://pastebin.com/j2zPXX2q`
6	Dropbox links	`https://pastebin.com/iBe2xhhS`

Table 5.3: Different formats of pastes

Paste 1

The content of this paste was based on a SQL (database) dump of the user table from the WordPress website. The real administrator account was removed along with the validation and control honey identities. It was further modified by randomly changing several characters in the password hashes for all but two of the users. Finally, to give the impression that the website was much older than in reality, the timestamps were changed.

WordPress uses a key stretching algorithm, Portable PHP password hashing (phpass)[1], to create the password hashes. Prior to publishing the paste, a password cracking tool called Hashcat[2] was used in an attempt to recover the passwords along with a popular wordlist. However, after several hours, it had failed to recover any of the correct passwords. Whilst this was not an accurate or thorough test, it was decided to further weaken one of the passwords to one that appeared in a top 207 probable password list[3]. Further experimentation suggested that this may have been unnecessary as it was discovered that a dedicated phpass cracking tool[4] could recover both of the passwords in under five minutes using the same wordlist.

Paste 2

The second paste took the form a screen scrape of a database query to show the contents of the WordPress user table. As well as the modifications made to paste one, the hashes were changed to the result of a different hashing technique, salted MD5. WordPress utilised this method until version 2.5 but still supports it for backwards compatibility.

Paste 3

This paste was designed to look like the results of an SQL injection attack on a bespoke website. In this case, all of the data was fabricated, and the passwords were displayed as simple MD5 hashes.

Paste 4

Paste 4 was identical to **paste 2** with the exception of the valid credentials. It was however published on a different paste website, Slexy (`www.slexy.org`).

[1] `http://www.openwall.com/phpass/`
[2] `https://hashcat.net/hashcat/`
[3] `https://github.com/danielmiessler/SecLists/blob/master/Passwords/probable-v2-top207.txt`
[4] `https://github.com/micahflee/phpass_crack`

Paste 5

The last two pastes were different from the previous ones in that they contained links to the credentials rather than the credentials themselves.

The content of paste 5 used a report from a vulnerability scanner, WP Scan[1], to display a link to a database backup of the website. The user table had been heavily modified as per paste 1 and the password hashes were changed to use the same hashing method described for paste 2. Only one valid login was contained within the dump. A script was written so an email would be sent if the file was downloaded. The publishing of this paste was delayed in order to see if it would be discovered naturally.

Paste 6

The final paste consisted of five links to Dropbox shared folders. Whilst four of the links were invalid, the third URL was for a "Private" folder stored within one of the honeypot identities. This folder was set up to contain several personal but not necessarily sensitive documents (e.g. photos) to give it the appearance of a real folder. Amongst the files was an Excel document containing the login details for the honeypot identity's "ISG Project" email and Gmail account – in this case, different passwords were set for each email account. Due to the potential ease of access, if discovered, the posting of this paste was also delayed.

5.2.3 Changes to honey identity email dataset

Four of the pastes contained a pair of valid credentials. This led to the possibility that a single intruder could access email accounts for both honey identities. This would immediately reveal the true nature of the accounts since the email dataset was identical for all of them. Whilst this would still be a record of unauthorised access, it is likely that the intruder would abandon their attack and it would not be possible to analyse any potential activity. Therefore, specific emails from each pair were removed so that each had completely different inboxes.

[1] https://wpscan.org/

5.3 Publishing the credentials

The first four pastes were published over the course of two days at the start of July 2018. They were posted at different times during the day to reduce the possibility of the same person finding all of the pastes. Paste 5 was not published for reasons that are discussed later in this Chapter. Paste 6 was published a fortnight after the initial pastes. The URLs of the pastes are displayed in Table 5.4 along with the time and date they were published.

Paste	Honey identities	URL	Date (2018)	Time
1	8, 9	`https://pastebin.com/RM6rxpFt` [1]	3rd July	17:23
2	2, 10	`https://pastebin.com/JR8aKJzQ`	4th July	09:38
3	5, 6	`https://pastebin.com/kqe9wxL5`	4th July	23:00
4	4, 7	`https://slexy.org/view/s20aJqpU4f`	5th July	14:30
5	3	N/A	N/A	N/A
6	1	`https://pastebin.com/Pi6qePvr`	18th July	11:30

Table 5.4: The published pastes

5.4 Initial observations

This section discusses the visibility of the different pastes along with any affect that their publication had on the website.

5.4.1 Paste views

The number of page views is displayed on Pastebin for all public pastes. By periodically checking the experiment's pastes, it was possible to keep track of the number of visits. Figure 5.2 shows the view counts after five minutes, one hour and one day. It was noted that after 24 hours, the number of subsequent views for each paste was negligible.

The results for pastes 1 and 2 were almost identical. Paste 6 received a similar number of views in the first five minutes however after that it continued to be viewed at a steady rate. This is likely due to it being easier to find in the search.

[1] The first paste was published with a setting to expire after one month and so this URL is no longer valid. All subsequent pastes did not have this setting to facilitate marking of this project.

Paste 3 received substantially more traffic. A security researcher's Twitter bot, @dumpmon, spotted the paste and posted the URL in a tweet[1] just three minutes after it had been published. An email from *Have I Been Pwned?* alerting to the presence of an `isgproject.org` email address was received at the same time. It's likely that other tools could have also noticed it.

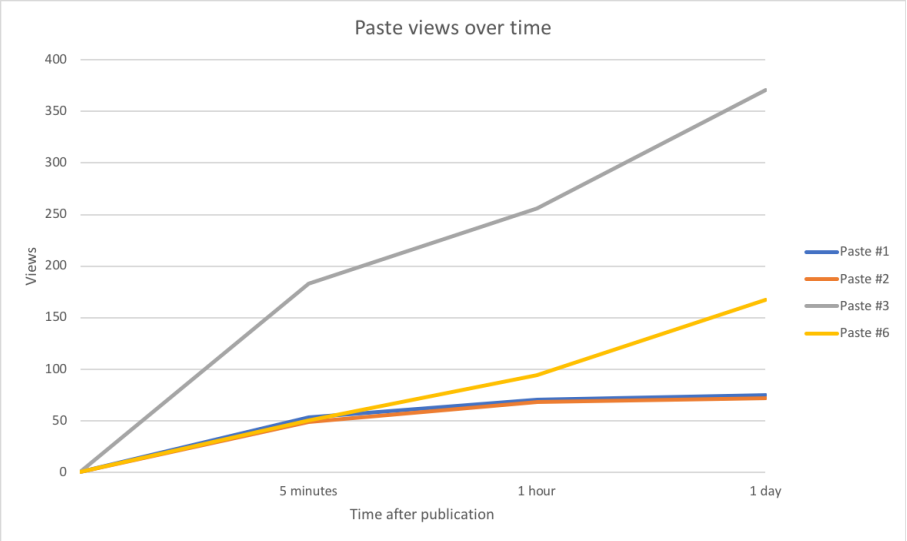


Figure 5.2: A chart showing the number of views for each paste

The visibility of the paste on search engines was also monitored. However, it should be noted that the status was slightly inconsistent; results would disappear from the index only to return the next day. Pastes 1 and 2 took three days to appear on the Google search index (which powers the Pastebin search functionality) however they could be found using DuckDuckGo's search (`www.duckduckgo.com`) on the same day. Conversely, paste 6 appeared in Google's index within 24 hours but took over three days to be indexed by DuckDuckGo.

Slexy does not display page views so it was not possible to monitor paste 4. It was spotted by another Twitter bot running similar code to the @dumpmon[2]. However, the page was not indexed by either Google or DuckDuckGo during the observation period.

[1] `https://twitter.com/dumpmon/status/1014630727592357892`

[2] `https://twitter.com/ecohostile/status/1014864220838510592`

5.4.2 Web server traffic

The website and webmail were monitored to see if the presence of the pastes resulted in an increase of activity. Both access logs and search engine visibility were taken into consideration however it should be noted that it is impossible to verify any traffic was related to the pastes.

Soon after the first paste was published, the domain started to appear on a variety of websites that display the DNS history for domains. It is possible that these sites parse pastes to look for valid domains, but it is more likely that they had simply discovered the website prior to the changes on 29th June 2018.

On 9th July 2018, the website became unresponsive due to excessive CPU usage. The server was rebooted, and the CPU usage returned to normal levels. Later that day, the website received a spike in traffic according to Google Analytics. Strangely all these visits were for the same 404 error page (`www.isgproject.org/isgproject.org/`) only and referred by `visitorjam.com` (which redirected to Google landing page). A further investigation of the access log revealed whilst these 404 visits had different IP addresses, they all had the same user agent. This would suggest that they originated from a web crawler or a bot. A WHOIS search revealed that `visitorjam.com` was owned by Pingl (`www.pingl.net`); a site that offers a service to spam analytics reports.

In the subsequent weeks, "legitimate" traffic to the website remained low according to Google Analytics, however, examination of the access logs revealed that the site was being scanned for vulnerabilities, and by security researchers, at least once a day.

5.5 Unauthorised access

A total of five events of unauthorised access were observed during the experiment. This section describes the first of these events, with the remainder discussed in *5.7*.

5.5.1 Intruder #1 – 18th July 2018 at 12:04

The first instance of unauthorised access occurred on 18th July 2018 at 12:04 using the credentials leaked via paste 6. This was only 34 minutes after the paste had been published.

The security event was alerted by a new login notification email from Google (Figure 5.3). This was immediately followed by an email containing a 2FA SMS verification code from Dropbox (Figure 5.4)

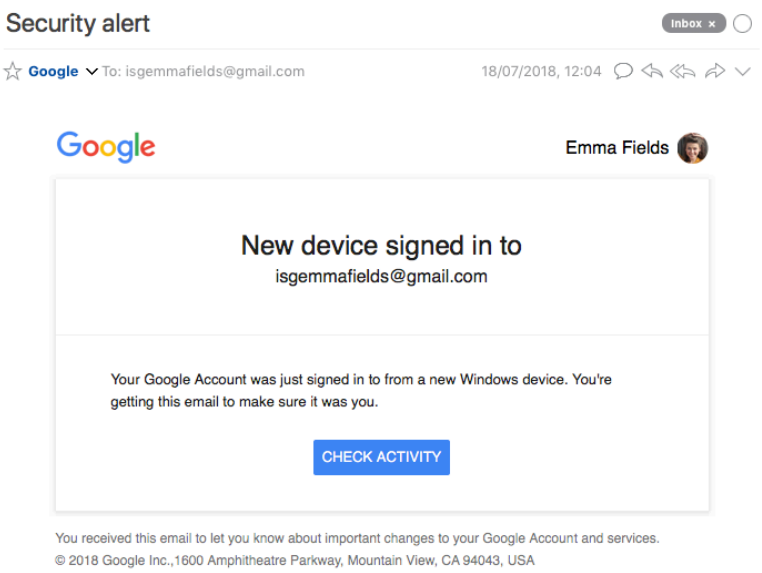


Figure 5.3: New login notification email from Google

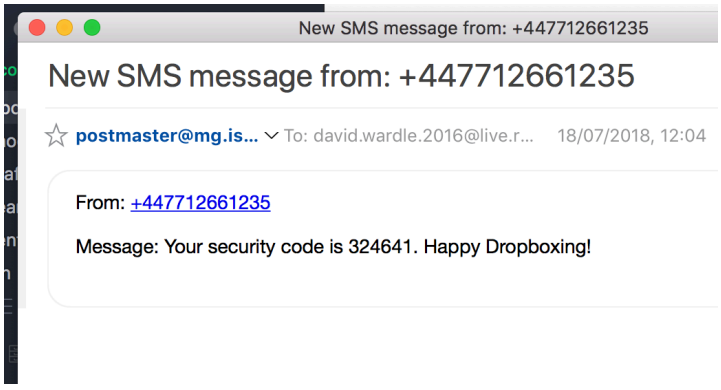


Figure 5.4: Email containing the 2FA SMS verification message from Dropbox

As a reminder, the source paste contained a link to a shared folder on Dropbox which included a spreadsheet displaying the passwords for both Gmail and "ISG Project" email. The latter password had been used for all of this honey identity's other accounts.

The Gmail inbox comprised of a single unread email containing a honeytoken URL. There was no evidence to suggest that the intruder viewed this email. The intruder[1] then tried to

[1] Since the intruder did not gain full access to the Dropbox account, the IP address was not recorded for the login. It is therefore feasible, but unlikely, that there were two separate people using the credentials at the exact same time.

sign into Dropbox but was not successful since 2FA SMS verification had been enabled on that account. Despite using the password for the "ISG Project" email to facilitate the Dropbox login, the intruder did not attempt to log into that email account (either using Webmail or IMAP/POP3). Nor did they log into any of the other accounts that made up this full honey identity.

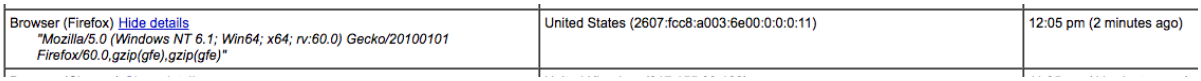


Figure 5.5: User agent and IP address for the first intruder

It was possible to gain more information about the intruder by logging into the Gmail account. The intruder used an IPv6 address (Figure 5.5) which related to Kentucky, USA (Figure 5.6).

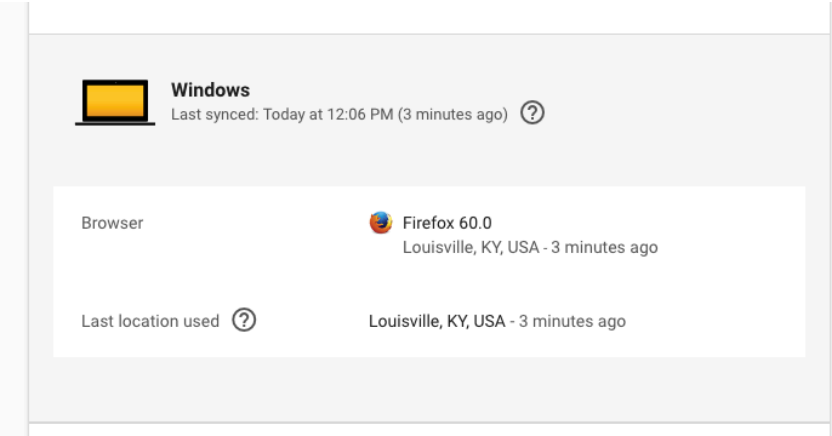


Figure 5.6: More user-friendly display of the same information

5.6 Additional leaks

The initial plan had been to publish paste 5 a few weeks after the initial leaks to see if the backup file was discovered organically. However, due to the complexity of this paste and limited events related to the others, it was thought that it would be unlikely to lead to any results, especially in a shortened time frame. For these reasons, it was decided against posting the paste. Instead two further pastes, containing plaintext passwords, were published.

5.6.1 Preparation

Another partial honey identity was created; the details of which were used to register to Gmail and Dropbox. Those accounts were prepared in the same manner as for all of the other honey identities except that the Gmail address was used when registering for Dropbox. The generated digital claims are shown in Table 5.5.

#	Title	First name	Last name	Gender	Date of Birth	Password	Type
13	Ms	Katie	Davies	Female	27/02/1983	Lacrosse2018	Partial

Table 5.5: The final honey identity

5.6.2 Pastes

Two new pastes were then created each containing a single pair of email address and plaintext password along with the title "dropbox.com". One paste contained the Gmail address belonging to the latest honey identity and the other displayed the "ISG Project" email address for a separate honey identity (initially included in paste 1 which had by then expired). The contents of these pastes are shown in Figures 5.7 and 5.8.

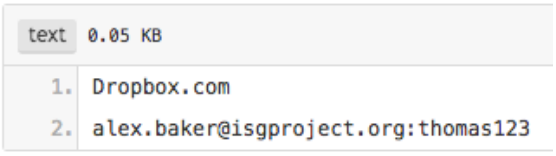


Figure 5.7: The contents of paste 7

Whilst a paste containing a single email and password pair had been initially discounted for this experiment, it was felt that the differences between the two may indicate if there was a preference to private and public email accounts.

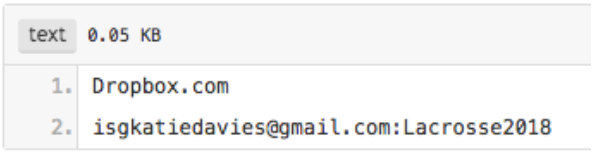


Figure 5.8: The contents of paste 8

5.6.3 Publication

The late publication of these credentials meant that the observation period could be no longer than two weeks, but it was hoped that the plaintext passwords would reap similar

results to paste 6. The two pastes were posted at a similar time on subsequent days and were set to expire after one week (Table 5.6). This presented the option of repeating the experiment with the same credentials/pastes at a different time in the day the following week if necessary.

Paste	Honey identity	URL	Date (2018)	Time
7	9	`https://pastebin.com/KNuqK0Sy` [1]	6th August	11:30
8	13	`https://pastebin.com/Gpdxnnt1` [1]	7th August	12:15

Table 5.6: The final two pastes

5.6.4 Paste visibility

Pastes 7 was viewed significantly more times in the first 24 hours than paste 8 (Figure 5.9). The most likely reason for this was that paste 7 was in the Pastebin search index almost instantly. Conversely, paste 8 did not appear in the search results during the observation period. The @Dumpmon twitter bot did not tweet links to either of the pastes.

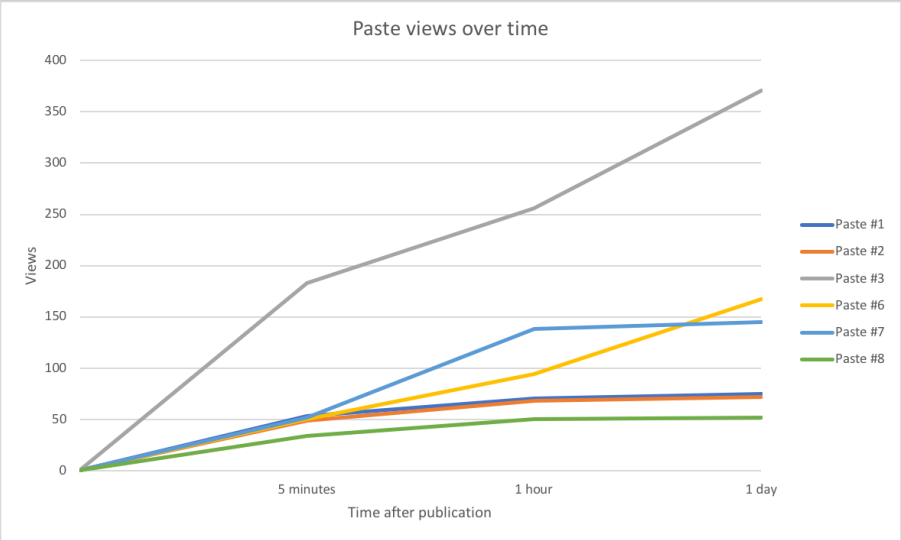


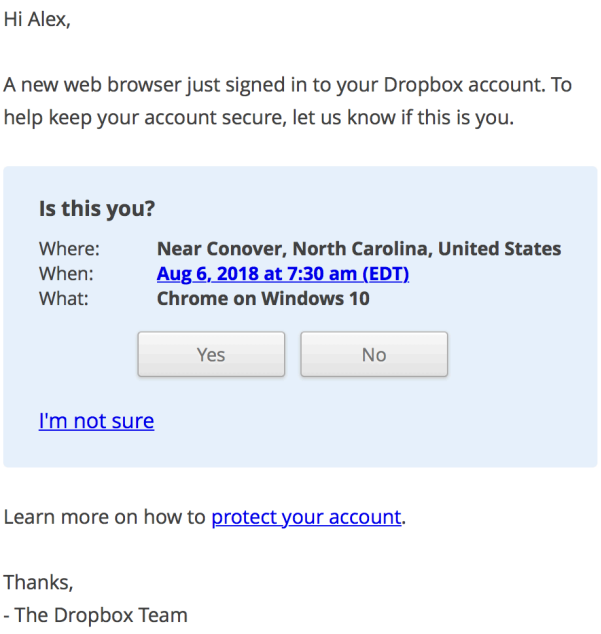
Figure 5.9: Updated chart showing the number of views for each paste

[1] Pastes have now expired.

5.7 More unauthorised access

5.7.1 Intruder #2 – 6th August 2018 at 12:30

The second security event came exactly one hour after paste 7 was published, alerted by a new login notification email from Dropbox (Figure 5.10). Unfortunately, it was not possible to learn anything more about the intruder. Despite checking the Security page for active web sessions as soon as the email was received, the intruder had already logged out. It may be that they instantly recognised the true nature of the account or a tool was being used to validate the credentials. There were no attempts to log into the related email account.


Hi Alex,

A new web browser just signed in to your Dropbox account. To help keep your account secure, let us know if this is you.

Is this you?

Where: **Near Conover, North Carolina, United States**
When: **Aug 6, 2018 at 7:30 am (EDT)**
What: **Chrome on Windows 10**

Yes No

I'm not sure

Learn more on how to protect your account.

Thanks,
- The Dropbox Team

Figure 5.10: New login notification email from Dropbox

5.7.2 Intruder #3 – 11th August 2018 at 15:49

The next case of unauthorised activity occurred on 11th August 2018 at 3:49 pm (Figure 5.11) using the credentials leaked in paste 8 – almost 100 hours after the publication of it.

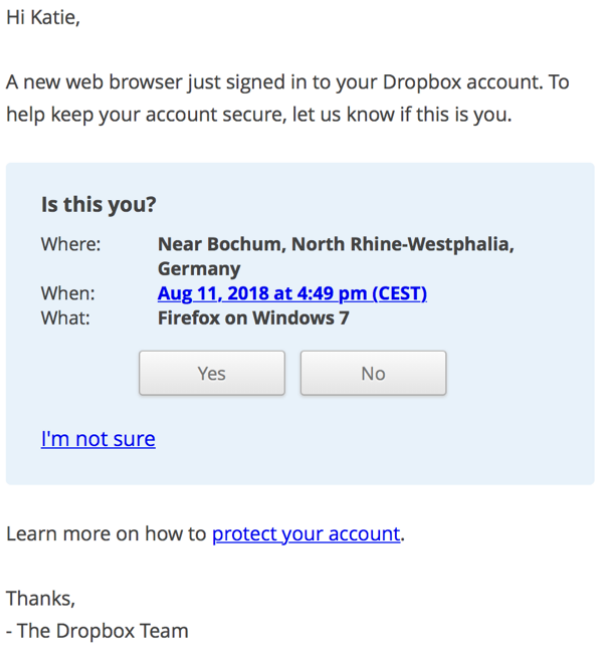


Figure 5.11: Another new login notification email from Dropbox

In this instance, it was possible to get the IP address for the intruder along with information regarding the web session (Figure 5.12).

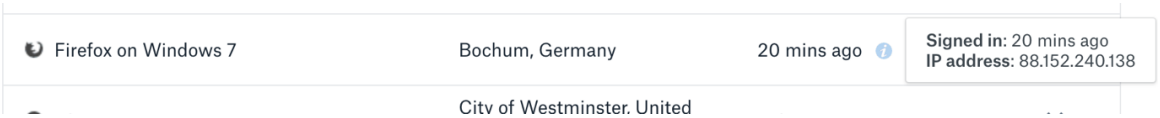


Figure 5.12: The IP address and browser details for intruder #3

The intruder did not log into the Gmail account nor did they attempt to access the "ISG Project" website or email.

5.7.3 Intruder #4 – 13th August 2018 at 22:39

The fourth case of unauthorised access occurred using the Gmail credentials leaked through paste 6. The details of this intruder are displayed in Figures 5.13 and 5.14.

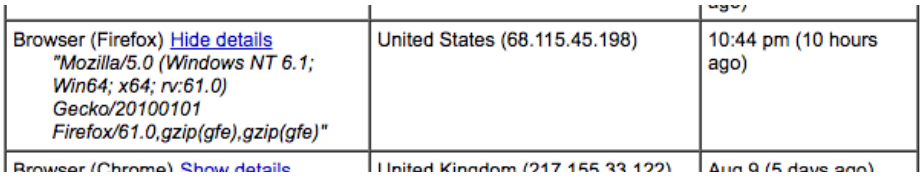


Figure 5.13: User agent and IP address for the intruder #4

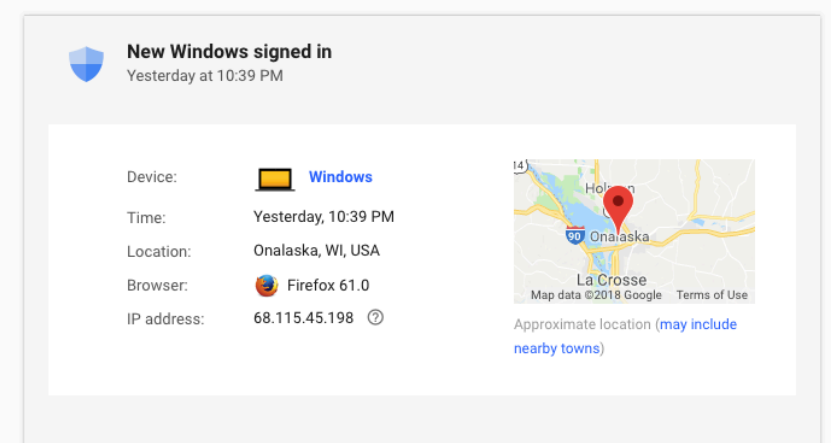


Figure 5.14: User-friendly display of the same information

As with intruder #1, there was no evidence to suggest that this person had viewed the email containing the honeytoken. Nor did they log into the "ISG Project" email or any of the other honey accounts. However, it was possible to see that the intruder had conducted a search using Google whilst still signed into the Gmail account (Figure 5.15).

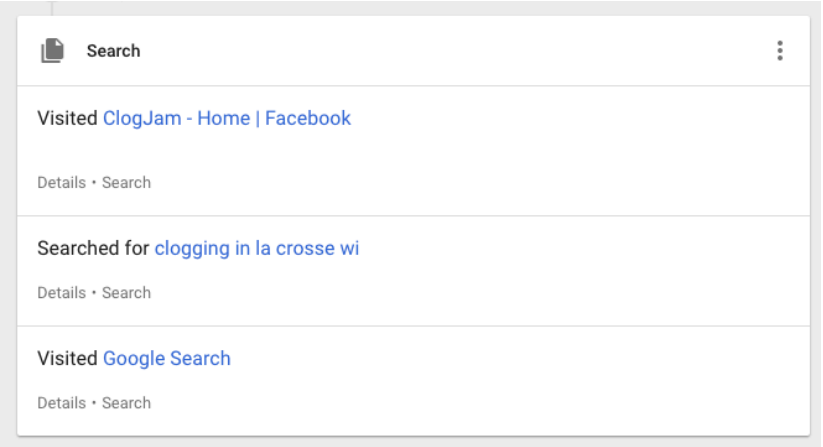


Figure 5.15: Google search activity by intruder #4

5.7.4 Intruder #5 – 14th August 2018 at 12:25

The final security event happened 12 hours later (Figure 5.16), on the last day of the observation period. As in the previous case, the intruder used the credentials leaked in paste 6 to log into the Gmail account. Again, there was no evidence that they logged into any other account and they did not visit the honeytoken URL in the email. Google did not list any further activity for this account.

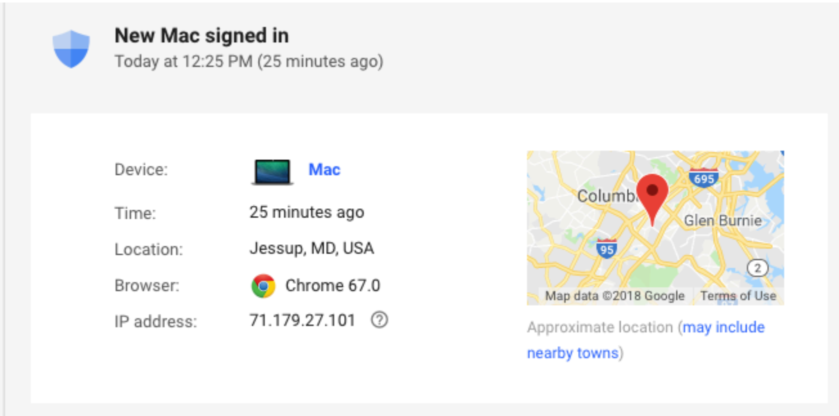


Figure 5.16: Location and browser details of the final intruder

The web server access logs revealed that a browser with the same IP address had viewed the "ISG Project" website, navigating from the home page to that specific honey identity's profile page. An extract of the log file is displayed in Figure 5.17.

```
71.179.27.101 - - [14/Aug/2018:11:28:26 +0000] "GET / HTTP/1.1" 301 178 "-" "Mozilla/5.0
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET / HTTP/1.1" 200 2371 "-" "Mozilla/5.0
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-includes/css/dashicons.min.css?ve
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:27 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:28 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:28 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /people/ HTTP/1.1" 200 3093 "https://
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/832998-768x512.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/1036623-768x513.j
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/213117-768x511.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/325924-768x513.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/1138903-768x512.j
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/265036-768x512.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/themes/isgproject/assets/
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/818819-768x512.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/1138903-300x200.j
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/832998-300x200.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/450271-1-768x512.
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/842567-1-768x512.
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/1036623-300x200.j
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/265036-300x200.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/213117-300x200.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/325924-300x200.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/842567-1-300x200.
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/818819-300x200.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/450271-1-300x200.
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/774095-768x512.jp
71.179.27.101 - - [14/Aug/2018:11:28:40 +0000] "GET /wp-content/uploads/74095-300x200.jpg
71.179.27.101 - - [14/Aug/2018:11:28:48 +0000] "GET /profile/emma-fields/ HTTP/1.1" 200 2
71.179.27.101 - - [14/Aug/2018:11:28:48 +0000] "GET /wp-content/themes/isgproject/assets/
```

Figure 5.17: An extract of a log file showing intruder's activity on the website

5.8 Summary

After six weeks, it was necessary to end the observation period to allow sufficient time for to write up the project. A summary of all of the instances of the unauthorised activity and the different intruders is displayed in Tables 5.7 and 5.8 respectively. The findings from this experiment, along with potential improvements, are discussed in the next chapter.

Alert	Service	Date	Time	Honey identity	Intruder
Login email	Gmail	July 18th	12:04	1	1
2FA SMS	Dropbox	July 18th	12:04	1	1
Login email	Dropbox	August 6th	12:30	9	2
Login email	Dropbox	August 11th	15:49	13	3
Login email	Gmail	August 13th	22:39	1	4
Activity	Gmail	August 14th	00:15	1	4
Login email	Gmail	August 14th	12:25	1	5

Table 5.7: All of the monitoring alerts in the observation period

Intruder	Source paste	Hours after leak	Location	OS	Browser
1	6	0:36	Louisville, KY, USA	Windows 7	Firefox 60
2	7	1:00	Conover, NC, USA	Windows 10	Chrome
3	8	99:34	Bochum, Germany	Windows 7	Firefox
4	6	635:09	Onalaska, WI, USA	Windows 7	Firefox 61
5	6	648:55	Jessup, MD, USA	macOS (High Sierra)	Chrome 67

Table 5.8: Summary of intruders and time it took to get owned

6 Discussion

This chapter discusses the results and shortcomings of the experiment, improvements that can be made to the framework and other future work.

6.1 The experiment

6.1.1 Results

There were five events of unauthorised logins during the observation period of the experiment. The fastest occurred only 34 minutes after the credentials were published and the slowest took 27 days. With a small result set, it is difficult to draw any conclusions, other than speculative ones.

Use of credentials

It had been expected that each paste containing a plaintext password would result in at least one event of unauthorised access. However, it was surprising that these were the only cases. The most viewed paste was paste 3 which contained MD5 hashes. It would not have required much work to recover the passwords. Indeed, just by using the hash as a search term on Google[1], it would have been possible to obtain the correct password. Despite the visibility and the ease of recovery, the credentials from that paste were not used. That said, there is no visible difference between the hash of a weak password and that of a strong one other than the actual hash value. In other words, it would not have been possible for someone to tell that the hashes would have been easy to crack just by looking at it. It is likely that with a small data breach for a private website, the effort involved in cracking was not considered worth the reward of (a maximum of) ten credentials.

It was also a surprise that there were no cases of a validated password being used to log into another service. Even in the case of intruder #1, who used the password advertised for the `isgproject.org` email to log into Dropbox, they had not previously validated it.

Intruder locations

The details of the user's web browser and location were captured for each event, however, these details should be considered with caution. The web browser is determined by the

[1] For example, `https://www.google.com/search?hl=en&q=79b5afeffc388a330c59aee934bc9163`

`User-Agent` string in the HTTP header which can easily be spoofed. The location reported is often inaccurate as it tends to relate to the ISP rather than the individual. It can also be falsified through the use of proxies or VPNs. Onaolapo et al determined that cybercriminals on paste websites exhibited a level of *location malleability*; masquerading their origins of access to appear closer to the advertised location [23]. The locations were not provided in this experiment's pastes, but all had English sounding names. Therefore, it makes sense that the intruders were "from" the US (i.e. the largest English-speaking country) in four of the five cases. Figure 6.1 shows the reported locations of the intruders on a map.

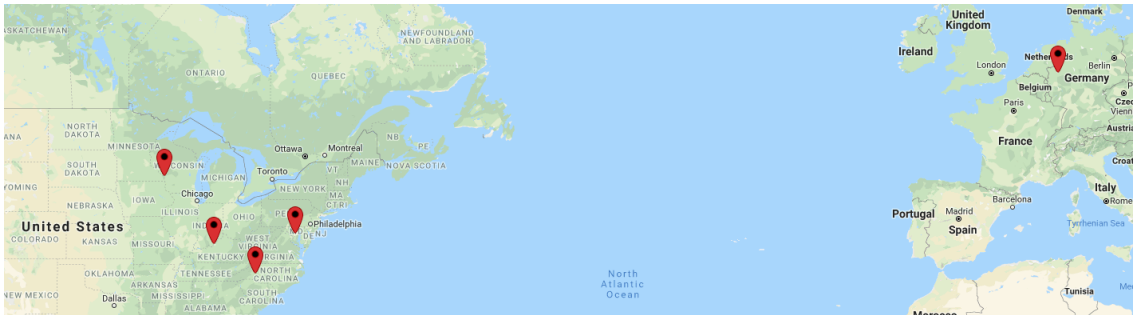


Figure 6.1: A map showing the reported locations of the intruders

All of the available IP addresses were checked against the Spamhaus (`www.spamhaus.org`) blocklists. The IP address for intruder #4 appeared on one list as part of a network of malware-infected computers, displayed in Figure 6.2. This would suggest that the machine was being used as a proxy to connect to the account and that their search activity was simply a red herring rather than an actual interest in clogging.

RESULTS OF LOOKUP

68.115.45.198 is listed

This IP address was detected and listed 3 times in the past 28 days, and 0 times in the past 24 hours. The most recent detection was at Sat Aug 4 23:35:00 2018 UTC +/- 5 minutes

This IP address is infected with, or is NATting for a machine infected with the Conficker malicious botnet.

Figure 6.2: Results of a search for an intruder's IP address on Spaumhaus

Intruder activity

There was no evidence of further activity after any of the intruders had gained access to an account. With the exception of intruder #1, none of the intruders re-used the credentials to log into another account. There were no login attempts (successful or otherwise) for any of

the private email or website accounts during the entirety of the observation period. Similar activity was observed by Onaolapo et al [23] in their study. They named these types of access as *Curious* and described them as follows:

> *These accesses constitute the most basic type of access to stolen accounts. After getting hold of account credentials, people login on those accounts to check if such credentials work. Afterwards, they do not perform any additional action.*

They also acknowledged that this type of access includes those cybercriminals experienced enough to quickly determine the purpose of the account and thus avoid interactions with it after logging in.

Paste visibility

The initial visibility of the pastes relied on them being discovered on the "Archive" page. This page displays the most 50 recently created pastes. If there was a lot of activity at the same time that a paste was published, then it would only appear on this page for a short time. This could explain why only one paste was spotted by the @dumpmon Twitter bot.

Another important variable in each paste's visibility was the amount of time it took to appear in the search index. This ranged from minutes to three days, and one paste was not indexed at all. The fact that the credentials from that paste were used several days after it left the "Archive" page, without any search visibility, is somewhat mystifying.

The final two logins occurred on the last two days of the observation period; roughly 27 days after the source paste was published. The view count jumped from approximately 200 to over 500 in that time. It is difficult to explain this since Pastebin does not offer more detailed analytics. It was not possible to find any web pages that linked to this paste, so it may have been that searches for pages containing Dropbox links had seen an increase. It was surprising that the 300 views resulted in only two additional security events. It is conceivable that potential intruders gave up after visiting the first two dead links. Equally likely is that people viewing the Dropbox folder did not act on the plaintext credentials. Another possibility is that this paste was discovered by search bots looking for Dropbox URLs. However, the nature of this paste required a human (or a *very* advanced bot) to actually obtain and use the credentials from the link.

Removal of pastes

There is a content restriction against material that "is unlawful or promotes unlawful activities" in Pastebin's acceptable use policy [47]. Even though the pastes were designed to look as though they were of illicit nature, none of them were removed from the website. Furthermore, none of the accounts that had their passwords published in plaintext were disabled by the respective service provider. This would suggest that those service providers do not search for the presence of credentials as a proactive means of protection.

6.1.2 Shortcomings of the experiment

Timescale

Given the timescale of this MSc project and the time taken for research and implementation, the experiment phase was conducted over a relatively short period of just over six weeks. In comparison, Onaolapo et al [23] monitored access to their Gmail accounts for seven months. With more time, it would have been possible to generate a lot more (full) honey identities, varied the methods used to publish the credentials and allowed for a longer period of observation.

Publishing credentials

The success that other researchers [23] had leaking credentials through paste websites was a big factor in the decision to take a similar approach. However, they did not reveal the exact format of their pastes other than that they were email and password pairs. It is unknown whether they had the ethical concerns regarding re-publishing previously leaked passwords discussed in *Chapter 5*. Furthermore, their experiment took place over two years ago since which time several of the paste websites that they used have ceased to exist. It may also be the case that cybercriminals are aware that Pastebin's popularity has led to its use by security researchers and as such avoid it.

Pastes 2, 3, 4 and 6 were all configured so that they would never expire. This was done to facilitate the marking of this project. However, an unforeseen downside to this was that those credentials could not be re-used in any future pastes since it would have been suspicious for the same credentials to appear in multiple pastes. Additionally, it would have no longer been possible to link the credentials to their source paste. A better approach would have been to configure them to expire after one or two weeks. This would allow for the

credentials (or even the entire paste) to be republished at a later date which would be conducive for further experimentation.

Primary email account

A fictitious company was created to provide a "cover story" for the publishing of credentials. It was acknowledged that it was easy to identify the company as a fictional one, however, it was not considered that the stolen credentials for a private email address may be less desirable than one from a public service. This was indicated by the results from paste 6 where the credentials for both the Gmail and the private email accounts were displayed in plaintext. The Gmail account was accessed three times but there were no attempts to log into the private email. This is by no means conclusive and further research would need to be carried out to see whether this was a factor.

6.2 Future work

This section discusses the various work that can be carried out in the future. This includes possible enhancements to the honey identities and monitoring infrastructure, further experimentation and other research topics.

6.2.1 Improvements to the honey identities

Bigger digital footprints

The size of the honey identities was limited due to the various constraints discussed in *Chapter 3.2*. This was, along with the timescale, the biggest restriction on the experiment. DeBlasio et al did not have the same ethical concerns with breaking terms and conditions when they developed Tripwire. They argued that the scientific merit of their work outweighed the low legal risk [30].

It is hoped that the body of work presented in this project would assist in receiving permission from service providers to create artificial accounts on their platform. Furthermore, this could also lead to partnerships which would offer the prospect of additional monitoring tools and bulk account creation.

Additional attributes

The honey identities generated had sufficient attributes to register on the websites chosen in *Chapter 3.3*. With a bigger digital footprint, more attributes may be required. Some of these could be problematic but others would be simple to generate. A few examples are as follows:

- **Address.** A randomly generated address would be of poor quality and would likely be rejected by most web services. A better approach would be to use a real address however this could be difficult. It would not be appropriate to use someone else's address without permission. The address of the University could be used but this could then be used to distinguish honey identities. Another option could be to use a different service address for each identity, but this would be expensive.
- **Place of birth.** Place of birth and other personal data, such as physical attributes and marital status, could be randomly generated but other related attributes would need to be considered in that process.
- **Phone number**. As mentioned with regards to 2FA SMS verification in *Chapter 4.3.2*, a valid phone number could be provided using Twilio or a similar service.
- **Bank account.** Valid banking details and other financial data would require a partnership with a bank.

More developed digital footprints

The honey identities used in the experiment were generated just one week prior to the publication of their credentials. This was not an issue for some web services but on most social networking websites the date the account was created is public information. An example of this is shown in Figure 6.1. This information could help to distinguish a honey identity from a real one. To avoid this, the identity would either have to be generated a long time before any experiment or the date would need to be falsified through a partnership with the service provider.

Similarly, none of the honey identities were active after the initial content was populated. The credibility of the honey identities would be greatly enhanced if they were sufficiently active on various social media platforms. Despite the availability of tools to automate the posting of content, this would still be a large project in itself. Furthermore, it would probably require, at the very least, approval from the service provider to avoid the accounts being flagged as spam and disabled.

Figure 6.3: A public Twitter profile showing the "born" date

The email accounts were populated using a publicly available dataset. Even after certain modifications, it would have been easy to trace the original source of these emails. It would be better to create the dataset from scratch and to develop a system to send new emails to the honey identities after the preliminary seeding.

6.2.2 Improvements to monitoring infrastructure

The monitoring infrastructure relied on the observation of a single email inbox and manually recording any events. In some cases, for example Dropbox, it was important to react quickly to gather further information on the intruder as it was only displayed for current web sessions. With a single observer, this could have proved to be challenging if an event had occurred in the middle of the night.

Whilst it may not be realistic to remove all manual processes there is a lot that could be enhanced with automation. The incoming emails could be parsed and fed into a database. This would, in turn, allow for easier tracking and analysis as well as triggering other information gathering scripts. It would also lead to the possibility of a web application to view and analyse the data along with additional alerting mechanisms such as SMS messages.

6.2.3 Further experimentation

After addressing the shortcomings of the experiment and improving the honey identities, there would be a lot of scope for further experimentation.

Comparing the desirability of different honey identities

With larger, more developed digital footprints, it would be possible to create multiple genres of honey identities. The contents of the email account, as discussed in *6.2.1*, would also help to define the character of a honey identity. It would then be feasible to conduct an experiment comparing these different genres. For example, Male vs Female, American vs Chinese, Young professionals vs Gamers. An investigation of this nature would help to identify the factors that make an identity desirable to cybercriminals. This, in turn, could be used to further improve the honey identities.

Partial password re-use

Another aspect that could be studied is the partial re-use of passwords. In the experiment conducted for this project, the same password was used for all of a honey identity's account (in most cases). Studies have shown that it is actually more common for users to partially re-use a password rather than exactly re-use it. Partial re-use is when the same substring of a password is used for multiple passwords, for example *Arsenal123* and *Arsenal2018*. Wang et al developed an algorithm that they claim can guess 30% of passwords modified in this manner within 10 attempts [4].

Different means of publishing the credentials

In addition to underground forums, malware and phishing (see *Chapter 5.2*), "accidental" means could be investigated. For example, there were a number of data breaches last year involving Amazon Web Services (AWS). Misconfiguration of *buckets* meant that private data was made public and, in one case, sensitive personal information for 123 million households was publicly exposed [48].

Wait to be hacked

To fully answer the question posed in this project's title, an experiment could adopt the approach used for Tripwire and wait for a data breach [30]. This would require a significant number of honey identities and a very long observation period however it would, arguably, produce more accurate results than DeBlasio et al's research. One weakness of their study was that a data breach would go unnoticed should an attacker not use the credentials to log into the respective honey email account. By using a honey identity, one can speculate that it would be more likely that a login would occur in at least one of its accounts.

6.2.4 Other research topics

The author of this project was surprised by the speed with which the website started to experience vulnerability scans and brute force attacks. Further research into this could make an interesting project, with a similar title, in its own right.

6.2.5 Business use

Shabtai et al own a patent for a system that is designed to offer protection against reconnaissance and APTs. It comprises of the generation of artificial profiles and accounts, monitoring the activity of these accounts and reporting any suspicious activity from third parties to contact the user accounts [49]. Their framework differs from the one in this project in that they are only monitoring for incoming messages rather than any unauthorised activity. Their theory is that any communication to these fake identities could indicate the presence of a sophisticated and targeted attack.

A similar tactic could be utilised with honey identities and the monitoring infrastructure proposed in this project. Since all incoming emails are forwarded to a single mailbox, it would be straightforward to scan these emails for certain keywords, URLs, and malware. If this was combined with the Tripwire approach for data breach detection, it could prove to be an effective early warning system for a company.

Whilst the limited number of results was slightly disappointing, this was a known risk of the project due to the short period of time available to conduct the experiment and, furthermore, the objective of the experiment was to test the implementation framework and monitoring infrastructure. In that regard, the experiment was a success. The five recorded events of unauthorised access demonstrated that the monitoring infrastructure worked well, and it would only need a small number of improvements for further research. By additionally enhancing the honey identities and addressing the acknowledged shortcomings of the experiment, there is a lot of scope for future work and a viable commercial product could even be developed.

This page intentionally left blank

7 Conclusion

The objective of this study was to answer the question:

How long does it take to get owned?

To achieve this, it was first necessary to understand several background topics covering identity, the use of honeypots and the illegal sharing of personal data. This was supplemented with research into different methods for monitoring access to online accounts. Using this preliminary research, a prototype framework for generating honey identities was designed and implemented along with an infrastructure to monitor their activity. Finally, an experiment was conducted to test the framework by publishing the credentials for eleven identities on paste websites in several different formats. There were five instances of unauthorised activity, related to three different pastes, with the fastest occurring only 34 minutes after the leak of the relevant password.

It would be unwise to draw too many conclusions from the results of the experiment. With just five intruders, the sample size is too small to analyse and identify trends. It was expected that the presence of plaintext passwords would lead to unauthorised access, however, it was a surprise that the pastes containing password hashes did not lead to any intruder logins. It could be that the "right" people did not notice the pastes or that they considered the effort in cracking was not worth the reward. Despite the small result set, the monitoring infrastructure was proven to work and with improvements to the honey identities, this could be an excellent platform for increased scope and further experimentation.

During the course of this project, the use of stolen credentials has been investigated but the underlying issue is the prevalence of poor password behaviour. It would be easy to blame the user, and expect them to accept any consequences, as better password practices have been promoted for decades, but the regular mind simply cannot remember lots of strong passwords. This problem is exacerbated by complex password restrictions and regular password expiration [5]. There are also significant costs to service providers and organisations in preventing, detecting and cleaning up compromised accounts [7]. As shown in the case of Dropbox, one instance of password re-use can, in turn, lead to a significant data breach [36].

Service providers could do more to prevent password re-use or encourage the use of technology to mitigate the potential risk. For example, the *Have I Been Pwned?* API allows a website to securely check for the presence of a password in a breach and thus reject them. During the course of this project, GitHub have implemented a similar approach to warn users should their password have appeared in a breach (Figure 7.1) [50].

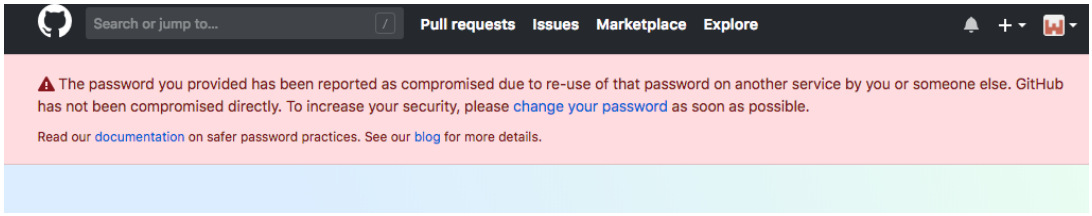


Figure 7.1: A warning that is now displayed on GitHub

However, this tactic could just lead to an increase in *partial* password re-use or worse, alienate a potential customer. To quote Grzegorz Milka, a Google software engineer [51]:

> *It's about how many people would we drive out if we force them to use additional security.*

Over half of Americans do not recognise the term *two factor authentication* [52]. It is therefore not surprising that the adoption rate remains low decades after its inception. Milka revealed that less than 10% of active Google accounts use 2FA [51]. Yet this is actually considered a high percentage in comparison to other services. For example, Dropbox reported that their adoption rate was less than 1% [53].

Password managers help users generate random, strong and unique passwords for every account without the need to memorise them. However, like 2FA, usage remains low and it tends to be a tool for the more security aware user [51].

A more radical solution to the problem of poor password behaviour would be to remove passwords altogether. Their issues are well-known and have been a matter for debate since 1979 [54]. Numerous replacement schemes have been proposed in the intervening years, such as graphical passwords, Passfaces, grids, and token-based credentials, but all have failed to achieve widespread adoption. Unfortunately, users are familiar with passwords and it is difficult to implement change [55].

The unfortunate truth is that, for the foreseeable future, passwords will remain as the main form of authentication on most websites. Whilst this is the case, users will continue to (re-)use weak passwords, websites will continue to be the source of data leaks, cybercriminals will continue to extract and trade credentials from these, and those same users will get owned.

This page intentionally left blank

8 Bibliography

[1] Verizon, "2018 Data Breach Investigations Report," 2018.

[2] L. Ablon, Rand Corporation, and Institute for Civil Justice (U.S.), *Consumer attitudes toward data breach notifications and loss of personal information*. 2016.

[3] T. Evans, "World's Biggest Data Breaches," *information is beautiful*, 2018. [Online]. Available: http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/. [Accessed: 01-Aug-2018].

[4] C. Wang, S. T. K. Jan, H. Hu, and G. Wang, "Empirical Analysis of Password Reuse and Modification across Online Service," 2017.

[5] S. Pearman *et al.*, "Let's Go in for a Closer Look: Observing passwords in their natural habitat," *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur. - CCS '17*, pp. 295–310, 2017.

[6] N. Mueller, "Credential stuffing," *OWASP*, 2015. [Online]. Available: https://www.owasp.org/index.php/Credential_stuffing. [Accessed: 08-Aug-2018].

[7] K. C. Wang and M. K. Reiter, "How to end password reuse on the web," pp. 1–16, 2018.

[8] B. Parkinson, D. E. Millard, K. O'Hara, and R. Giordano, "The digitally extended self: A lexicological analysis of personal data," *J. Inf. Sci.*, vol. 44, no. 4, pp. 552–565, 2018.

[9] R. Rodogno, "Personal Identity Online," *Philos. Technol.*, vol. 25, no. 3, pp. 309–328, Sep. 2012.

[10] R. Clarke, "The digital persona and its application to data surveillance," *Inf. Soc.*, vol. 10, no. 2, pp. 77–92, 1994.

[11] S. Olshansky, "Online Identity: Who, Me?," *Internet Society*, 2016. [Online]. Available: https://www.internetsociety.org/resources/doc/2016/online-identity-who-me/. [Accessed: 18-Feb-2018].

[12] L. Srivastava, T. Kelly, C. Yung Lu, and L. Yu, "ITU Internet Report 2006: digital.life," Geneva, 2006.

[13] S. Clauß and M. Köhntopp, "Identity management and its support of multilateral security," *Comput. Networks*, vol. 37, no. 2, pp. 205–219, 2001.

[14] M. Madden, S. Fox, A. Smith, and J. Vitak, "Online identity management and search in the age of transparency," *Pew Internet Am. Life Proj.*, no. December, p. 50, 2007.

[15] J. Mellmer, R. T. Young, A. D. Perkins, J. M. Robertson, and J. Sabin, "Managing digital identity information," US8631038B2, 2000.

[16] L. Spitzner, "The honeynet project: Trapping the hackers," *IEEE Secur. Priv.*, vol. 1, no. 2,

pp. 15–23, 2003.

[17] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A Survey on Honeypot Software and Data Analysis," 2016.

[18] M. Burgess, "The Emmanuel Macron email hack warns us fake news is an ever-evolving beast," *Wired*, 2017. [Online]. Available: https://www.wired.co.uk/article/france-election-macron-email-hack. [Accessed: 25-Feb-2018].

[19] C. Dickey, "Did Macron Outsmart Campaign Hackers?," *Daily Beast*, 2017. [Online]. Available: https://www.thedailybeast.com/did-macron-outsmart-campaign-hackers. [Accessed: 02-Feb-2018].

[20] M. Scott, "U.S. Far-Right Activists Promote Hacking Attack Against Macron," *The New York Times*, New York, 06-May-2017.

[21] L. Spitzner, "Honeytokens: The Other Honeypot," *Symantec*, 2003. [Online]. Available: https://www.symantec.com/connect/articles/honeytokens-other-honeypot. [Accessed: 01-Aug-2018].

[22] S. Thorpe, "SMS for 2FA: What Are Your Security Options?," *Authy*, 2016. [Online]. Available: https://authy.com/blog/security-of-sms-for-2fa-what-are-your-options/. [Accessed: 01-Aug-2018].

[23] J. Onaolapo, E. Mariconti, and G. Stringhini, "What Happens After You Are Pwnd : Understanding The Use Of Leaked Account Credentials In The Wild," *Proc. ACM SIGCOMM Conf. Internet Meas. Conf.*, pp. 65–79, 2016.

[24] B. Butler, B. Wardman, and N. Pratt, "REAPER: An automated, scalable solution for mass credential harvesting and OSINT," *eCrime Res. Summit, eCrime*, vol. 2016–June, pp. 71–80, 2016.

[25] E. Nunes *et al.*, "Darknet and deepnet mining for proactive cybersecurity threat intelligence," *IEEE Int. Conf. Intell. Secur. Informatics Cybersecurity Big Data, ISI 2016*, pp. 7–12, 2016.

[26] T. Hunt, "Making Light of the 'Dark Web' (and Debunking the FUD)," 2018. [Online]. Available: https://www.troyhunt.com/making-light-of-the-dark-web-and-debunking-the-fud/. [Accessed: 15-Feb-2018].

[27] J. Shakarian, A. T. Gunn, and P. Shakarian, "Exploring malicious hacker forums," in *Cyber Deception: Building the Scientific Foundation*, S. Jajodia, V. S. Subrahmanian, V. Swarup, and C. Wang, Eds. Cham: Springer International Publishing, 2016, pp. 259–282.

[28] K. Thomas *et al.*, "Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and*

Communications Security, 2017, pp. 1421–1434.

[29] A. Shulman, "The underground credentials market," *Comput. Fraud Secur.*, vol. 2010, no. 3, pp. 5–8, 2010.

[30] J. DeBlasio, S. Savage, G. M. Voelker, and A. C. Snoeren, "Tripwire: Inferring Internet Site Compromise," *Internet Meas. Conf.*, p. 14, 2017.

[31] T. Malderle, M. Wübbeling, S. Knauer, A. Sykosch, and M. Meier, "Gathering and Analyzing Identity Leaks for a Proactive Warning of Affected Users," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, 2018, pp. 208–211.

[32] T. Hunt, "Have I been pwned? Pastes," *Have I been pwned?*, 2018. [Online]. Available: https://haveibeenpwned.com/Pastes. [Accessed: 18-Feb-2018].

[33] S. Migliano, "Dark Web Market Price Index (UK Edition)," *Top10vpn.com*, 2018. [Online]. Available: https://www.top10vpn.com/privacy-central/cybersecurity/dark-web-market-price-index-feb-2018-uk/. [Accessed: 01-Aug-2018].

[34] J. Swearingen, "Gmail Gets a Major Face-lift and Productivity Boost, Starting Today," *New York Magazine*, 2018. [Online]. Available: http://nymag.com/selectall/2018/04/how-to-turn-on-google-gmail-redesign-and-new-features.html. [Accessed: 01-Aug-2018].

[35] "Celebrating half a billion users," *Dropbox*, 2016. [Online]. Available: https://blogs.dropbox.com/dropbox/2016/03/500-million/. [Accessed: 01-Aug-2018].

[36] S. Gibbs, "Dropbox hack leads to leaking of 68m user passwords on the internet," *The Guardian*, 2016. [Online]. Available: https://www.theguardian.com/technology/2016/aug/31/dropbox-hack-passwords-68m-data-breach. [Accessed: 01-Aug-2018].

[37] M. Mazzilli, "Twitter warns fake account purge to keep erasing users, shares drop 19 percent," *Reuters*, 2018. [Online]. Available: https://www.reuters.com/article/us-twitter-results/twitter-beats-estimates-on-revenue-but-monthly-usage-falls-idUSKBN1KH17X. [Accessed: 01-Aug-2018].

[38] J. Finkle, "Twitter urges all users to change passwords after glitch," *Reuters*, 2018. [Online]. Available: https://www.reuters.com/article/us-twitter-passwords/twitter-urges-all-users-to-change-passwords-after-glitch-idUSKBN1I42JG. [Accessed: 01-Aug-2018].

[39] C. Jones, "Twitter says 250,000 accounts have been hacked in security breach," *The Guardian*, 2013. [Online]. Available: https://www.theguardian.com/technology/2013/feb/02/twitter-hacked-accounts-reset-security. [Accessed: 01-Aug-2018].

[40] A. Yuhas, "Hacker who stole nude photos of celebrities gets 18 months in prison," *The Guardian*, 2016. [Online]. Available: https://www.theguardian.com/technology/2016/oct/27/nude-celebrity-photos-hacker-prison-sentence-ryan-collins. [Accessed: 01-Aug-2018].

[41] A. Balakrishnan and J. Boorstin, "Instagram says it now has 800 million users, up 100 million since April," *CNBC*, 2017. [Online]. Available: https://www.cnbc.com/2017/09/25/how-many-users-does-instagram-have-now-800-million.html. [Accessed: 01-Aug-2018].

[42] C. Newton, "An Instagram hack hit millions of accounts, and victims' phone numbers are now for sale," *The Verge*, 2017. [Online]. Available: https://www.theverge.com/2017/9/1/16244304/instagram-hack-api-bug-doxagram-selena-gomez. [Accessed: 01-Aug-2018].

[43] "Microsoft to acquire GitHub for $7.5 billion," *Microsoft News Centre*, 2018. [Online]. Available: https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/. [Accessed: 01-Aug-2018].

[44] GitHub, "GitHub Security," 2018. [Online]. Available: https://help.github.com/articles/github-security/. [Accessed: 01-Aug-2018].

[45] S. Sharwood, "Uber quits GitHub for in-house code after 2016 data breach," *The Register*, 2018. [Online]. Available: https://www.theregister.co.uk/2018/02/07/uber_quit_github_for_custom_code_after_2016_data_breach/. [Accessed: 01-Aug-2018].

[46] "Brute Force Attacks," *WordPress.org*, 2018. [Online]. Available: https://codex.wordpress.org/Brute_Force_Attacks. [Accessed: 01-Aug-2018].

[47] Pastebin, "Pastebin.com Terms of Service," 2018. [Online]. Available: https://pastebin.com/doc_terms_of_service. [Accessed: 01-Aug-2018].

[48] D. O'Sullivan, "Home Economics: How Life in 123 Million American Households Was Exposed Online," *UpGuard*, 2017. [Online]. Available: https://www.upguard.com/breaches/cloud-leak-alteryx. [Accessed: 18-Feb-2018].

[49] A. Shabtai, R. Puzis, and Y. Elovici, "Social network honeypot," US 9509716 B2, 2015.

[50] GitHub, "New improvements and best practices for account security and recoverability," *The GitHub Blog*, 2018. [Online]. Available: https://blog.github.com/2018-07-31-new-improvements-and-best-practices-for-account-security-and-recoverability/. [Accessed: 08-Aug-2018].

[51] I. Thomson, "Who's using 2FA? Sweet FA. Less than 10% of Gmail users enable two-

factor authentication," *The Register*, 2018. [Online]. Available: https://www.theregister.co.uk/2018/01/17/no_one_uses_two_factor_authentication/. [Accessed: 01-Aug-2018].

[52] P. H. O'Neill, "Most Americans have never heard of multi-factor authentication," *Cyberscoop*, 2017. [Online]. Available: https://www.cyberscoop.com/two-factor-authentication-duo-security-yubikey/. [Accessed: 01-Aug-2018].

[53] P. Heim, "An inside look at how we keep customer data safe," *Dropbox*, 2016. [Online]. Available: https://blogs.dropbox.com/business/2016/02/dropbox-customer-data-safety/. [Accessed: 01-Aug-2018].

[54] R. Morris and K. Thompson, "Password Security: A Case History," *Commun. ACM*, vol. 22, no. 11, pp. 594–597, Nov. 1979.

[55] S. Aebischer *et al.*, "Pico in the Wild: Replacing Passwords, One Site at a Time," *Proc. 2nd Eur. Work. Usable Secur.*, no. April, pp. 1–13, 2017.

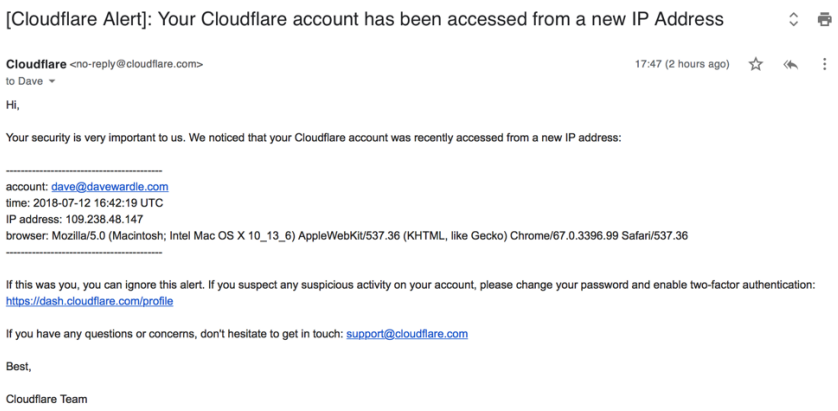
This page intentionally left blank

Appendix A
Further examples of monitoring alerts

This appendix displays screenshots of monitoring alerts from a variety of websites. They have been included to show the different levels of detail that is provided by the different service provider.

Login notification emails

Figure A.1 is an example of a new login notification email from Cloudflare. It displayed a lot more information than was included in similar emails from any of the web services selected for the experiment.


[Cloudflare Alert]: Your Cloudflare account has been accessed from a new IP Address

Cloudflare <no-reply@cloudflare.com> 17:47 (2 hours ago)
to Dave

Hi,

Your security is very important to us. We noticed that your Cloudflare account was recently accessed from a new IP address:

---

account: dave@davewardle.com
time: 2018-07-12 16:42:19 UTC
IP address: 109.238.48.147
browser: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36

If this was you, you can ignore this alert. If you suspect any suspicious activity on your account, please change your password and enable two-factor authentication:
https://dash.cloudflare.com/profile

If you have any questions or concerns, don't hesitate to get in touch: support@cloudflare.com

Best,

Cloudflare Team

Figure A.1: A new login notification email from Cloudflare

Login history and recent activity

Figures A.2 to A.10 are screenshots taken of the login history and recent activity pages on a number of websites.

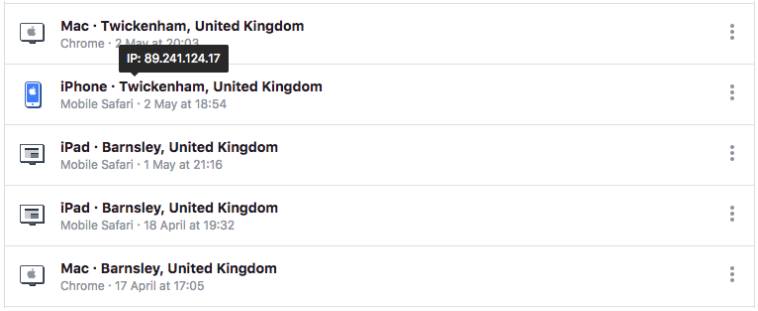


Figure A.2: An extract of the login history page on Facebook

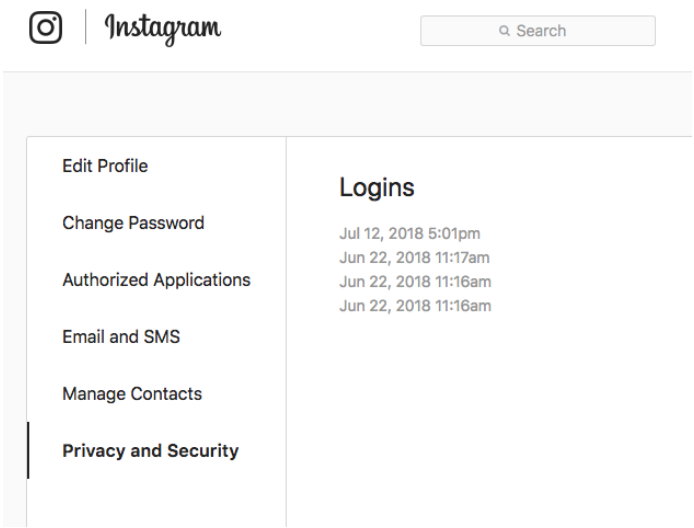


Figure A.3: Instagram only displayed the dates and times of logins

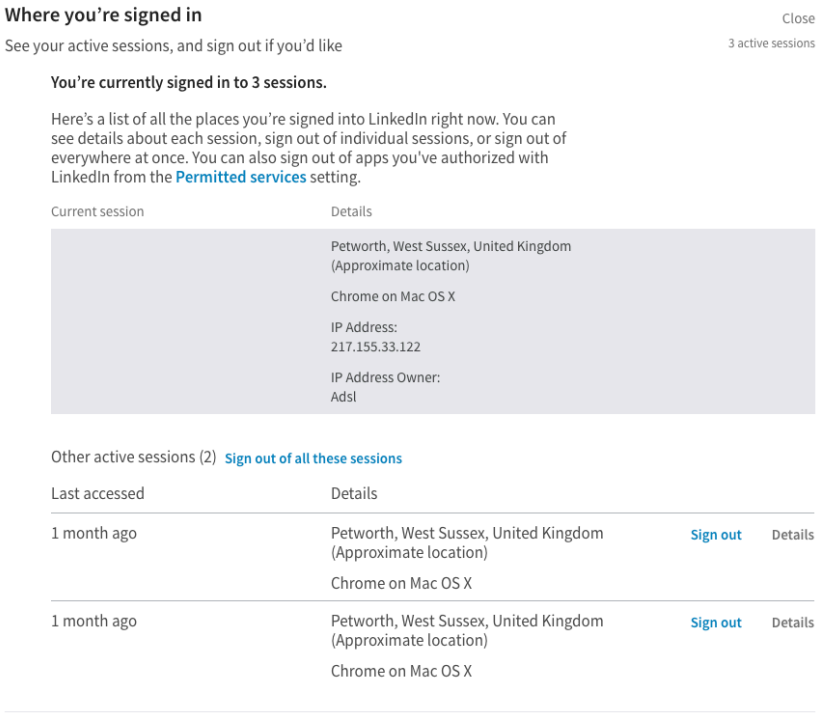


Figure A.4: LinkedIn displayed information about active web sessions only

01/05/2018, 13:00:24 GMT	United Kingdom (WSX) 217.155.33.122	iPad
30/04/2018, 15:17:47 GMT	United Kingdom (HAM) 151.226.161.191	Smart TV
28/04/2018, 15:46:13 GMT	United Kingdom (SRY) 89.241.124.17	Streaming stick
26/04/2018, 04:50:22 GMT	United Kingdom (SRY) 188.221.53.39	Smart TV
17/04/2018, 10:33:04 GMT	United Kingdom (HNS) 89.241.120.176	Computer
16/04/2018, 17:01:02 GMT	United Kingdom (HNS) 89.241.120.176	Streaming stick

Figure A.5: The device and location details on Netflix were nonspecific

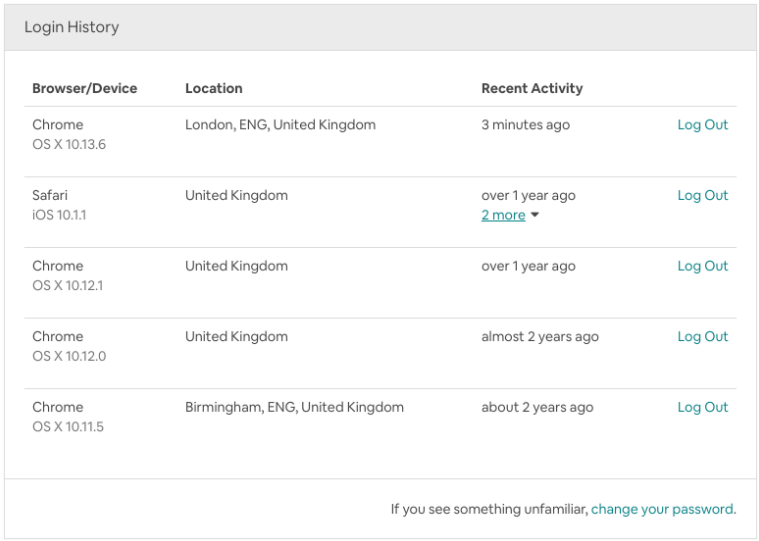


Figure A.6: Airbnb did not provide the IP address but did include the session status

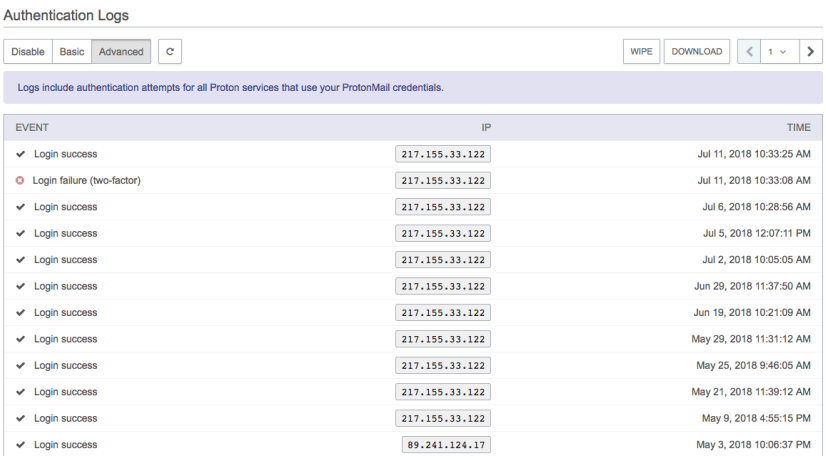


Figure A.7: Protonmail displayed the details of failed logins

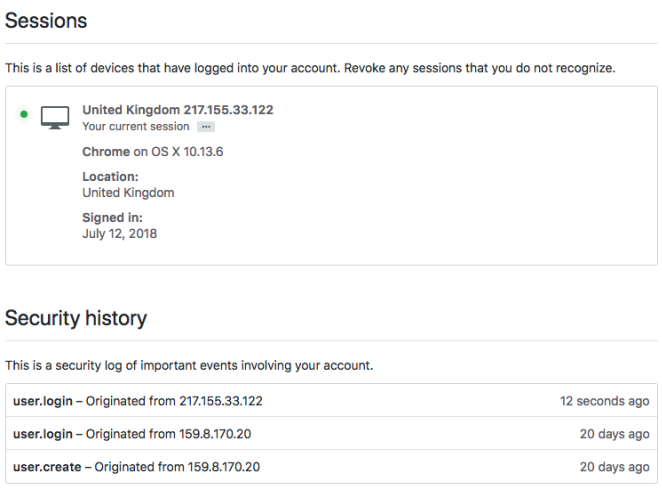


Figure A.8: As well as a list of active sessions, GitHub displayed a full audit log of security-related actions

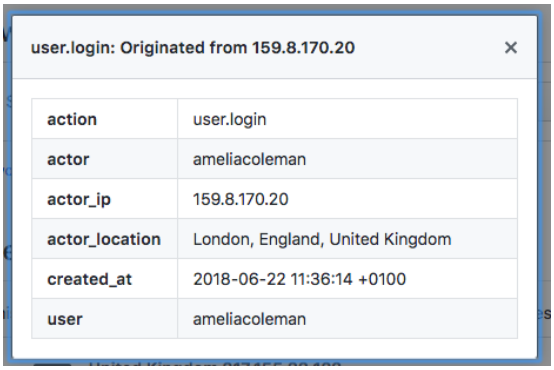


Figure A.9: Further information of one of the events displayed in A.8

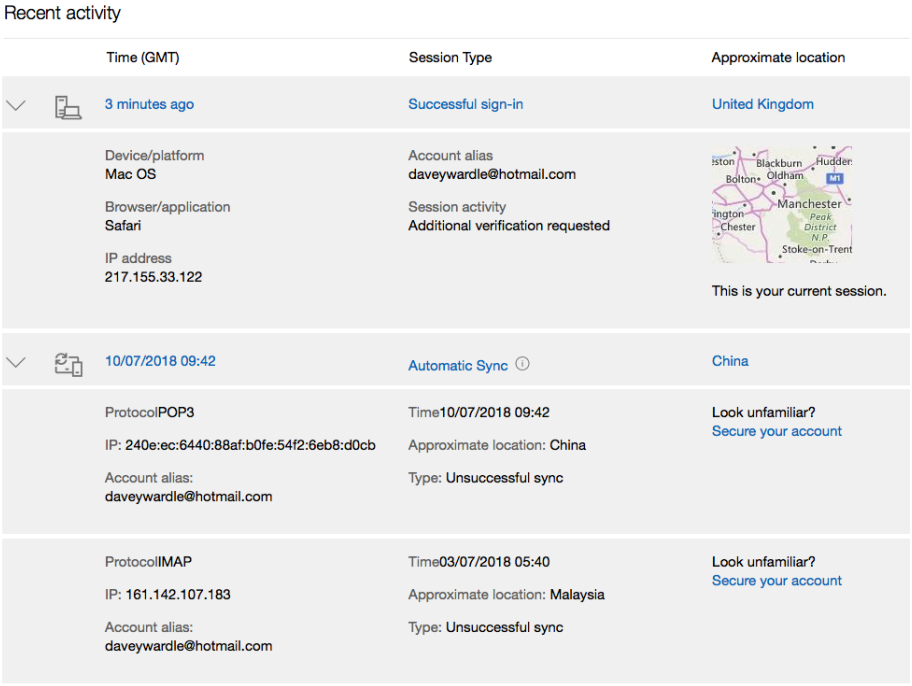


Figure A.10: Hotmail displayed information about unsuccessful syncs

Appendix B

Source code

This appendix contains all of the source code for tools that were developed during the course of the project.

Listing B.1: The Honey Identity generator source code

```
# generate.py

import argparse
import os
from steps import *


def main():
  parser = argparse.ArgumentParser(description='Create a full honeypot identity')
  parser.add_argument('-r', dest='region', help='specify region')
  parser.add_argument('-f', dest='file', help='json file to load')

  args = parser.parse_args()

  # Check being run with root privileges
  if os.geteuid() != 0:
    exit('You need to have root privileges to run this script.')

  # step 1. load user profile
  if (args.file == None):
    # either from feed
    # make it is easy to specify different regions.
    # must be correct format (eg united-states, india, germany, spain)
    if (args.region == None):
      region = 'england'
    else:
      region = args.region

    user = step01.load_and_preview(region)
  else:
    # @TODO: load pre-generated profile from JSON file
    parser.print_help()
    exit(0)

  # step 2. create isgproject.org email
  step02.create_email_account(user['account'], user['displayname'],
user['base_password'])

  # step 3. create isgproject.org website account
  step03.create_wordpress_account(user['account'], user['displayname'],
user['base_password'])

if __name__ == '__main__':
  main()
```

```
# step01.py

import json
import random
import urllib.request
import argparse

def load_and_preview(region):
  proceed = 'N'

  while proceed != 'Y':
    user = load_feed(region)
    print("[+] Generated the following user details:\n\n\
  Name: {0} {1} {2}\n\
  Date of Birth: {3}\n\
  Login: {4} / {5}\n".format(user['title'], user['firstname'], user['lastname'],
user['dob'], user['base_username'], user['base_password']))

    proceed = input("[+] Do you wish to proceed? [Y/N] ").upper()

  # save our new user to local JSON file
  filename = save_json(user)

  return user

def load_feed(region):
  # Load JSON feed from uinames
  response = urllib.request.urlopen('https://uinames.com/api/?ext&amount=1&region=' +
region.replace(" ", "+") + '&gender=random')
  str_response = response.read().decode('utf-8')
  j = json.loads(str_response)

  # Convert to our own format
  user = {
    'title' : j['title'].title(),
    'firstname' : j['name'],
    'lastname' : j['surname'],
    'gender' : j['gender'].title(),
    'dob' : j['birthday']['dmy'],
    'account' : j['name'].lower() + '.' + j['surname'].lower(),
    'displayname' : j['name'].title() + ' ' + j['surname'].title(),
  }

  # create a "base" username and password. these may need to altered depending on
service limitations/restrictions

  # Basic username = firstnamelastname. This can easily have a prefix (eg isg) or a
suffix (eg 1) to be unique.
  user['base_username'] = j['name'].lower() + j['surname'].lower()

  # load a random real password
  # this file is based on
https://github.com/danielmiessler/SecLists/blob/master/Passwords/darkweb2017-top10K.txt
  # with first 1000 removed along and then only 9 character long passwords.
  user['base_password'] = random.choice(open('passwords.txt').readlines()).replace('\n',
"")

  return user

def save_json(user):
  filename = user['account'] + '.json'
  with open('./json/' + filename, 'w') as outfile:
    json.dump(user, outfile)

  return filename
```

```
def main():
  parser = argparse.ArgumentParser(description='Create a honeypot identity only')
  parser.add_argument('-r', dest='region', help='specify region')
  args = parser.parse_args()

  # make it is easy to specify different regions.
  # must be correct format (eg united-states, india, germany, spain)
  if (args.region == None):
    region = 'england'
  else:
    region = args.region

  load_and_preview(region)

if __name__ == '__main__':
  main()
```

```
# step02.py

import pymysql
import argparse
import os
import datetime
from os import listdir
from os.path import isfile, join
import shutil
import imaplib
from base64 import b64encode
from hashlib import sha512
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

def generate_ssha512_password(p):
  p = str(p).strip()
  try:
    salt = os.urandom(8)
    pw = sha512(p.encode('utf-8'))
    pw.update(salt)
    return '{SSHA512}' + b64encode(pw.digest() + salt).decode()
  except ImportError as e:
    print('[-]' + e)

def create_email_account(user, display, password):

  try:
    cursor = pymysql.cursors.DictCursor
    connection =
pymysql.connect(host='localhost',user='vmailremote',password='<redacted>',db='vmail',cha
rset='utf8mb4',cursorclass=cursor)

    email = user + "@isgproject.org"
    hashed = generate_ssha512_password(password)

    # mailbox directory
    now = datetime.datetime.now()
    ts = str(now.year) + '.' + str(now.month).zfill(2) + '.' + str(now.day).zfill(2) +
'.' + str(now.hour).zfill(2) + '.' + str(now.minute).zfill(2) + '.' +
str(now.second).zfill(2)
    maildir = 'isgproject.org/' + user[:1] + '/' + user[1:2] + '/' + user[2:3] + '/' +
user + '-' + ts + '/'

    try:
      with connection.cursor() as cursor:
        # Create a new record
        sql = "INSERT INTO `mailbox` (`username`, `password`, `name`,
`storagebasedirectory`, `storagenode`, `maildir`, `quota`, `domain`, `active`,
`local_part`, `created`) VALUES (%s, %s, %s, '/var/vmail', 'vmail1', %s, '1024',
'isgproject.org', '1', %s, NOW())"
        cursor.execute(sql, (email, hashed, display, maildir, user))

        sql = "INSERT INTO `forwardings` (`address`, `forwarding`, `domain`,
`dest_domain`, `is_forwarding`) VALUES (%s, %s,'isgproject.org', 'isgproject.org', 1)"
        cursor.execute(sql, (email, email))

      # connection is not autocommit by default. So you must commit to save
      # your changes.
      connection.commit()

      print('[+] Created email account: ' + email)

      # creates maildir folder
      print('[+] Logging into new email account')
```

```
      test_email_login(email, password)

      # copy default emails
      default_inbox_emails(user, display, maildir)
      default_sent_emails(user, display, maildir)

      # set up BCC so all incoming email goes to special account
      with connection.cursor() as cursor:
        sql = "INSERT INTO `recipient_bcc_user` (`username`, `bcc_address`, `domain`,
`created`, `modified`) VALUES (%s, 'incoming@isgproject.org', 'isgproject.org', NOW(),
NOW())"
        cursor.execute(sql, (email))

      connection.commit()

    finally:
      connection.close()

  except pymysql.err.OperationalError as e:
    #print('[-] ERROR: Cannot connect to database. Please ensure IP has been
whitelisted')
    print('[-] ERROR: Cannot connect to database.')
    exit(0)

def test_email_login(email, password):
  # can't get chrome headless working on server so using deprecated phantomjs
  driver = webdriver.PhantomJS('phantomjs')
  driver.get("https://isgproject.org/mail/")

  assert "Roundcube Webmail :: Welcome to Roundcube Webmail" in driver.title

  element = driver.find_element_by_id('rcmloginuser')
  element.send_keys(email)

  element = driver.find_element_by_id('rcmloginpwd')
  element.send_keys(password)

  element = driver.find_element_by_id('rcmloginsubmit')
  element.click()

  # @TODO: Actually check that login has worked rather than assuming...
  assert "Roundcube Webmail :: Inbox" in driver.title

  driver.quit()

def default_inbox_emails(user, display, maildir):
  # this account has already been populated with modified enron emails
  baseDir = '/var/vmail/vmail1/isgproject.org/k/a/t/katie.davies-
2018.03.15.15.02.24/Maildir/.INBOX.ENRON/cur/'
  # copy + modify to this directory
  copyDir = '/var/vmail/vmail1/' + maildir + 'Maildir/new/'
  print('[+] Creating default inbox emails')
  copy_default_emails(user, display, baseDir, copyDir)

def default_sent_emails(user, display, maildir):
  # this account has already been populated with modified enron emails
  baseDir = '/var/vmail/vmail1/isgproject.org/k/a/t/katie.davies-
2018.03.15.15.02.24/Maildir/.Sent/cur/'
  # copy + modify to this directory
  copyDir = '/var/vmail/vmail1/' + maildir + 'Maildir/.Sent/new/'
  print('[+] Creating default sent emails')
  copy_default_emails(user, display, baseDir, copyDir)


def copy_default_emails(user, display, baseDir, copyDir):
```

```
  # get list of all files
  files = [f for f in listdir(baseDir) if isfile(join(baseDir, f))]

  filenum = 0

  # loop through all emails
  for file in files:
    # open + read file
    email = open(baseDir + file, 'r')
    emailLines = email.read().splitlines()
    email.close()

    output = []

    # loop through all lines in email + modify as necessary
    for line in emailLines:

      # get date that email was "sent" - needed for filename
      if line.startswith('Date: '):
        dateStr = line.replace('Date: ','')
        dateStr = dateStr.replace(' +0000', '')
        date = datetime.datetime.strptime(dateStr, '%a, %d %b %Y %H:%M:%S')

      else:
        # missed these dates when creating initial dataset
        line = line.replace('2002','2018')
        line = line.replace("'02","'18")
        line = line.replace('/02','/18')

        # replace all references of base user (Katie Davies) with new name
        names = display.split()
        line = line.replace('katie.davies',user)
        line = line.replace('Katie', names[0])
        line = line.replace('Davies', names[1])

      output.append(line)

    # basic numbering for filename
    filenum += 1
    m = 21500 + filenum
    p = 1800 + filenum

    filename = str(int(date.timestamp())) + '.M' + str(m) + 'P' + str(p) + '.ip-172-31-
24-31:2,S'

    # save modified email
    with open(copyDir + filename, 'w') as f:
      f.write('\n'.join(output))

    # change file owner + group to vmail
    shutil.chown(copyDir + filename, user='vmail', group='vmail')


def main():
  parser = argparse.ArgumentParser(description='Create an isgproject.org email account')
  parser.add_argument('-u', dest='user', help='specify username')
  parser.add_argument('-n', dest='display', help='specify display name')
  parser.add_argument('-p', dest='password', help='specify password')

  args = parser.parse_args()

  # Check being run with root privileges
  if os.geteuid() != 0:
    exit('You need to have root privileges to run this script.')
```

```
  if (args.user == None) | (args.display == None) | (args.password == None):
    parser.print_help()
    exit(0)
  else:
    user = args.user
    display = args.display
    password = args.password

  create_email_account(user, display, password)

if __name__ == '__main__':
  main()
```

```
# step03.py

import argparse
import os
import subprocess

def create_wordpress_account(user, display, password):
  # simple python wrapper for WP Cli command
  email = user + "@isgproject.org"
  names = display.split()
  os.chdir("/var/www/html")
  subprocess.call("wp user create {0} {1} --display_name='{2}' --user_pass={3} --
first_name='{4}' --last_name='{5}' --role='isg' --allow-root".format(user, email,
display, password, names[0], names[1]), shell=True)

def main():
  parser = argparse.ArgumentParser(description='Create an isgproject.org WordPress
account')
  parser.add_argument('-u', dest='user', help='specify username')
  parser.add_argument('-n', dest='display', help='specify display name')
  parser.add_argument('-p', dest='password', help='specify password')

  args = parser.parse_args()

  if (args.user == None) | (args.display == None) | (args.password == None):
    parser.print_help()
    exit(0)
  else:
    user = args.user
    display = args.display
    password = args.password

  create_wordpress_account(user, display, password)

if __name__ == '__main__':
  main()
```

Listing B.2: Source code for script to update email timestamps

```
# honey.timestamps.py

import pymysql
from os import rename,listdir
from os.path import isfile, join
import datetime
from dateutil.relativedelta import relativedelta
import shutil

def update_timestamps(path):

  files = [f for f in listdir(path) if isfile(join(path, f))]

  filenum = 0

  # loop through all files + update timestamp
  for file in files:
    email = open(path + file, 'r')
    emailLines = email.read().splitlines()
    email.close()

    output = []
    newDate = None

    for line in emailLines:
      # skip any emails that aren't "internal" ie message-id does not contain
JavaMail.evans@thyme
      #if line.startswith('Message-ID:') and "JavaMail.evans@thyme" not in line:
      # break

      # only update those with correct date format, more likely to be internal emails
      if line.startswith('Date: ') and "+0000" in line:
        dateStr = line.replace('Date: ','')
        dateStr = dateStr.replace(' +0000', '')
        date = datetime.datetime.strptime(dateStr, '%a, %d %b %Y %H:%M:%S')

        # add one day to the date
        newDate = date + relativedelta(days=1)

        # check if new date is in future
        if newDate > datetime.datetime.now():
          newDate = None
          break

        line = 'Date: ' + newDate.strftime('%a, %d %b %Y %H:%M:%S') + ' +0000'

      # only update those with correct date format, more likely to be internal emails
      if line.startswith('Date: ') and "+0100" in line:
        dateStr = line.replace('Date: ','')
        dateStr = dateStr.replace(' +0100', '')
        date = datetime.datetime.strptime(dateStr, '%a, %d %b %Y %H:%M:%S')

        # add one day to the date
        newDate = date + relativedelta(days=1)

        # check if new date is in future
        if newDate > datetime.datetime.now():
          newDate = None
          break

        line = 'Date: ' + newDate.strftime('%a, %d %b %Y %H:%M:%S') + ' +0100'
```

```
      output.append(line)

    if (newDate == None):
      continue

    # save file
    with open(path + file, 'w') as f:
      f.write('\n'.join(output))

    # rename file with new timestamp prefix
    newfile = str(int(newDate.timestamp())) + '.' + file.split('.', 1)[-1]
    rename(path + file, path + newfile)

def main():

  try:
    cursor = pymysql.cursors.DictCursor
    connection =
pymysql.connect(host='localhost',user='<redacted>',password='<redacted>',db='vmail',char
set='utf8mb4',cursorclass=cursor)

    try:
      with connection.cursor() as cursor:
        sql = "SELECT `maildir` FROM `mailbox` WHERE `username` NOT LIKE '%postmaster%'
AND `username` NOT LIKE '%incoming%'"
        cursor.execute(sql)
        results = cursor.fetchall()
        for result in results:
          path = '/var/vmail/vmail1/' + result['maildir'] + 'Maildir/cur/'
          update_timestamps(path)
          path = '/var/vmail/vmail1/' + result['maildir'] + 'Maildir/.Sent/cur/'
          update_timestamps(path)

    finally:
      connection.close()

  except pymysql.err.OperationalError as e:
    print('[-] ERROR: Cannot connect to database.')
    exit(0)

if __name__ == '__main__':
  main()
```

Listing B.3: Generic registration bot source code

```
# register.generic.py

import json
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
from selenium.webdriver.support.ui import WebDriverWait
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support import expected_conditions as EC

# init driver
def init_driver():
  options = webdriver.FirefoxOptions()
  options.set_headless(True)
  driver = webdriver.Firefox(options=options)
  driver.wait = WebDriverWait(driver, 5)
  return driver

# try to register
def register(driver, site, profile):
  register_attempt(driver, site, profile, 1)
  # take screenshot of final page regardless of success or not
  driver.save_screenshot('screenshot.png')
  return

def register_attempt(driver, site, profile, attempt):

  driver.get(site['url'])

  try:
    element = WebDriverWait(driver, 4).until(
      EC.presence_of_element_located((By.CSS_SELECTOR, site['wait_for']))
    )

    # click some things that need clicking
    for selector in site['pre_clicks']:
      element = driver.find_element_by_css_selector(selector)
      element.click()

    # loop through text fields and fill with profile data
    for selector, map in site['field_mappings'].items():
      element = driver.find_element_by_css_selector(selector)
      element.send_keys(profile[map])

    # select elements are special
    for selector, value in site['select_fields'].items():
      select = Select(driver.find_element_by_css_selector(selector))
      select.select_by_index(value)

    # loop through extra text fields and fill with hardcoded data
    for selector, value in site['extra_fields'].items():
      element = driver.find_element_by_css_selector(selector)
      element.send_keys(value)

    # click some more things that need clicking
    for selector in site['extra_clicks']:
      element = driver.find_element_by_css_selector(selector)
      element.click()
```

```
    # submit form
    form = driver.find_element_by_css_selector(site['form'])
    form.submit()

    # check if register successful
    # try to find success text within 4 seconds
    try :
      element = WebDriverWait(driver, 4).until(
        EC.presence_of_element_located((By.PARTIAL_LINK_TEXT, site['success']['text']))
      )
      print('[+] Success!')

      # click some more things that need clicking
      for selector in site['post_clicks']:
        element = driver.find_element_by_css_selector(selector)
        element.click()

      return
    except TimeoutException:
      if attempt == site['max_attempts']:
        # too many attempts, give up
        print('[+] Failed!')
        return
      else:
        # @TODO: try to work out what's gone wrong

        # @TODO: is username taken or incorrect format?

        # @TODO: is password incorrect format?

        # now let's try again
        attempt += 1
        register_attempt(driver, site, profile, attempt)
        return

  except TimeoutException:
    print('[-] ERROR: Could not load registration form')
    return

  return

def main():
  # load sites settings (JSON)
  with open('github.json', 'r') as f:
    site = json.load(f)

  # load user profile (JSON)
  with open('user.json', 'r') as f:
    profile = json.load(f)

  driver = init_driver()
  register(driver, site, profile)
  time.sleep(5)
  driver.quit()

if __name__ == '__main__':
  main()
```

Listing B.4: JSON configuration file for instagram.com

```
{
  "url" : "https://www.instagram.com/",
  "max_attempts" : 1,
  "form" : "form",
  "wait_for" : "input[name=emailOrPhone]",
  "pre_clicks" : [
    "input[name=emailOrPhone]"
  ],
  "field_mappings" : {
    "input[name=emailOrPhone]" : "email",
    "input[name=fullName]" : "displayname",
    "input[name=password]" : "base_password"
  },
  "select_fields" : {
  },
  "extra_fields" : {
  },
  "extra_clicks" : [
  ],
  "success" : {
    "url" : "",
    "text" : "Suggested for you"
  },
  "post_clicks" : [
  ]
}
```

Listing B.5: JSON configuration file for github.com

```
{
  "url" : "https://github.com/join",
  "max_attempts" : 1,
  "form" : "form",
  "wait_for" : "#user_password",
  "pre_clicks" : [
    "#user_login"
  ],
  "field_mappings" : {
    "#user_login" : "base_username",
    "#user_email" : "email",
    "#user_password" : "base_password"
  },
  "select_fields" : {
  },
  "extra_fields" : {
  },
  "extra_clicks" : [
    "#signup_button"
  ],
  "success" : {
    "url" : "",
    "text" : "Welcome to GitHub"
  },
  "post_clicks" : [
    ".setup-form button[type=submit]"
  ]
}
```

Listing B.6: Source code for automated credential check

```
# heartbeat.py

import json
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support import expected_conditions as EC

# init driver
def init_driver():
  options = webdriver.FirefoxOptions()
  #options.set_headless(True)
  driver = webdriver.Firefox(options=options)
  driver.wait = WebDriverWait(driver, 5)
  return driver

# try to login
def login(driver, site, profile):
  driver.get(site['url'])

  try:
    element = WebDriverWait(driver, 4).until(
      EC.presence_of_element_located((By.CSS_SELECTOR, site['wait_for']))
    )

    # click some things that need clicking
    for selector in site['pre_clicks']:
      element = driver.find_element_by_css_selector(selector)
      element.click()

    # loop through text fields and fill with profile data
    for selector, map in site['field_mappings'].items():
      element = driver.find_element_by_css_selector(selector)
      element.send_keys(profile[map])

    # submit form
    form = driver.find_element_by_css_selector(site['form'])
    form.submit()

    # check if login successful
    # try to find success text within 4 seconds
    try :
      # SUCCESS!
      element = WebDriverWait(driver, 4).until(
        EC.presence_of_element_located((By.PARTIAL_LINK_TEXT, site['success']['text']))
      )
      print('[+] Success!')

      return
    except TimeoutException:
      # OH NO!
      print('[+] Failed!')
      return

  except TimeoutException:
    print('[-] ERROR: Could not load login form')
    return
```

```
  return

def main():
  # load sites settings (JSON)
  with open('instagram-login.json', 'r') as f:
    site = json.load(f)

  # credentials hardcoded for proof-of-concept
  profile = {
    "username" : "emmafields5",
    "password" : "tacobell1"
  }

  driver = init_driver()
  login(driver, site, profile)

if __name__ == '__main__':
  main()
```

Listing B.7: JSON configuration file for instagram.com

```
{
  "url" : "https://www.instagram.com/accounts/login/",
  "form" : "form",
  "wait_for" : "input[name=username]",
  "pre_clicks" : [
    "input[name=username]"
  ],
  "field_mappings" : {
    "input[name=username]" : "username",
    "input[name=password]" : "password"
  },
  "success" : {
    "url" : "",
    "text" : "Suggested for you"
  }
}
```

Listing B.8: Script to scan email logs for logins

```
# check email logs
import re

logins = []
fails = []

def scan_log(log):

  # ignore any line containing these IPs (and email address)
  safe_ips = re.compile('209.85.|217.155.33.122|127.0.0.1|incoming@isgproject.org')

  with open("/var/log/dovecot/" + log + ".log", 'r') as f:
    for line in f:
      if re.search(safe_ips, line):
        continue

      # if contains this text then successful login
      if "login: Login: user=" in line:
        logins.append(line)

      # if contains this text then failed login
      if "auth failed" in line:
        fails.append(line)

def main():

  # scan the imap log
  scan_log('imap')

  # scan the pop3 logo
  scan_log('pop3')

  # print results
  if logins:
    print("*** SUCCESS LOGINS ***\n")
    print("".join(logins))

  if fails:
    print("*** FAILED LOGINS ***\n")
    print("".join(fails))

if __name__ == '__main__':
  main()
```

Listing B.9: Twilio webhook

```
<?php
  require 'vendor/autoload.php';
  use Mailgun\Mailgun;

  /**
  * This section ensures that Twilio gets a response.
  */
  header('Content-type: text/xml');
  echo '<?xml version="1.0" encoding="UTF-8"?>';
  echo '<Response></Response>'; // Left blank so no message sent in reply

  /**
  * This section actually sends the email.
  */
  # First, instantiate the SDK with your API credentials
  $mg = Mailgun::create( '<redacted>' );

  # Now, compose and send your message.
  $mg->messages()->send( 'mg.isgproject.org', [
    'from'    => "postmaster@mg.isgproject.org",
    'to'      => "david.wardle.2016@live.rhul.ac.uk,incoming@isgproject.org",
    'subject' => "New SMS message from: {$_REQUEST['From']}",
    'text'    => "From: {$_REQUEST['From']}\n\nMessage: {$_REQUEST['Body']}"
  ] );
```

Listing B.10: PHP honeytoken

```
<?php
  require 'vendor/autoload.php';
  use Mailgun\Mailgun;

        // Get Token ID + info. on visit
  $identifier = $_GET['identifier'];
  $userAgent = $_SERVER['HTTP_USER_AGENT'];
  $srcIP = $_SERVER['REMOTE_ADDR'];

  /**
  * This section actually sends the email.
  */

  # First, instantiate the SDK with your API credentials
  $mg = Mailgun::create( '<redacted>' );

  # Now, compose and send your message.
  $mg->messages()->send( 'mg.isgproject.org', [
    'from'    => "postmaster@mg.isgproject.org",
    'to'      => "david.wardle.2016@live.rhul.ac.uk,incoming@isgproject.org",
    'subject' => "Homebrew token triggered",
    'text'    => "Identifier: $identifier\nSource IP: $srcIP\nUser agent: $userAgent"
  ] );

  // Redirect user to 404 error
  http_response_code( 404 );
  header( 'Location: /404/' );
  exit;
```

Listing B.11: WordPress hooks

```
// logs the incorrect password used in failed login attempt
// to be used in conjunction with sucuri plugin
function wp_authenticate_log( $username, $password ) {
  // make sure password not blank
  if ( ! empty( $password ) ) {
    // and valid user
    $user = get_user_by('login', $username );
    if ( ! $user || ! wp_check_password( $password, $user->user_pass, $user->ID ) ) {
      error_log( "FAILED LOGIN: $username $password " . $_SERVER['REMOTE_ADDR'] );

      // Special canary tokens - fake passwords for an administrator user
      $tokens = [ 'YW1lbGlhLmN!', 'amFtaWUuYmFrZX!', 'Z3JhY2llLmxld$', 'ZHlsYW4ud2F0c$'
];
      if ( $username === 'administrator' && in_array( $password, $tokens ) ) {
        wp_mail(
          [ "david.wardle.2016@live.rhul.ac.uk", "incoming@isgproject.org" ],
          "Homebrew token triggered",
          "Identifier: $password\nUse Sucuri for more info!"
        );
      }
    }
  }
}
add_action( 'wp_authenticate' , __NAMESPACE__ . '\\wp_authenticate_log', 30, 2 );

// if user is logged in, log page view
function loggedin_user_pageview() {
  $user = wp_get_current_user();
  if ( $user->exists() && ! current_user_can( 'manage_options' ) ) {
    error_log( "PAGEVIEW: {$user->user_login} viewed " . get_permalink() );
  }
}
add_action( 'template_redirect',  __NAMESPACE__ . '\\loggedin_user_pageview' );
```

Listing B.12: Roundcube plugin

```
<?php
/**
 * Login_notify
 *
 * Sends an email alert whenever a user logs into Roundcube
 *
 * @version 1.0
 * @author davewardle
 */

class login_notify extends rcube_plugin {

  public function init() {
    $this->add_hook( 'login_after', [
      $this,
      'login_after'
    ] );
  }

  function login_after() {
    $rcmail = rcmail::get_instance();
    $user = $rcmail->user;
    $username = $user->data['username'];
    $user_agent = $_SERVER['HTTP_USER_AGENT'];
    $remoteip = rcube_utils::remote_ip();

    $safe_ips = [ '217.155.33.122' ];

    if ( ! in_array( $remoteip, $safe_ips ) ) {

      $mg = Mailgun\Mailgun::create( '<redacted>' );
      $mg->messages()->send( 'mg.isgproject.org', [
        'from'    => "postmaster@mg.isgproject.org",
        'to'      => "david.wardle.2016@live.rhul.ac.uk,incoming@isgproject.org",
        'subject' => "New Roundcube login",
        'text'    => "User: $username\nSource IP: $remoteip\nUser agent: $user_agent"
      ] );

    }

  }

}
```

Appendix C

Paste files

All of the pastes created for the experiment are displayed in full in this Appendix.

Listing C.1: Paste 1

```
INSERT INTO `wp_users` VALUES
(1,'administrator','$P$BVLxjpxMno/PYMhgCO.n9YTlcRnY1b1','administrator','admin@isgprojec
t.org','','2016-07-02
13:29:20','',0,'administrator'),(2,'dylan.watson','$P$BqYVYm9BcK./kRioV/1ETc2jmksex5/','
dylan-watson','dylan.watson@isgproject.org','','2016-07-22 14:49:27','',0,'Dylan
Watson'),(5,'emma.fields','$P$BAs3bhdChkwidwb6Jpz.isOlZvxHuJ.','emma-
fields','emma.fields@isgproject.org','','2017-06-19 15:01:17','',0,'Emma
Fields'),(6,'amelia.coleman','$P$BPShuM7em.wkfQqQtj6rRzC0bUB3WC/','amelia-
coleman','amelia.coleman@isgproject.org','','2017-06-20 17:03:30','',0,'Amelia
Coleman'),(7,'charles.sutton','$P$BkRB6Q9iOp/9PWSClv./s2amk4UxVO.','charles-
sutton','charles.sutton@isgproject.org','','2017-06-20 17:07:03','',0,'Charles
Sutton'),(8,'jamie.baker','$P$BODbm9/h1nM69c/1YYUr5PpvbPnrGH/','jamie-
baker','jamie.baker@isgproject.org','','2017-10-13 12:31:45','',0,'Jamie
Baker'),(9,'brian.peterson','$P$BZQ4L/TxIEKSMPoLKrh3mChWuqZGHs1','brian-
peterson','brian.peterson@isgproject.org','','2018-03-16 14:09:22','',0,'Brian
Peterson'),(10,'gracie.lewis','$P$B8tZ9T1RYNkWCVlUXNxzAWjeuwOuHL/','gracie-
lewis','gracie.lewis@isgproject.org','','2018-03-16 14:46:54','',0,'Gracie
Lewis'),(11,'charlotte.baker','$P$Bu3zr0Ly3BkkLAuThwKiY5EiDmjz8h/','charlotte-
baker','charlotte.baker@isgproject.org','','2018-04-07 14:48:36','',0,'Charlotte
Baker'),(12,'alex.baker','$P$BPhdPZzG/T9xEHPXoAilB1GHBpa5PD.','alex-
baker','alex.baker@isgproject.org','','2018-04-22 17:49:44','',0,'Alex
Baker'),(13,'isabel.griffiths','$P$BIb8sSAJklSOvOlXzB6AOs5aQS3N0o0','isabel-
griffiths','isabel.griffiths@isgproject.org','','2018-04-22 18:31:54','',0,'Isabel
Griffiths');
```

Listing C.2: Paste 2

```
+----+-----------------+-----------------------------------------------------------------
+------------------+---------------------------------+----------+---------------------+-
--------------------+-------------+------------------+
| ID | user_login      | user_pass                                                        |
user_nicename    | user_email                      | user_url | user_registered     |
user_activation_key | user_status | display_name     |
+----+-----------------+-----------------------------------------------------------------
+------------------+---------------------------------+----------+---------------------+-
--------------------+-------------+------------------+
|  1 | administrator   | $1$f30XNqFl$6wzPLXCgS.GSghapYLtrS1                               |
administrator    | administrator@isgproject.org    |          | 2014-07-02 13:29:20 |
|           0 | administrator    |
|  4 | amelia.coleman  | $1$7TVrXPDU$KCZ9c.QsmTahZ2oMmgJ4v.                               |
amelia-coleman   | amelia.coleman@isgproject.org   |          | 2015-02-13 17:03:30 |
|           0 | Amelia Coleman   |
|  5 | emma.fields     | $1$mH0scC7I$bsr54s/uk.e1JYnyz4v/d/                               |
emma-fields      | emma.fields@isgproject.org      |          | 2015-06-19 15:01:17 |
|           0 | Emma Fields      |
|  7 | charles.sutton  | $1$e2Y9xCDk$Yo8W8gMhFzZShu7mFzq9v0                               |
charles-sutton   | charles.sutton@isgproject.org   |          | 2015-06-20 17:07:03 |
|           0 | Charles Sutton   |
|  8 | jamie.baker     | $1$m4FfmGHK$T1/uPKYn/pPS1UmmHEubk0                               |
jamie-baker      | jamie.baker@isgproject.org      |          | 2015-06-20 17:31:45 |
|           0 | Jamie Baker      |
|  9 | brian.peterson  | $1$IEPKoBZ6$VwNyGjb6gDHG0.iI1dX86.                               |
brian-peterson   | brian.peterson@isgproject.org   |          | 2015-07-02 14:09:22 |
|           0 | Brian Peterson   |
| 10 | gracie.lewis    | $1$xQ89n3Nh$91YrkA2sKaBbZelM/QR6t1                               |
gracie-lewis     | gracie.lewis@isgproject.org     |          | 2015-07-02 14:16:54 |
|           0 | Gracie Lewis     |
| 11 | charlotte.baker | $1$AQBsjBCU$eyIo/ZvOUGFTDhQR7SMWE.                               |
charlotte-baker  | charlotte.baker@isgproject.org  |          | 2015-07-22 14:38:36 |
|           0 | Charlotte Baker  |
| 12 | dylan.watson    | $1$DT92XJOr$7.DQ0Pev.PCAuTMSNCkqN1                               |
dylan-watson     | dylan.watson@isgproject.org     |          | 2015-07-22 14:44:27 |
|           0 | Dylan Watson     |
| 13 | alex.baker      | $1$5U42cogK$bnAwUxb//viMaNrhrj5XQ.                               |
alex-baker       | alex.baker@isgproject.org       |          | 2015-07-22 14:56:44 |
|           0 | Alex Baker       |
| 14 | isabel.griffiths | $1$aTFVLwuD$vPVB774jrJMUlT3EJn7kK1                              |
isabel-griffiths | isabel.griffiths@isgproject.org |          | 2015-08-14 11:21:54 |
|           0 | Isabel Griffiths |
+----+-----------------+-----------------------------------------------------------------
+------------------+---------------------------------+----------+---------------------+-
--------------------+-------------+------------------+
```

Listing C.3: Paste 3

```
                          ::::::::::HACKED BY 5N1P3R::::::::::::::

#[ Database: isg_website                    ]
#[ Table   : admin_user                     ]
#[ Columns : admin_user_id,username,password ]

1,admin,0c148356ead38c15a8d1760fc9e631d6

#[ Database: isg_website                    ]
#[ Table   : user                           ]
#[ Columns : user_id,username,password,email ]

1,dylan,acfee3da5c432c1c7021f972eda60be7,dylan.watson@isgrpoject.org
2,lewis,2ba62da5c89bbc5965cb0434ffb62b90,lewis.clarke@isgproject.org
3,brian,79b5afeffc388a330c59aee934bc9163,brian.peterson@isgproject.org
4,joe,2b49a8c86865524892f0aae2e4309ec8,joe.cooper@isgproject.org
5,mo,3d3d4123cc9f79e4005b8ef7c78ac8e4,mo.johnson@isgproject.org
6,gracie,f5b677ff789127c17cbb3114903794c4,gracie.lewis@isgproject.org
7,alex,0fe6677f5901dfe64849674e15303ba0,alex.baker@isgproject.org
8,amy,dded7bc3d6198401c47a2d67c3f49cba,amy.lloyd@isgproject.org
```

Listing C.4: Paste 4

```
+----+-----------------+-----------------------------------------------------------------
+------------------+---------------------------------+----------+---------------------+-
--------------------+-------------+------------------+
| ID | user_login      | user_pass                                                       |
user_nicename    | user_email                      | user_url | user_registered     |
user_activation_key | user_status | display_name     |
+----+-----------------+-----------------------------------------------------------------
+------------------+---------------------------------+----------+---------------------+-
--------------------+-------------+------------------+
|  1 | administrator   | $1$f30XNqFl$6wzPLYCgS.GSghapYLtrS1                              |
administrator    | administrator@isgproject.org    |          | 2014-07-02 13:29:20 |
|           0 | administrator    |
|  3 | jamie.baker     | $1$Js64PJ20$WP7YrgU5eT1ikELUaMXlh.                              |
jamie-baker      | jamie.baker@isgproject.org      |          | 2015-02-13 17:31:45 |
|           0 | Jamie Baker      |
|  5 | emma.fields     | $1$mH0scC7I$bsr54a/uk.e1JYnyz4v/d/                              |
emma-fields      | emma.fields@isgproject.org      |          | 2015-06-19 15:01:17 |
|           0 | Emma Fields      |
|  7 | charles.sutton  | $1$e2Y9xCDk$Yo8W8fMhFzZShu7mFzq9v0                              |
charles-sutton   | charles.sutton@isgproject.org   |          | 2015-06-20 17:07:03 |
|           0 | Charles Sutton   |
|  9 | brian.peterson  | $1$IEPKoBZ6$VqNyGjb6gDHG0.iI1dX86.                              |
brian-peterson   | brian.peterson@isgproject.org   |          | 2015-03-02 14:09:22 |
|           0 | Brian Peterson   |
| 10 | gracie.lewis    | $1$xQ89n3Nh$91TrkA2sKaBbZelM/QR6t1                              |
gracie-lewis     | gracie.lewis@isgproject.org     |          | 2015-03-02 14:16:54 |
|           0 | Gracie Lewis     |
| 11 | charlotte.baker | $1$YjznQIbb$hNAb4gV8ck.kdbQAL1cbZ/                              |
charlotte-baker  | charlotte.baker@isgproject.org  |          | 2015-04-22 14:38:36 |
|           0 | Charlotte Baker  |
| 12 | dylan.watson    | $1$DT92XJOr$7.SQ0Pev.PCAuTMSNCkqN1                              |
dylan-watson     | dylan.watson@isgproject.org     |          | 2015-04-22 14:44:27 |
|           0 | Dylan Watson     |
| 13 | alex.baker      | $1$5U42cogK$bnWwUxb//viMaNrhrj5XQ.                              |
alex-baker       | alex.baker@isgproject.org       |          | 2015-04-22 14:56:44 |
|           0 | Alex Baker       |
| 14 | isabel.griffiths | $1$aTFVLwuD$vPCB774jrJMUlT3EJn7kK1                             |
isabel-griffiths | isabel.griffiths@isgproject.org |          | 2015-08-14 11:21:54 |
|           0 | Isabel Griffiths |
+----+-----------------+-----------------------------------------------------------------
+------------------+---------------------------------+----------+---------------------+-
--------------------+-------------+------------------+
```

Listing C.5: Paste 5

```
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 2.9.4
          Sponsored by Sucuri - https://sucuri.net
      @_WPScan_, @ethicalhack3r, @erwan_lr, @_FireFart_
_______________________________________________________________


[i] It seems like you have not updated the database for some time
[i] Last database update: 2018-07-03
[?] Do you want to update now? [Y]es  [N]o  [A]bort update, default: [N] > y
[i] Updating the Database ...
[i] Update completed
[i] The remote host tried to redirect to: https://isgproject.org/
[?] Do you want follow the redirection ? [Y]es [N]o [A]bort, default: [N] >Y
[+] URL: https://isgproject.org/
[+] Started: Mon Jul 30 16:38:49 2018

[+] Interesting header: LINK: <https://isgproject.org/>; rel=shortlink
[+] Interesting header: SERVER: nginx
[!] SQL export file found: https://isgproject.org/backup.sql
[+] This site has 'Must Use Plugins' (http://codex.wordpress.org/Must_Use_Plugins)
[+] XML-RPC Interface available under: https://isgproject.org/xmlrpc.php   [HTTP 405]

[+] Enumerating WordPress version ...

[i] WordPress version can not be detected

[+] WordPress theme in use: isgproject

[+] Name: isgproject
 |  Location: https://isgproject.org/wp-content/themes/isgproject/
 |  Style URL: https://isgproject.org/wp-content/themes/isgproject/style.css
 |  Referenced style.css: wp-content/themes/isgproject/assets/css/style.css
 |  Theme Name: ISG Project
 |  Theme URI: https://isgproject.org/
 |  Description: Bespoke WordPress theme for ISG Project

[+] Enumerating plugins from passive detection ...
[+] No plugins found passively

[+] Finished: Mon Jul 30 16:39:37 2018
[+] Elapsed time: 00:00:47
[+] Requests made: 397
[+] Memory used: 16.086 MB
```

Listing C.6: Paste 6

```
https://www.dropbox.com/sh/kqcznxcntx287ff/AAA78iHff8c6iQX-6rJdPGCSa?dl=0
https://www.dropbox.com/sh/dj54xx9sf79odzb/AAC2gAglYRgtloiQ5NZb2cA7a?dl=0
https://www.dropbox.com/sh/xi9td61gair0dem/AADf1jjph6VymNRfOhRIQf1Ia?dl=0
https://www.dropbox.com/sh/gjo2anm0lzqucwm/AABcJT5uVAXTOtkTKL2Vv6Cwa?dl=0
https://www.dropbox.com/sh/7zxm49mr6d8jn83/AADVRuKmBrSOnjXWxxYz2lTYa?dl=0
```

Listing C.7: Paste 7

```
Dropbox.com
alex.baker@isgproject.org:thomas123
```

Listing C.8: Paste 8

```
Dropbox.com
isgkatiedavies@gmail.com:Lacrosse2018
```