Detecting semantic social engineering attacks with the weakest link: Implementation and empirical evaluation of a human-as-a-security-sensor framework

Ryan Heartfield, George Loukas

Computing and Information Systems
University of Greenwich, UK
E: r.j.heartfield@gre.ac.uk

Abstract

The notion that the human user is the weakest link in information security has been strongly, and, we argue, rightly contested in recent years. Here, we take a step further showing that the human user can in fact be the strongest link for detecting attacks that involve deception, such as application masquerading, spearphishing, WiFi evil twin and other types of semantic social engineering. Towards this direction, we have developed a human-as-a-security-sensor framework and a practical implementation in the form of Cogni-Sense, a Microsoft Windows prototype application, designed to allow and encourage users to actively detect and report semantic social engineering attacks against them. Experimental evaluation with 26 users of different profiles running Cogni-Sense on their personal computers for a period of 45 days has shown that human sensors can consistently outperform technical security systems. Making use of a machine learning based approach, we also show that the reliability of each report, and consequently the performance of each human sensor, can be predicted in a meaningful and practical manner. In an organisation that employs a human-as-a-security-sensor implementation, such as Cogni-Sense, an attack is considered to have been detected if at least one user has reported it. In our evaluation, a small organisation consisting only of the 26 participants of the experiment would have exhibited a missed detection rate below 10%, down from 81% if only technical security systems had been used. The results strongly point towards the need to actively involve the user not only in prevention through cyber hygiene and user-centric security design, but also in active cyber threat detection and reporting.

1. Introduction

In information security, it is often posited that the human user is the "weakest link" [1, 2, 3] because even the strongest technical protection systems can be bypassed if an attacker successfully manipulates the user into divulging a password, opening a malicious email attachment or visiting a compromised website. Deception-based threats targeting the user are both a pervasive and existential threat to computer systems, because on any system the user-computer interface is always vulnerable to abuse by authorised users, with or without their knowledge. It is well known that users are key for configuring security mechanisms. Here, we show that they can also be better than technical security measures at detecting threats, especially when a threat is based on deception of the user rather than exploitation of a specific technical flaw. Our scope is on semantic attacks defined in [4] as "the manipulation of user-computer interfacing with the purpose to breach a computer system's information security through user deception". Table 1 describes the variety of such attacks in the wild today, with an indicative selection of 25 different types, and Figure 1 illustrates a sample of these attacks to highlight their deception vectors visually.

With such extremes in diversity, there is an increasing need to develop defences that can operate holistically across the wider attack space. As semantic attacks target the user-computer interface, it is particular difficult for technical defences to identify them. This is because attacks primarily employ cosmetic or behavioural deception vectors which typically leave very few technical traces for a computer program to analyse. Technical defences do exist and some have been shown to work very well for certain threats (e.g., phishing emails), but can be very poor at detecting conceptually almost identical attacks on different platforms (e.g phishing instant messaging). Deploying a different technical defence for each of the semantic attacks that an organisation may be targeted by is, of course, impractical. For technical defence systems to stand a chance in detecting a wide range of semantic attacks, defence mechanisms would require the ability to interpret both visual and behavioural information, contextually and across multiple user-interface platforms. We argue that this makes the human user an attractive candidate for actively participating in detection.

2. The Human-as-a-Security-Sensor paradigm

From a system standpoint, humans are autonomous, multisensory systems, equipped with the ability to pro-

Attack Pseudonym	Description
Phishing	Attempt to obtain access to sensitive information by disguising as a trustworthy entity in an electronic communication
Spear phishing	Phishing attack designed to target a specific person/organisation/system
QRishing	Phishing attack using quick response (QR) codes (to disguise a link to malicious website or file)
Blue Snarfing	Phishing attack enticing a user to install a malicious file allowing access to the users device via the bluetooth protocol
Smishing	Phishing attack on simple message service (SMS) in mobile devices
DriveyBy download	Implanting a malicious file on a vulnerable web platform
Waterhole	Highly targeted DriveBy download attack, implanting on specific websites target visits
File Masquerading	Disguising a malicious file to appear as a legitimate file type
Multimedia Masquerading	Disguising a malicious application appear as multimedia (e.g., video)
GUI Confusion	A mobile application confusing users by impersonating another (e.g., banking app) to obtain sensitive information
Visual SSL spoofing	Using fake SSL verification logos or GUI components to deceive users into thinking they are on a secure website
Scareware	Malicious program tricking a user into buying/downloading unnecessary often malicious software, such as fake antivirus
Malvertisement	An online advertisement that incorporates or installs malware.
WiFi Evil Twin	A fraudulent WiFi access point that often spoofs other nearby access points that appears to be legitimate.
Trojan Horse	Type of malware that is often disguised as legitimate software, such as a game that also acts as a key-logger.
Self XSS	Operates by tricking users into copying and pasting malicious content into their browsers' web developer console.
Typosquatting	Form of cybersquatting relying on typographical errors made by users when inputting an address into a browser.
Tabnabbing	A type of phishing where a website changes to impersonate popular websites
Sharebaiting	Enticing content posted on social media which users share on their profile, spreading fake apps, news and phishing URLs
Click Jacking	Concealing hyperlinks beneath legitimate click-able content, causing the user to commit hidden actions
Cursor Jacking	Variation of clickjacking, users are deceived by a custom cursor image where the pointer is displayed with an offset.
Spamdexing	Manipulation of search engine indexes where a website repeats unrelated phrases to manipulate relevance or prominence
Torrent Poisoning	Intentionally sharing corrupt data and malware with misleading file names using the BitTorrent protocol
Fake App	Variation of trojan horse, rogueware, scareware on mobile devices where a malicious app masquerades as a legitimate one
Fake Plugin	Malicious social media plugin typically spread by automatically re-posting a fake video to victims' and friends profile

Table 1: Examples of semantic attacks observed in the wild

duce inferential output data based on multiple experiential and environmental input data. This capability means that humans (as physical sensors) are often well placed to provide information in various contexts where technical systems alone are not adequate. Taking into consideration the user-computer interface, which is the primary interface targeted by semantic attacks, it is therefore implicit that human sensors are able to perform detection across all possible platforms and systems in which semantic attacks may be deployed.

The aim of the Human-as-a-Security-Sensor (HaaSS) paradigm, where user reports are encouraged and taken into account so as to strengthen an organisation's cyber situational awareness [5], is not to replace technical security systems, but to complement or augment existing technical means in detecting and mitigating certain semantic attacks by leveraging human sensing capacity and experience. In fact, despite many advancements in technical defences against semantic attacks, it has long been realised in the security industry that the users need to be at the core of any system's security design [6, 7, 8, 9, 10, 11, 12]. Our aim is to progress a step further, by empowering users to directly contribute to the security of themselves, their organisation and even the wider community. For the detection of semantic social engineering attacks, users require an interface that provides them with functionality to report suspicious or anomalous activity that uses deceptive attack vectors rather than technical exploitation; for which the human user often is a more accurate sensor than an organisation's technical security systems.

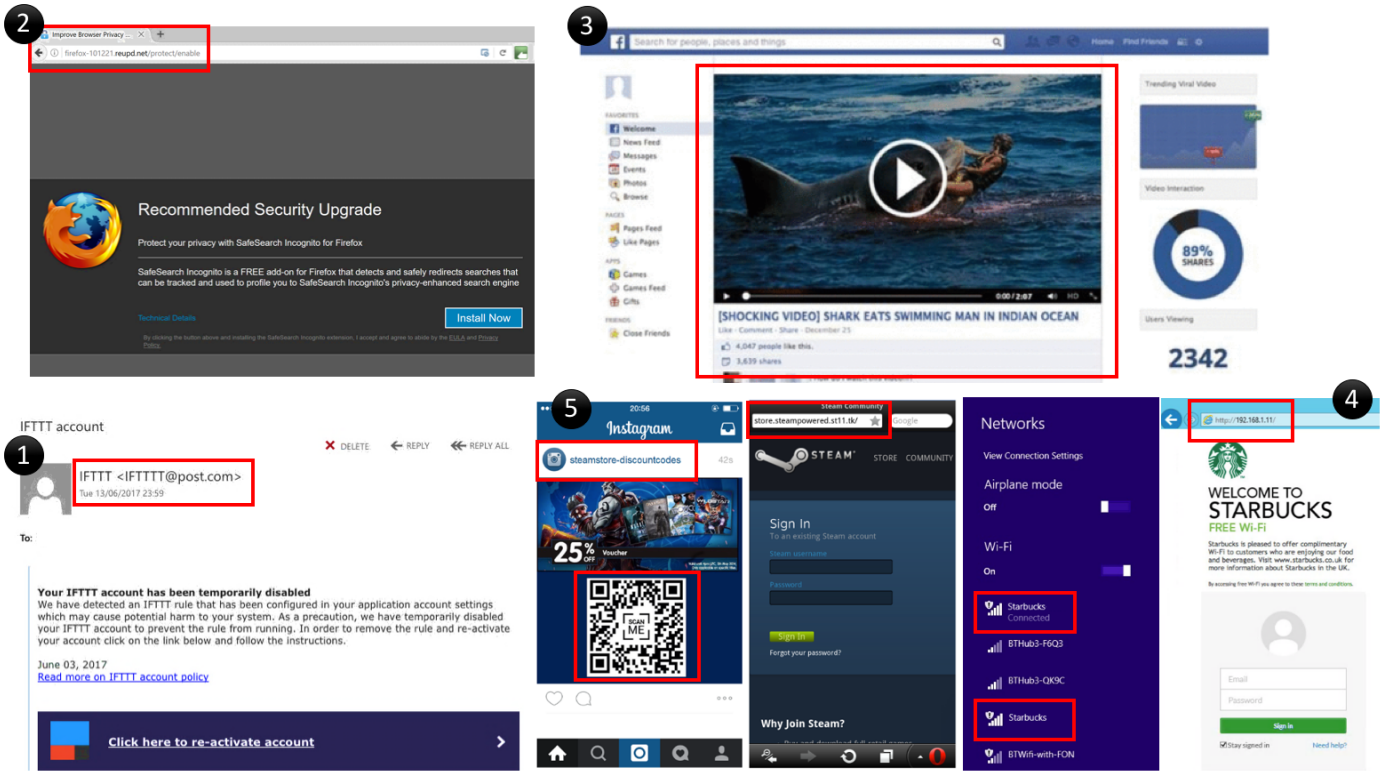
More specifically, HaaSS can be used to actively enhance existing technical defence mechanisms by combining telemetry generated by user threat detection with threats flagged by technical defence platforms; helping confirm the existence and highlight the extent of the threat, or crucially, for detecting semantic attacks that have been largely undetectable by technical systems. In this capacity, HaaSS allows for proactive and preemptive detection of semantic attacks by positioning (and empowering) the user as a platform security sensor in order to identify and report suspected attacks in real-time.

For the function of HaaSS in the context of semantic attacks, as well as the wider computer security threat space, we propose the following definition:

Human-as-a-Security-Sensor. The paradigm of leveraging the ability of human users to act as sensors that can detect and report information security threats.

Over the last few years, the concept of the human as a sensor has seen increasing application for the detection of threats and adverse conditions in physical space, for instance to detect unfolding emergencies [13] and noise pollution [14], and monitor water availability [15]. These successes in physical space have served as motivation for applying and evaluating the concept in detection of threats in cyber space too, especially for semantic social engineering attacks, where technical security mechanisms have traditionally been limited in scope or accuracy. Hence, our goal here is to design a complete HaaSS framework, apply it in the implementation of a prototype HaaSS system, and evaluate it in a real-world context on variety of novel semantic social engineering attacks and against existing technical security mechanisms.

Figure 1: Example semantic attacks: 1) Phishing e-mail 2) Typosquatting 3) Social media multimedia masquerading 4) WiFi Evil Twin 5) QRishing



2.1. Related Work

Initial work in human sensor networks, instead of utilising direct human sensor information (e.g., a social media post), has employed the metadata generated by humans (e.g., mobile phone location data) for modelling and developing situational awareness in natural disasters [16, 17]. However, in [18], Wang et al. have explored the concept of human sensor networks and associated data formed through social networks. The researchers have suggested analysing the reliability of human sensor reporting based on the modeling of networked human sources who report observations (in the physical world), the filtering of unreliable report data, and the evaluation of an algorithm that can distinguish between reliable and unreliable data in the real-world. Developing a derived maximum likelihood estimation model, Wang et al. have utilised data from Twitter to evaluate their model's ability to accurately determine the corrections of reports from humans sensors.

In the context or computer security, Stembert et al. [19] have recently proposed combining a reporting function with blocking and warning of suspicious emails and the provision of educative tips, so as to harness the intelligence of expert and novice users in detecting email phishing attacks in a corporate environment. Initial experimental results of their mock-up have been encouraging for the applicability of the HaaSS concept in detecting phishing email threats. Another recent example was demonstrated by Malisa et al. [20], who developed an accurate and automated mobile application spoofing detection system by leveraging user visual similarity perception; integrating the human sensing data collected as a component of the technical system's detection decision making. In the case of both approaches, to accurately detect the specific semantic attack, each defence system explicitly relies on user expertise and knowledge. However, each system is also limited in the same way as conventional defence systems, in that they are attack-specific.

In the commercial space, security companies PhishMe and Wombat Security have developed security products that specifically integrate reporting software into organisations' email and web platforms in order for users to report suspected phishing attacks, thereby providing a form of human sensor detection telemetry that is used by an organisation to respond to credible threats. *PhishMe Reporter* [10] provides users with a simple email client add-in (across a limited range of e-mail software applications only) which allows users to click on a reporting button that forwards suspicious emails to an organisation's internal security team. When combined with *PhishMe Triage* [21], user reported emails are forwarded to a phishing-specific incident response platform that allows for automatic analysis, prioritisation and response (through integration with other security platforms such as anti-malware) by an organisation's security operations centre (SOC). Wombat Security's *PhishAlarm* and *PhishAlarm Analyser* [11] provide conceptually similar features to *PhishMe Reporter*

and *PhishMe Triage*, respectively, but have extended user reporting options to include SMS and USB phishing attacks, as well as implementing a machine learning approach to provide a real-time ranking of reported emails' predicted threat potential.

At organisational level, there are examples of in-house security teams which have begun to set up platforms and information security teams with the purpose to draw on their user base for detection of suspected threats. For instance, the Cyber Emergency Response Team at Oxford University have established an information security policy that employs users as sensors of suspected phishing threats through a reporting portal that encourages students and staff to report suspected phishing attacks on websites and email [8].

Online open-source community platforms have demonstrated the utility of human sensors of phishing threats in a crowd-sourcing context by allowing anonymous user reporting of suspected phishing emails and websites. For example, the well-known online phishing repository *PhishTank* [22] provides Internet users with a platform in which to anonymously report and review suspected phishing websites and emails, with report classification (e.g., attack / no attack) performed by means of majority vote amongst PhishTank community users. By comparison, another open-source platform called Millersmiles [23] limits online users reporting to a simple form that captures suspected phishing emails.

At the time of writing, however, existing commercial, organisational and community-led crowd-sourcing human sensor platforms remain relatively immature, due to the lack of a formal framework on which to base the implementation of HaaSS systems, and almost exclusive focus on phishing attacks; which addresses only a small portion of the problem space. Furthermore, commercial offerings are currently designed for organisations, and are therefore impractical for personal users to engage with. Perhaps most important is the absence of a means to measure sensor reliability which is based on generic susceptibility indicators, applicable across a wide range of semantic attacks vectors. As a result, it remains unclear how performance metrics generated solely from phishing reports can provide a lasting or accurate measure of human sensor detection efficacy when faced with other deception-based threats.

The taxonomy presented in [4] has simplified the semantic attack problem space into a set of fixed classification criteria, which was then utilised to compare attacks' functional constructs against the existing state-of-the-art for defence. The approach elicited key defence techniques, which in combination, were shown to address the full attack space, namely, user awareness, machine learning and sandboxing. However, up til this point, no approach to defence had combined these defence elements into a single architecture for detecting and preventing a wide range of semantic attacks.

In [5], Heartfield et al. have demonstrated how user susceptibility to semantic attacks can be reliably predicted by utilising a user's profile features which can be measured ethically, automatically and in real-time. Conducting a large scale study of over four thousand participants and a smaller study of over 315 participants, they developed machine learning models for predicting a user's detection efficacy in reporting suspected threats across a range of different semantic attacks. Crucially, the approach allowed for the models and feature-set to be integrated directly into a technical system as a tool for measuring the reliability of HaaSS reports for classification and prioritisation. Expanding on this preliminary work, in [24], the same authors conducted a preliminary laboratory-based study which integrated the machine learning models into an early prototype HaaSS system. As per the conclusions made in [5], the results of the initial study demonstrated the reliability of a users attack reporting (and therefore attack detection efficacy) depends on the human sensor's activity profile, as defined by characteristics including the amount and type of security training, familiarity with each system, frequency and duration of system access etc.

However, before building a system that depends extensively on a particular type of sensor (and the human sensor is no exception), one needs to be able to measure or estimate its overall reliability in a representative environment. In the case of HaaSS, and forming the basis of this work, this means expanding upon theoretical observations typically made under survey, questionnaire or laboratory conditions (which often limit the ability to produce lasting empirical conclusions about a security system's practical usefulness [25]), by testing the concept under real-world conditions.

Here, we comprehensively evaluate the HaaSS concept for detecting semantic attacks by designing a technical HaaSS framework, and developing, deploying and testing a prototype HaaSS system in a real-world context and across a wide range of semantic attacks. Initial work presented in [24] has sought to test users' ability to detect semantic attacks as HaaSS sensors within an interactive laboratory environment using a specifically developed reporting mechanism. In this early work, the primary goal was to evaluate whether users would utilise effectively a reporting function to capture suspected threats within a role-play scenario; comparing their simulation performance against a number of existing technical defences for the same attacks.

In this work, we evaluate the HaaSS concept end-to-end, from initial HaaSS sensor attack detection to final report classification by a HaaSS platform's security operations analyst; systematically putting to the test all the components of a HaaSS system's framework, under empirical, real-world conditions.We begin by providing a detailed breakdown of the frameworks core functions to be used by researchers and developers as a systematic blueprint on which to develop their own HaaSS systems. Then, by expanding considerably on the initial work presented in [24], we develop fully the Human-as-a-Security-Sensor framework with automatic HaaSS sensor integration, addition of automated sensor collection for activity-based features,

the online HaaSS remodelling functionality, and implementation of cloud based reporting for self-efficacy based features, in a revised version of Cogni-Sense, our prototype HaaSS system developed using the framework architecture. Benefiting from the complete implementation of the framework, we have conducted the first empirical experiment for utilising a HaaSS system in the context of semantic attack detection, across several more types of attack vectors beyond those used in [24]. Uniquely, we have also compared the results of the HaaSS sensors against a wide range of technical platforms and systems which claim to provide protections against such threats. Moreover, beyond an initial technical defence evaluation made in [24], here we test against a set of new attacks against a wide range of existing technical defences which also include those that exist on participant's own devices as part of the real-world experiment; which are subsequently shown to have effectively no impact on user's ability to detect and report suspected semantic attacks.

At the time of writing, this report represents the first empirical HaaSS experiment that addresses a wide range of existing and emerging semantic attacks, outside of a traditional laboratory environment.

3. A Human-as-a-Security-Sensor framework architecture

To build upon the work in [5] and systematise HaaSS sensors within a practical and technical defence system, we propose a HaaSS framework formulated by a set of three core processes which organise users as physical sensors (i.e., the HaaSS Sensor) within a typical cyber security defence architecture. Below, each of the processes are described according to their domain-specific functions in the HaaSS framework. Each discrete process (attack detection, classification and response) represents a collection of modular technical system functions.

• **Process 1 - Detection**. For a HaaSS sensor, threat detection can be an active or passive process depending on the context of attack exposure (e.g., actively searching for threats, or symptomatic exposure based on their activity profile). The detection process also continuously monitors HaaSS sensors to establish their detection efficacy across different user-interfaces.

• **Process 2 - Classification**. Here, HaaSS threat reporting is translated into decision actions for informing or triggering technical security measures. Scoring of the detection efficacy is employed as a filter to automatically respond to, forward or prioritise reports for manual review.

• **Process 3 - Response**. The result of an attack report classification (e.g., detection decision: true/false) is the deployment of threat mitigation functions (e.g., blocking a website URL, adding an email domain to spam lists, creating a malicious file signature, or sending out user threat awareness notifications) or otherwise. This process is also responsible for feeding back classification decisions with the aim to improve classification accuracy whilst adapting to a specific HaaSS sensor-base.

Using these high-level process-driven constructs, the HaaSS framework is formulated as an architecture with a series of functional components on a linear path (Figure 2). In the next subsections, each component is described in detail as to their role and function. Experimental system settings are included as tested in the prototype technical implementation called *Cogni-Sense*, which is discussed in detail in section 4.

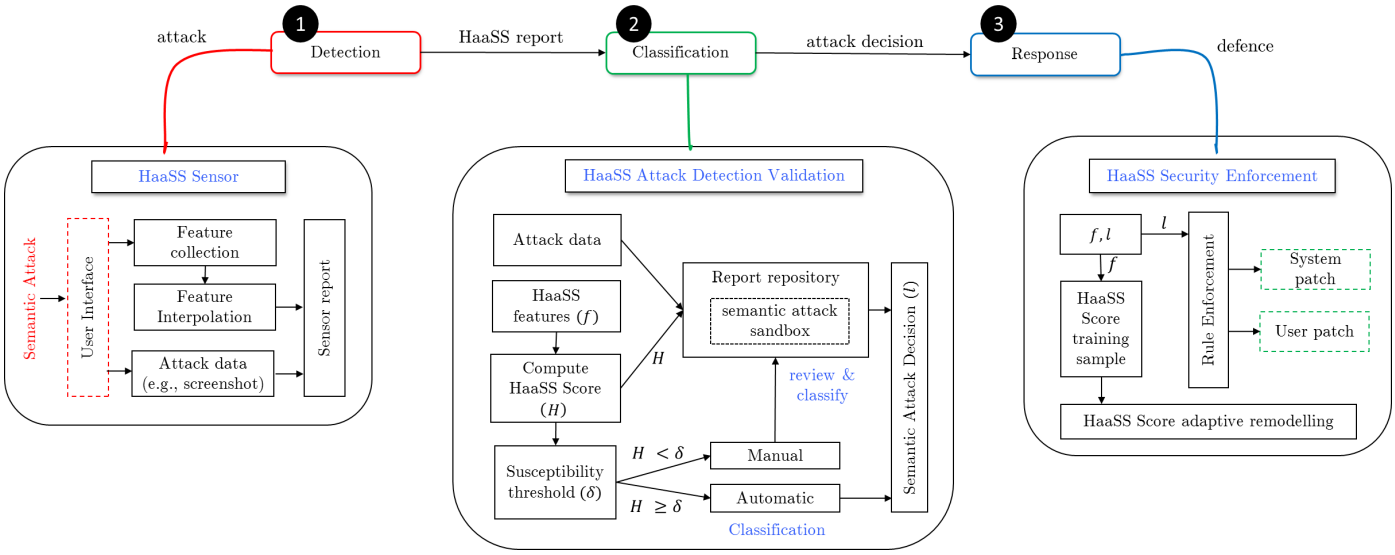
Detection system process: functional components

• **User Interface (semantic attack exposure)**. The user interface is an implicit component in the HaaSS framework and the source of all HaaSS sensors exposure to semantic attacks. The user interface applies to any computer system, whether served locally, remotely, by cyber or physical means.

Feature collection. HaaSS sensor activity on the user-computer interface generates real-time profile data to formulate the features for computing a HaaSS sensor's detection efficacy. Certain feature data are generated through HaaSS sensor interaction with the user interface and are collected by a HaaSS reporting platform integrated with the user interface. Where practical, feature monitoring can be automatic, but for specific features, such as self-efficacy based data, manual HaaSS sensor input may be required. Features can be collect both locally or remotely depending on implementation (e.g., whether computer security training records are referenced in a local or remote database).

• **Sensor Report (Feature interpolation and attack data)** An attack report is initiated by a HaaSS sensor when a suspected semantic attack is detected. The HaaSS report interface provides a mechanism for generating an attack report using the existing user interface available to the HaaSS sensor. A report captures, in real-time, the HaaSS sensor feature-set for the specific attack report context (e.g., for the specific platform). The sensor report extracts the HaaSS sensor feature-set, where raw feature data is interpolated and discretised into a format readable by the HaaSS susceptibility model, as described in more detail in section 3.1.1. The formatted feature-set is unique to each report, formulated dynamically based on the attack report context and time. The sensor report attaches attack data alongside HaaSS features to be sent for classification and response (e.g., defence). In a HaaSS report, the attack data can be built by extracting key threat information automatically from the user interface or allowing manual data input by the HaaSS sensor. For example, attack data may consist of video, images, files, links, interface meta-data, text description (supplied by HaaSS sensor) and diagnostic data on the platform serving the

Figure 2: Human-as-a-Security-Sensor defence framework



user interface. Sensor reports are delivered to a remote platform for classification in the HaaSS attack detection validation component, where attack data is forwarded to a semantic attack sandbox and HaaSS features are used to compute the sensors detection efficacy for the report.

Classification system process: functional components

• **Compute HaaSS Score H (HaaSS features).** A focal point of the HaaSS framework is the measurement of HaaSS sensor detection efficacy and reliability for semantic attacks reports. The H score forms the primary decision making process within the HaaSS framework and provides a mechanism to distinguish between credible threats for executing defence mechanisms. HaaSS report classification is initiated on-demand when an attack report is received by a HaaSS sensor and is computed as a validation measurement of the HaaSS sensor's attack detection efficacy. The validation measure utilises a susceptibility model (using the approach developed in [5]) to generate a detection probability metric which we coin as the HaaSS score (H). The H score is primarily used to determine the likelihood of whether a HaaSS report is a credible semantic attack or not, which, depending on the score and classification threshold defined, would result in automatic classification and immediate execution of security enforcing functions for reports classified as attacks. Alternatively, it is used to inform manual report classification of the likeliness of the report being a semantic attack. By default, once computed, each attack report received by a HaaSS sensor is assigned a H score generated based on the reports accompanying HaaSS features. We expand considerably upon the concept of the H Score (H) in section 3.1.

• **Report repository (Attack data, HaaSS features and the Semantic attack sandbox).** Within the HaaSS framework, the report repository stores all received HaaSS reports received by HaaSS sensors and serves as the storage source which supplies attack data and report information for review and classification in a semantic attack sandbox. Conventional computer security sandboxes are designed as safe containers which evaluate heuristically the semantics of untrusted code execution, or the meta-data of computer system files in a secure environment away from the host computer platform to detect attack patterns or anomalous activity. In HaaSS, the semantic attack sandbox is designed to expose behavioural and cosmetic user-interface attributes for analysis, to distinguish between legitimate or malicious intent (as suspected by the HaaSS sensor). As deception vectors (see taxonomy in [4]) primarily target the user-computer-interface, instead of technical software analysing system behaviour, it is human users who form part of the sandbox architecture by analysing the system behaviour through human sensory (e.g. visual interpretation). Here, human users security operators of the HaaSS system or peer HaaSS sensors. Currently there exist a number of online platforms providing sandbox functionality for email and website phishing, but to a limited extent. Online phishing repository PhishTank provides a web interface for reporting and reviewing phishing website and email attacks, supplying a screenshot of the report and for phishing websites the ability to interact with the attack itself via a HTML iframe (providing the phishing website not been removed from circulation) [22]. However, phishing email archive Millersmiles provides only a simple text scrape of phishing emails for review [23]. Unlike the HaaSS framework, these environments are limited to phishing attacks, do not integrate with security systems and provide no mechanism for report prioritisation based on the reporters' detection efficacy profiles. In the framework, all HaaSS sensor reports are forwarded to the semantic attack sandbox in a secure container for analysis, which also provides a mechanism for pro-active defence, where report classifications can be reviewed again if historic automatic

or manual classifications turned out to be incorrect.

• **Susceptibility threshold (δ)**. The HaaSS score susceptibility threshold controls when the system automatically classifies a HaaSS report as an attack (1), non-attack (0) or whether it defaults to an unclassified state (e.g., NULL). The threshold allows for adjustable control of false positive and false negative report classification, which vary depending on the HaaSS score model accuracy. For a configured threshold, if the upper or lower threshold is met, the HaaSS report is automatically classified (which also serves to automatically set the response label when adding the report to the HaaSS model training and validation dataset). In the case of an unclassified report state (i.e., HaaSS score <upper threshold and > lower threshold), the HaaSS report is marked as unclassified in the semantic attack sandbox for manual classification. Here, the HaaSS score is then utilised as a prioritisation metric for manual classification by a HaaSS report reviewer (e.g., security operations/platform personnel).

Response system process: functional components

• **Rule enforcement (System and user defence)**. The classification of each HaaSS report results in an attack decision which is true or false, which results in a response that issues a security enforcement function on a set of pre-configured rules based on the context of the report. This involves implementing a function that would protect against the semantic attack on the system or user and requires the HaaSS system to have a security enforcement module (SEM) that is integrated locally or remotely to external technical security platforms. The rule enforcement process provides traceability of HaaSS sensing and detection through to autonomous and preemptive defence measures against semantic attacks, by utilising report attack decision to invoke rule enforcement that results in technical security configuration and execution. For example, in the case of an organisational HaaSS system, for a HaaSS report of a phishing website classified as credible (automatically or manually), the HaaSS system can enforce a rule that sends a configuration setting to the organisations web proxy (using the port website URL) to block the phishing website. This would require API connectivity or middle-ware to translate the rule to configuration input. Another example, in a wider use case setting might be email distribution containing the details of the phishing website which is sent to all HaaSS subscribers (e.g., home internet users and other HaaSS sensors using the system). For each attack decision response, the HaaSS features and attack decision (true or false) are fed back into the H score training sample for the H score adaptive remodelling function.

• **HaaSS Score adaptive remodelling**. Adaptive remodelling is a continuous feedback mechanism designed to learn the behaviours and profiles of a HaaSS sensorbase by periodically re-training the HaaSS score prediction model and improving its accuracy. The remodelling procedure activates when the overall HaaSS detection distribution within the training data has deviated significantly from the HaaSS score models' original training data distribution. Here, by distribution we refer to the user detection rate which is based on the number of HaaSS report samples with correct or incorrect classification in the most recent baseline training and validation dataset. The task is periodic (e.g., running once a day, week or month), employing mean deviation as the trigger to invoke remodelling. The calculation takes x as the values of the most recent base-lined training data distribution and the current (at that point in time) distribution where $\textit{mean deviation} = \frac{\sum|x-\mu|}{2}$. If distribution exceeds a specific deviation threshold (with experiment configuration set as 0.5), remodelling is invoked. When remodelling is triggered the HaaSS score model is retrained on the most current HaaSS data sample with any tuning parameters defined i.e., train and test sample split, machine learning algorithm, cross-validation (CV), automatic feature selection. After retraining, the HaaSS score model's predictive performance is reviewed at different class probability thresholds to define the optimum threshold (δ) for minimising false positives or false negatives when computing the HaaSS score and setting the threshold for automatic or manual report classification as defined in the classification process. Adaptive remodelling can be configured to run as an autonomous process, or manually by a HaaSS system administrator.

3.1. The HaaSS Score: predicting sensor reliability

Similarities between computer-based and human-based security sensors can be drawn by a mutual requirement to measure their attack detection accuracy. Here, a major distinction is that each human sensor's detection accuracy is different and is based on their own detection efficacy profile. For this reason, we employ a metric, which we call the H score, to represent the trustworthiness of a HaaSS report in terms of the sensor's predicted detection efficacy for the particular report's context. The H score is computed by extracting a HaaSS sensor's most recent detection efficacy profile attributes (i.e., their susceptibility indicator predictors) at the point in time that a report is initiated.

Determining a set of features to accurately predict a human sensor's detection efficacy when exposed to a semantic attack is challenging. This is especially true when these features are required to be integrated into a real-world technical system, where they need be measured ethically, automatically and preferably in real-time. These requirements render psychological (e.g., being stressed), personality (e.g., conscientiousness) and demographic based features (e.g., age, gender, ethnicity etc.) of little practical use. Furthermore, depending on the HaaSS sensorbase, certain features may prove to perform better than others and therefore the utility of HaaSS features may be influenced contextually. The HaaSS framework does not

strictly stipulate a particular susceptibility modelling algorithm or features for computing the H score. For the prototype development of *Cogni-Sense*, we have utilised the susceptibility model and features developed in [5]. We expand on the *Cogni-Sense* H score model in section 4.

In the HaaSS framework, the H score facilitates three key objectives of HaaSS sensor reporting, (1) classification of HaaSS reports (i.e. credible semantic attack / not a credible semantic attack) in order to enforce automated security enforcing functions where necessary, (2) prioritisation of HaaSS reports based on detection efficacy of sensor for manual classification (where the reports H score is either too low or too high for automatic classification), with the aim to minimise the exposure time of vulnerable users by enforcing review precedence, and (3) the ability for a HaaSS system to learn and adapt to the changing detection efficacy profiles of its incumbent HaaSS sensor-base through remodelling to improve prediction accuracy of the H score. In these applications the H score can function in two modes independently or simultaneously, classification mode for autonomous, system-initiated security enforcement and prioritisation mode for manual, human-initiated security enforcement. In classification mode, the H score probability is evaluated against a system defined classification threshold which determines whether automatic report classification will occur. Depending on the H score model and its overall accuracy ([5] reported an overall accuracy of 71%), different classification thresholds will result in either higher false positives or false negatives or an optimum minimum for both. For prioritisation mode, the higher the probability, the more trustworthy the report and therefore the higher precedence it is afforded by manual human review and classification.

The H score is primarily designed as a mechanism for accurate treatment of semantic attack reports that are received from HaaSS sensors, however it can also be utilised as monitoring tool for continuous analysis of HaaSS sensor "health" as represented by their predicted detection efficacy. For example, in the case where a HaaSS sensor lacks the detection efficacy to identify a specific semantic attack on a specific user-interface (and platform), it is reasonable to assume that a report of this particular threat would never be received from/by this sensor. Conversely, for HaaSS sensors that are over-suspicious and therefore highly sensitive to suspected deception on the user-computer interface, the HaaSS sensor may be responsible for producing many false positive reports. To identify these weaknesses and perhaps deliver targeted training where needed, periodic review of passive H scores for HaaSS sensor detection efficacy would be beneficial.

3.1.1. Predictors of detection efficacy

The features described here are representative of higher-level feature constructs, which in their own right provide potential future research avenues for deeper feature discovery and engineering. Auditable features are features which can be collected in real-time and automatically, either on the HaaSS sensor's client-side system or remotely. Feature collection is a continuous task that initiates at user log-in and operates passively until the user logs off or shuts down their system. For remote feature collection, computation of auditable efficacy features can be conducted in a batched type mode, e.g., on demand or in real-time depending on whether all auditable features are sent to a server-side HaaSS database. Real-time computation can be less efficient as platforms forwarding features to the HaaSS database whether on the users machine or in the Internet must continuously monitor user activity from different locations and sending multiple data streams over the network and therefore introduce scalability problems. Similarly, local feature collection on the user's system can be computed in real-time or in a batch type mode. The process of feature generation, which applies to both local or remote collection, is presented visually in Figure 3.

Below, we describe the technical process of auditable feature generation using local batch-mode collection only. In the HaaSS framework, it is assumed that a continuous monitoring mechanism collects raw data from each measurable user-interface that the HaaSS sensor accesses and interacts with (e.g., web platform, application, file, device etc.). On different systems and for a range of disparate user-interfaces, the method for collecting raw access data from user interaction varies and therefore the exact technique for raw access collection is not prescribed within the framework. We do, however, develop a specific collection technique within the prototype HaaSS platform *Cogni-Sense* in section 4. Here, the algorithms for feature generation and interpolation are designed to function on any source of raw access data, assuming that the data has an an accurate timestamp.

- **Frequency of access (FR)**. By default, frequency of access is measured using a month of activity. For example, for user platform activity to be classified with a frequency of "daily" access (i.e., there is a maximum of one day elapsed between platform usage), this daily activity must be continuous for a period of one month, without more than one day difference between accesses. Different frequency transition thresholds can be applied when increasing or decreasing the frequency granularity scale. When a HaaSS sensor attack report is initiated, the user-interface access activity specifically related to the report (e.g., platform, application, file ... etc.) is indexed in an array of dates $\cdot v$, where $v_1, v_2 ... v_n$ represent a series of date and time ordered independent days. It is assumed that the collection mechanism records each individual access of a platform user interface continuously and therefore may consist of multiple accesses on the same day. For the frequency algorithm, only the first recorded platform access is indexed for each specific day.
- **Duration of access (DR**). The feature measurement for Duration of access (DR) applies to a monitored user interface. Here, the same assumption for the access ac-

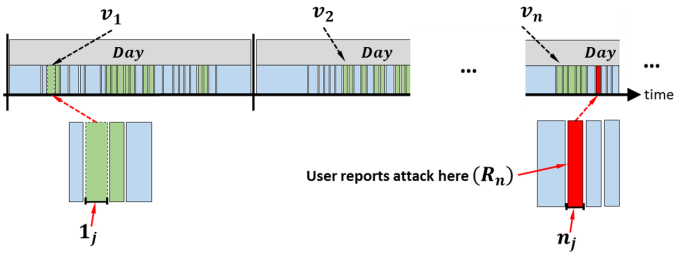
tivity collection mechanism is employed, where it is continuous for each individual access on a specific day and therefore may consist of multiple accesses on the same day. With this in mind, for each access in a period of one month, each timestamp is ordered to represent a start and stop elapsed time in seconds for each day.

- **Computer Security Training**. Computer security training is measured based on recency from the date it was last received, based on the training type (S1 - formal, S2 - work-based, S3 - self-study), delivery (S1 - coursework, S2 - video and games, S3 - video and websites) and platform type (e.g., email, social media, e-commerce etc.). Therefore, the elapsed time of computer security training increases until it is reset/reduced when training is next received. However, its recency is in relation to the date and time that a HaaSS sensor is made. Whilst monitoring activity of security training could utilise the frequency and duration algorithms for similar measurement criteria, here we assume that security training is recorded and validated prior to determining the time since it has been received, as well as the format, delivery method and platform. Security training is not locally measured, but on receiving a report to the HaaSS system, before H score prediction, the HaaSS sensor's detection efficacy features are queried on the HaaSS system for their security training record.

  In [5], each type of security training and the elapsed time since it was received (i.e., S1:3T) was measured separately from the delivery method related to that type (e.g., formal - coursework, work - games, self-study - websites and so on), as was elapsed time since security training for particular platform types (e.g., email, social media etc.). However, when measuring security training periodically it assumed all three would be entered as a single record as representative of a training received. So, although they are implemented and computed as separate features in the susceptibility model for generating the H score, it is assumed that there will be direct, time-based correlations between them that were not possible to accurately measure in the experiments conducted in [5]. In Figure 3, a visual representation of a continuous measurement time line is shown, where the frequency and duration feature algorithms applies interpolation and discretisation to raw recorded activity data from a HaaSS sensor.

For each of the auditable feature collection algorithms, the discrete measurement scale configuration can be adjusted to increase or decrease the granularity of activity monitoring. Frequency measured on a five point ordinal scale (daily, weekly, monthly, less than monthly, never) can be extended to measure a wider access frequency range (daily, every two days, weekly, every two weeks, every month, every two months etc.), or reduced to {daily, weekly, less than weekly}. The same applies to duration of access. Depending on the range and granularity of the discrete

Figure 3: Real-time collection of auditable HaaSS sensor features (FR and DR), variable green periods represent variable durations of access to the same platform and blue periods represent variable durations of access to other platforms over the period of one day. Access activity for frequency measurements is made by recording the first observed access instance for a day on a specific platform. Features are computed at the point in time when a user reports a suspected semantic attack for the specific platform in question, where R_n represents the nth report for jth platform interface accessed by the HaaSS sensor.



scale, this would either increase the required learning time to move between frequency threshold e.g., one month for a five-point scale, or decrease e.g., seven days for a three point scale such as daily, weekly, less than weekly. The greater the range of measurement, the more accurate the picture of HaaSS sensor activity. However, one constraint for increasing beyond a default scale is the need for retraining the model to utilise the new feature scale effectively.

Feature	Variable Format
Familiarity with platform	Not very, somewhat, very
Computer security self-efficacy	Novice to Expert (0:100)
Computer literacy self-efficacy	Novice to Expert (0:100)
Frequency of platform use	Never, <1x a month, 1x month, weekly, daily
Duration of platform use	None, <30 min, 30 min-1h, 1-2h, 2-4h, >4h
Time since ST (platform)	Never, >1y, ≤1y, ≤6m, ≤3m, ≤1m, ≤2w
Freq. of platform type use	Never, <1x a month, 1x month, weekly, daily
Dur. of platform type use	None, <30min, 30min-1h, 1-2h, 2-4h, >4h
Time since ST (plat. type)	Never, >1y, ≤1y, ≤6m, ≤3m, ≤1m, ≤2w
Time since ST (formal edu.)	Never, >1y, ≤1y, ≤6m, ≤3m, ≤1m, ≤2w
ST formal edu. (coursework)	No, Yes
Time since ST (at work)	Never, >1y, ≤1y, ≤6m, ≤3m, ≤1m, ≤2w
ST at work (videos)	No, Yes
ST Work-based (games)	No, Yes
Time since ST (self-study)	Never, >1y, ≤1y, ≤6m, ≤3m, ≤1m, ≤2w
ST self-study (websites)	No, Yes
ST self-study (videos)	No, Yes

ST = Infosec training, edu= education, y = year, m = month, w = week

Table 2: Random Forest Susceptibility model HaaSS features used to compute HaaSS Score H

3.1.2. Self-Efficacy Features

By self-efficacy, we refer to the generation and measurement of features that are supplied as part of self-assessment by HaaSS features in real-time (e.g., at the time of a HaaSS report), or over elapsed time as part of their HaaSS sensor profile. Whilst we include features that were selected during the modelling process in [5], the process for measuring self-efficacy applies generally for future features within the HaaSS framework.

- **Familiarity with platform**. Each time a user initiates an attack report, the user is required to provide their self-assessed familiarity for the platform interface they are reporting an attack on (as previously mentioned here we refer to the original scale defined in the susceptibility model developed in [5]). If practically available (e.g., in a fully integrated production HaaSS system), FR and DR features can be used to provide estimated validation of the familiarity reported by the user to define accepted correlation threshold's between the self-assessed familiarity and the actual frequency and duration in which the user accesses the platform; downgrading the familiarity value if this threshold is not satisfied. This mechanism can prevent users claiming familiarity with a system they do not use often and as result helps to preserve the integrity of the familiarity feature. The familiarity feature is supplied as a report meta-data input within the HaaSS sensor attack report interface.
- **Computer Literacy (CL)**. When a user updates their security training automatically through a compatible platform, or manually through an online form, they are requested to enter their self-assessed general computer literacy; using the original scale defined in the susceptibility model developed in [5]. If practically available (e.g., in a fully integrated production HaaSS system), the computer literacy metric can be compared against a user's recorded and validated computer usage, platform training and qualifications to develop an acceptable correlation threshold between between these attributes and the users self-assessed computer literacy. As demonstrated in experiment 2 in [5], platform-oriented features for frequency and duration of access can be used to accurately predict a user's computer literacy self-efficacy score. This mechanism can prevent users claiming a level of computer literacy that they are unlikely to have attained and as result helps to preserve the integrity of the computer literacy feature.
- **Security Awareness (SA)**. When a user updates their security training automatically through a compatible platform, or manually through an online form, they are requested to enter their self-assessed general computer security awareness; using the original scale defined in the susceptibility model developed in [5]. If practically available (e.g., in a fully integrated production HaaSS system), the security awareness metric can be compared against a user's recorded and validated computer security training, qualifications, correct HaaSS reports and detection of emulated attacks (e.g., in an embedded security training tool) to develop an acceptable correlation threshold between these attributes and the users self-assessed security awareness. As experiment 2 in [5] demonstrated, features of different types of security training can be used accurately predict a user's security awareness self-efficacy score. As with familiarity and computer literacy features, this mechanism can prevent users claiming a level of computer security awareness which they are unlikely to have attained and as result helps to preserve the integrity of the security awareness feature.

4. Cogni-Sense: a prototype HaaSS platform

Employing the HaaSS framework as a technical design blueprint, we have developed a prototype HaaSS platform called *Cogni-Sense*. The prototype's technical architecture is shown in Figure 4, where each of the coloured boxes within the architecture refer to a component's functional role within the HaaSS framework's defence system processes in Figure 2. By combining each of the system processes functional components into a real technical system, the development of *Cogni-Sense* provides a HaaSS sensor with a practical facility to report suspected semantic attacks and a platform in which to evaluate the concept of HaaSS for semantic attack detection. In Table 3 the technical components within *Cogni-Sense* are summarised, with direct traceability to the HaaSS framework functional components, by technical integration, platform configuration and a description of their functionality. The development of *Cogni-Sense* in this work demonstrates the technical feasibility of the HaaSS framework for building a real-world system around computer users, utilising the medium of human detection as a physical threat sensor for semantic attacks.

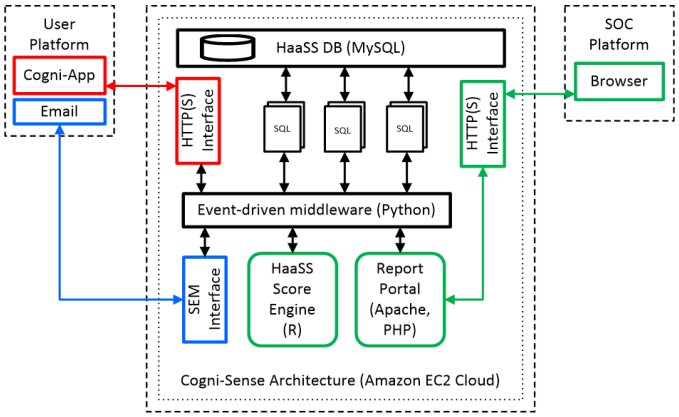
Framework	*Cogni-Sense*			
	Component	**Integration**	**Configuration**	**Component**
Detection(1)	User interface	Local host OS	Windows	HaaSS sensor device
	Feature collection	Self-efficacy auditable	Manual* Automatic*	*Cogni-Sense app* . and web form
	Sensor report	Host OS Interface Data	Windows 7/8/10 Python app Screenshot Text description Deception Vector	*Cogni-Sense* app
Classification (2)	Compute *H* score	H score Metric Mode	Random Forest Accuracy Class probability	*H* score engine (R)
	Report repository	Repository Sandbox	MySQL PHP Javascript	Report Portal
	Classification threshold	Upper Threshold Lower threshold	0.85* 0.1*	Python middleware
Response (3)	Rule enforcement	Trigger Rule	Attack classification Attack email alert	Python middleware SMTP interface
	H Score remodelling	Distribution Dev. trigger Train split Test split CV Feature selection ML algorithm ML Tuning	0.59* 0.05 0.8 0.2 10-Fold RFE RF 500 trees Mtry=4	H score engine (R) Python middleware

*[5] feature selection and threshold definition

Table 3: *Cogni-Sense* HaaSS platform configuration and alignment to the HaaSS framework architecture

The *Cogni-Sense* architecture consists of four key high-level components, (1) the HaaSS sensor detection platform (i.e., *Cogni-Sense* app), (2) a centralised cloud-based platform for classification and security response, (3) a security operations centre interface (e.g., web browser) that is used to access the cloud platform and (4) a security enforcement module (SEM) for the rule enforcement response process which provides integration between the cloud platform and

Figure 4: High-level overview of *Cogni-Sense* technical architecture



external security platforms. Each of the components are described below:

- ***Cogni-Sense* app**. The HaaSS sensor application is a multi-process python application that runs locally on the HaaSS sensor host device OS (in the case it was programmed for Windows OS). The local host OS was selected as the platform in which to develop the feature data collection and attack reporting interface as it provides the widest coverage of user-computer interfaces. For example, were the *Cogni-Sense* app developed as browser extension it would have only been able to report attacks and collect access activity from the browser interface, instead of wide range or other interfaces such as local applications, removable media, cyber-physical interfaces (e.g., NFC) etc. Therefore, the host OS provided the largest user-computer interface coverage available for the HaaSS sensor interface application.

  The first process is the HaaSS platform interface access activity monitoring for raw feature data collection. Whilst a number of mechanisms could have been used to extract platform interface identity, through application programmable interfaces (APIs) or graphical user interface libraries provided by the host OS or third-party applications, for simplicity we employ the PYWIN32 library to hook into the Windows WIN32 system for reading the text of applications windows in foreground (e.g., focused) of the user-computer interface which provides platform identity data that represents the identity of the user interface. This approach is less accurate and robust than validating directly the platform interface through an API, but is much less resource-intensive and proved suitable for the prototype implementation to record raw data used to create the H score auditable feature-set. In a production system, other programming platforms such as C++ (instead of Python) may prove more suitable for raw feature data collection due to their accessibility to lower-level interface functions in the host platform which benefit granular and accurate activity monitoring. For example, the employee monitoring software *ActivTrak* [26] provides platform-specific implementations

Figure 5: Platform usage meta-data collected by user activity monitor agent

Structure | Browse & Search | Execute SQL | DB Settings

TABLE activity_stream | Search | Show All

stream_id	Provider	Platform	Date	Time	Duration	Idle
913	Unknown Platform	Unknown Platform Type	2016-12-09	09:40:44	8.042	1.719
914	bbc.co.uk	news	2016-12-09	09:40:52	1.61	0.422
915	sharelatex	online collaboration	2016-12-09	09:40:54	7.853	1.25
916	skype	IM	2016-12-09	09:41:02	1.636	1.516
917	sharelatex	online collaboration	2016-12-09	09:41:03	2.354	0.922
918	facebook	social media	2016-12-09	09:41:06	217.199	126.031
919	bbc.co.uk	news	2016-12-09	09:44:43	1.977	0.438
920	facebook	social media	2016-12-09	09:44:45	36.708	6.141
921	twitter	social media	2016-12-09	09:45:22	10.8	1.391

for Windows and MAC OSX, using C++, to measure accurately user activity. However, in the case of this project such development would take extensive development time and would be expected to form part of a production monitoring system, which is outside the scope of this project. The raw data collection in the *Cogni-Sense* app for the activity monitoring process is shown visually in Figure 5. Platform interface recognition in the activity recording is performed by matching the window text in the foreground, using regular expressions against a known list of platforms in a local SQLITE database, where platforms not in this list are recorded as unknown with a watermark which allows for aggregated access and measurement of this specific unknown platform. Using this approach we only record specific platform access as required for the experiments in this project, preserving the privacy of participants. However, in a live system the SQLITE database can be updated with new platforms and interfaces in a modular fashion in the same approach used to download a white-list to a spam filter, web proxy or website category classifier.

The second process is the HaaSS sensor semantic attack reporting interface, which runs as a process in the system tray process as an eye icon, which when clicked, spawns a reporting window identifying the platform interface (as well as generating the local audi-table HaaSS features based on the platform context). Using the reporting window, the HaaSS sensor enters their platform interface familiarity self-efficacy feature, as well as attack meta-data such as suspected attack vector and general report-related information. A button is provided to send the report, which when clicked takes a screenshot of the user-computer interface, gathers the HaaSS features and meta-data and sends the report via HTTP to the cloud-platform, where classification and security response is applied to the HaaSS report. In the case of the HaaSS features, the feature generation algorithms described in the HaaSS framework in section 3.1.1 are programmed directly into *Cogni-Sense* app. The HaaSS attack reporting interface is shown in Figure 6.

- **Cloud plaform**.

Figure 6: The *Cogni-Sense* HaaSS reporting app icon running in system tray. When clicked, a reporting window is opened with the detected platform and report information

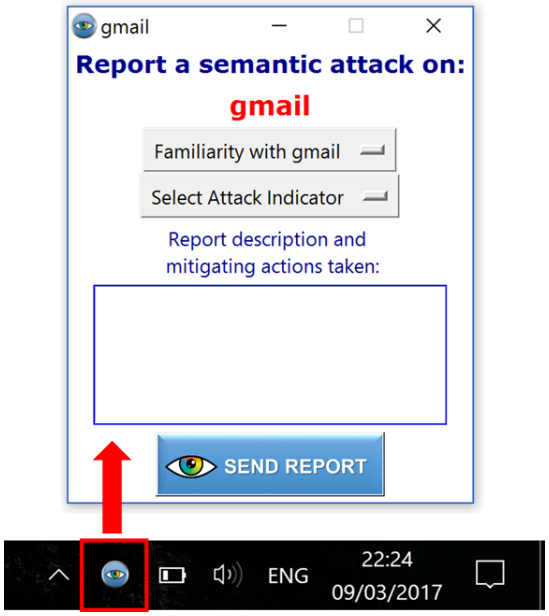
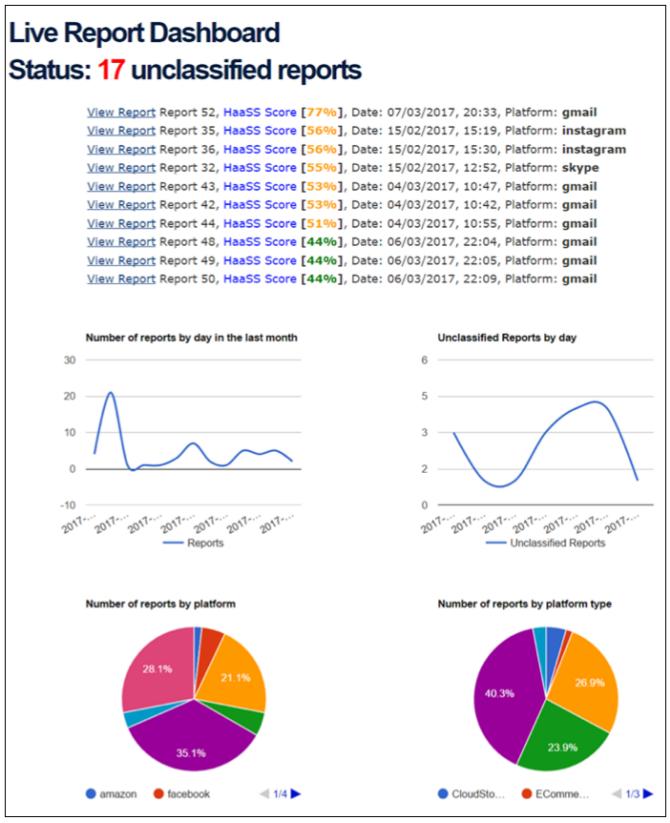


Figure 7: Example of the *Cogni-Sense* portal live report feed with predicted H score for HaaSS reports



The cloud component of *Cogni-Sense* was implemented within Amazon Web services on a Ubuntu Linux 14.04.3 virtual machine, with 1 GB RAM, a single-core 2.4 GHz Intel Xeon process with 30 GB of storage. In the cloud platform, the HaaSS processes of detection, classification and security response are coordinated by a Python middleware which provides integration and communication between different components of the system, such as a MySQL database which stores all HaaSS profile and report data, an R-based Random Forest H score engine (i.e., the susceptibility model), an Apache PHP and Javascript web-server which hosts the HaaSS report portal and sandbox, as well as interfaces to external security platform connectivity such as SMTP integration for confirmed semantic attack alerting and awareness training.

When a HaaSS attack report is received, it is stored in the report repository (MySQL database), where the H score is computed in the H engine, and depending on the score and configured susceptibility threshold, it is automatically classified or listed by H score priority on the report portal live feed dashboard. The report portal dashboard also includes analytics such as number of outstanding unclassified reports, the frequency of reports received, as well as the different platforms and platform types reported in attacks; which provides indications of where attacks are most concentrated or being targeted. The *Cogni-Sense* report portal dashboard is shown in Figure 7.

For the generation of the H scores, *Cogni-Sense* utilises the Random Forest susceptibility model developed in [5], integrating it into the technical system for HaaSS attack report classification and prioritisation. Random Forest is an ensemble decision tree machine learning algorithm, where a large number of decision trees are trained with different re-sampled versions of an original dataset and then each trained decision tree is used to predict data that was omitted from each sample, achieving low variance and naturally avoiding overt-fitting [27]. The model can operate in classification mode (0 or 1) or class probability mode (0 to 1). As strict classification is sensitive to false positive and false negative output, which would result in discarding mis-classified accurate reports or time spent reviewing non-attack reports, we used instead the class probability mode. The use of Random Forest in this case applies a "black-box" modelling approach to H score generation, which results in a loss of interpretability as to exactly why a HaaSS sensor's *Cogni-Sense* H score given their susceptibility indicators is higher (less susceptible) or lower (more susceptible). However, in comparison to simpler and more interpretable modelling algorithms, such as Logistic Regression used in [5], Random Forest is more capable of discovering non-linear relationships between a sensor's susceptibility indicators that may not be immediately obvious or intuitive. To better interpret the results of the H score, Random Forest provides a measure called variable importance which describes for each susceptibility indicator the degree to which it contributes towards

H Score model	Susceptibility indicators
Random Forest	CL (0.114), S3T (0.112), SA (0.107), FR1 (0.099), FA1 (0.097), S3_1 (0.087), S1T (0.073), S2T (0.073), S2_2 (0.068), FR2 (0.06), DR1 (0.05), S2_3 (0.048), S3_2 (0.048), DR2 (0.046), ST2 (0.039), S1_3 (0.033)

Table 4: Cogni-sense H score Random Forest variable importance for susceptibility indicators

prediction. In Table 4, variable importance is shown for the H score susceptibility indicators.

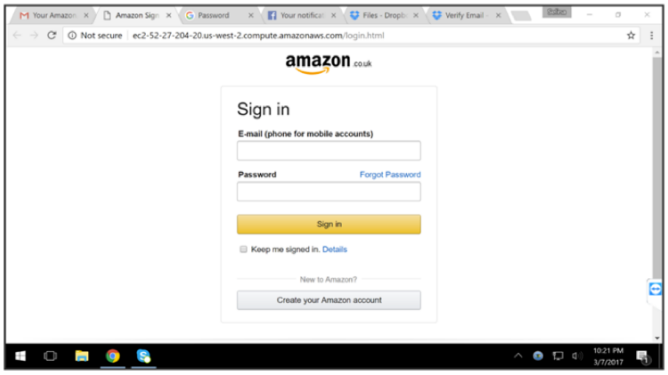
- **Security Operations Centre (SOC) platform**. The SOC platform refers to the Javascript-based web browsing platform used to interface with the *Cogni-Sense* cloud-based platform, where the live feed dashboard (e.g., report portal) and sandbox are located. The report portal live feed is the initial screen presented to SOC engineers (or peer HaaSS sensors) who manually review and classify semantic attack reports, where on selecting a report for manual review the image-container sandbox is opened for the reviewer. The sandbox provides an expandable image of the HaaSS attack report, the H score, report meta-data and a classification button (attack/not an attack) for report review. On selecting the report classification, the rule enforcement (response process) is triggered. The sandbox interface is shown in Figure 8, containing an example HaaSS attack report received by a HaaSS sensor in the experiment. An example of confirmed "semantic attack" classification for the HaaSS report in the sandbox is shown in Figure 8.
- **Security enforcement module (SEM)**. The SEM is configured as Python middleware which translates attack report classification into configuration parameters for security platforms that are integrated with the HaaSS system. For *Cogni-Sense*, integration with an SMTP server has been configured to automatically issue confirmed semantic attack alerts via e-mail to HaaSS-sensors. Given the modularity provided by the use of python middleware, in future iterations of *Cogni-Sense*, configuration rules issued via APIs to security products such as web proxies, anti-virus, firewalls etc., could easily be developed - however this is outside the scope of this project. For the confirmed attack report classification in Figure 8, SEM semantic attack e-mail alert response rule is shown in Figure 9.

In the next section, using the prototype HaaSS platform *Cogni-Sense*, we conduct an empirical case study experiment to evaluate the viability of HaaSS for semantic attack detection and its advantages over technical defence methods.

5. Evaluating the concept of HaaSS within a real-world experiment

In this section we put the prototype HaaSS platform *Cogni-Sense* to the test, to evaluate the concept of HaaSS under the conditions of a real-world scenario and within an

Figure 8: Example *Cogni-Sense* portal report screenshot for HaaSS report for Attack 3.2 (Amazon phishing website)



Report No. 54

Reporting User ID: 15

HaaSS Score [54%]

Report Date/Time: 07/03/2017, 22:21

Report Details/Comments:

non https, not amazon.com, instead is an Amazon aws instace

Enter Report Classifier Details/Mitigating actions:

Status: Unclassified

Classify Report

Real Attack | Not an Attack

empirical experiment environment. In this case study we compare the ability for HaaSS sensors to detect each semantic attack's deception vector [4] against a range of existing commercially available technical defences. Throughout the experiment we define detection as the identification of the deception within the process of a semantic attack and not the exploitation payload (e.g., execution of malware). That is, both HaaSS sensors and technical defences are evaluated by their ability to identify each semantic attack as a semantic attack, whereby a detection failure is recorded as either the HaaSS sensor or technical defence system allowing the deception vector to run up to the point of attack payload execution (e.g., clicking a link, entering login credentials, or opening a file). In cases where default behaviours of technical defence systems may implicitly block execution of attack payloads used in the experiment (e.g., executable file HTML opening an attack landing page - attack 1.2), as all attack payloads are emulated, this behaviour does not preclude the success of a real malware instance, which may evade such behaviour

Figure 9: Example of classified *Cogni-Sense* report, on selecting the option "Real Attack", the dataset for training the H score prediction is updated with the reporting users feature-set and classification decision as the training label. The Security Enforcement Module (SEM) then carries out any configured rules as part of the attack classification e.g., adding the report to user awareness training, sending out a threat alert email or adding the file name to a proxy gateway blacklist.

Report No. 54

Reporting User ID: 15

HaaSS Score [54%]

Report Date/Time: 07/03/2017, 22:21

Report Details/Comments:

non https, not amazon.com, instead is an Amazon aws instace

Enter Report Classifier Details/Mitigating actions:

Classification Status: Semantic Attack

Classification Date: 2017-03-15 23:01:23

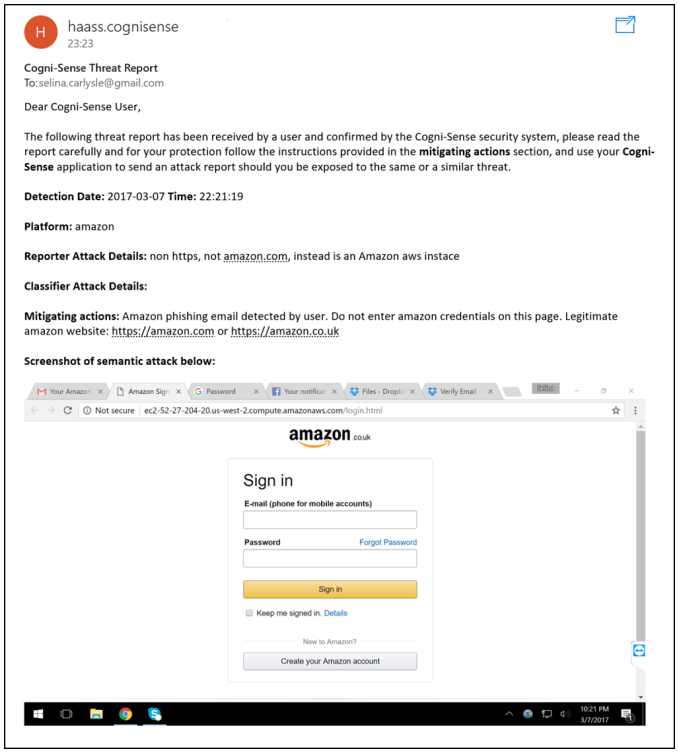
Classification Details/Mitigating actions:

Amazon phishing email detected by user. Do not enter amazon credentials on this page. Legitimate amazon website: https://amazon.com or https://amazon.co.uk

and successfully execute.

To measure distinctly different deception vectors within a semantic attack, where a complete attack process may logically consist of multiple phases (e.g., a semantic attack that directly lead to another separate semantic attack), we dissect each of the experiments interdependent attacks by utilising the classification methodology in [5] and treat each attack phase as a separate attack in its own right. This approach allows for measuring a HaaSS sensor's detection efficacy for each individual deception vectors in a multi-phase semantic attack (e.g., (1) clicking URL in email leads to (2) phishing website where entering credentials results in stolen identity). Thus, a user exploitation to any single phase in a multi-phase attack can also be viewed as exploitation to an individual semantic attack (which results in direct compromise e.g., (1) clicking URL in email leads to drive-by malware installation). For clarity, the attack model in Figure 11 highlights the points at which each semantic attack detection/exploitation is dissected and measured (deception vector text highlighted in red).

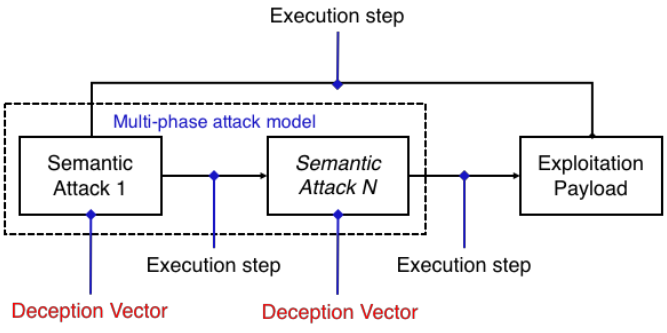
Figure 10: *Cogni-Sense* HaaSS report attack classification triggering SEM module rule: attack awareness email security enforcing function rule



5.1. *Zero-day semantic social engineering attacks*

The vast majority of semantic social engineering attacks, when initially launched, are largely undetectable by technical defence systems, because they primarily employ cosmetic or behavioural deception vectors and as a result often leave very small technical footprint that can be easily analysed by a computer system, especially if the deception has been designed to utilise legitimate and intended user functionality [4]. Consequently, technical heuristic detection capabilities have a limited view of potential attack vectors through user actions, instead of system interfacing malware. In most cases, technical defence systems are forced to rely on forensic attack and exploitation reports

Figure 11: Experiment attack model for measuring H score and exploitation for individual semantic attack's deception vector in both singular and multi-phase semantic attacks



Attack	Emulated Attack	Depend.	Description
1.1	Spear Phishing Email	-	Targeted email advertising fake job with GoogleDrive URL to job description PDF
1.2	Cloud Storage File Masquerading	1.1	Malware HTA file masquerading as PDF in online Google Drive folder
2.1a	Fake Facebook account	-	Friend request from fake Facebook account
2.1b	IM Phishing	-	Unsolicited Facebook message containing Facebook page link
2.2	Multimedia masquerading	2.1b	Malicious image link masquerading as Facebook video post
3.1	Phishing Email	-	Amazon order confirmation with tracking URLs leading to phishing website
3.2	Phishing website	3.1	Amazon login phishing website which captures user login details
4.1	Automated IM phishing	-	Unsolicited Facebook message from non-friend account with shortened URL
4.2	Phishing website	4.1	Facebook login phishing page
5.1	Spear USB	-	Packaged and branded USB designed specifically for target, delivered in the post
5.2	File masquerading	5.1	Executable masquerading as PDF file

Table 5: Experiment emulated semantic attacks sent to participants with indicated date and time at which the attacks were launched for all participants (this does not guarantee that participants were exposed to the attacks at the time of launch)

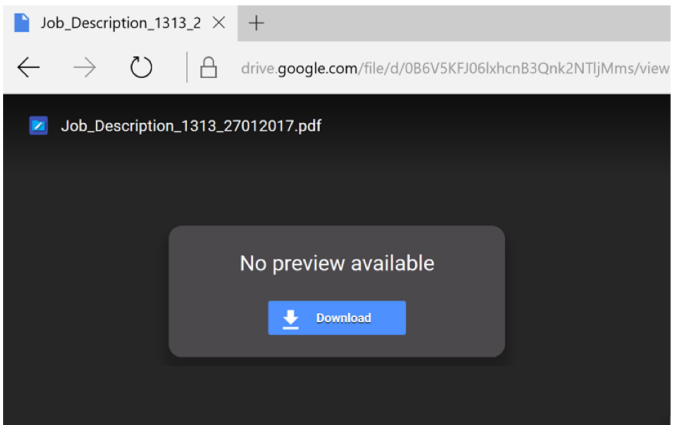
in order to develop attack signatures that can be used to match and filter potential threats.

For example, it is difficult to characterise a website as phishing if the URL is not registered with a spam database, and does not use obvious tricks such as popular domains names used as sub-domains, obfuscated by domain suffixes which are not related to the masqueraded website (e.g., www.amazon.net-shopping.tk). In cases where a phishing website name originates from a legitimate and credible service provider (and does not attempt to obfuscate its appearance), until the website has been reported as malicious (or contains easily identifiable malicious code or web re-directions in the web page), most technical defence platforms will not recognise the website as phishing. The same example can be seen in spam emails where spam protection mechanisms analyse components such as sender "From" and "To" address, subject title, domain, hyperlinks, attachments, salutation and common phrases (imposing a sense of urgency) which match known common patterns in conventional phishing attacks. However, if the email body consisted purely of a deceptive image from a domain name not registered in a black list, then character-based classification effectiveness is significantly reduced, as the spam protection is unlikely to have the ability to interpret the visual information in the image. On the other hand, human users, are implicitly interfaced with such attributes and are therefore better placed to decide whether system activity on the user interface is anomalous or not, based on their multisensory detection abilities, experience and knowledge.

As the semantic attacks evaluated in this case study were developed specifically for the experiment, and consequently have not been seen by technical defence systems or users before, they are assumed to be zero-day semantic social engineering attacks at the time of the experiments. In Figures 13 (semantic attack 2.1 and 2.2) and 12 (semantic attack 3.2), we provide two examples of semantic attacks executed within the experiment. Table 5 provides an overview of the full set of semantic attacks tested.

Whilst the majority of attacks employed in this experiment are distributed remotely via the Internet, the spear USB attack (5.1) branches into physical space and tests whether HaaSS sensor detection can be useful in detecting threats that cross a cyber-physical domain. An example of the spear USB attack is shown in Figure 14 and PDF file masquerading in Figure 15. Each of the spear USBs were designed to be targeted by printing on them official logos associated specifically to platforms the participants reported to use and have a specific affinity for in terms of their Internet profiles.

Figure 12: File masquerading attack in Google drive cloud storage platform, filename appears to be a PDF but is in fact a HTA file when downloaded.

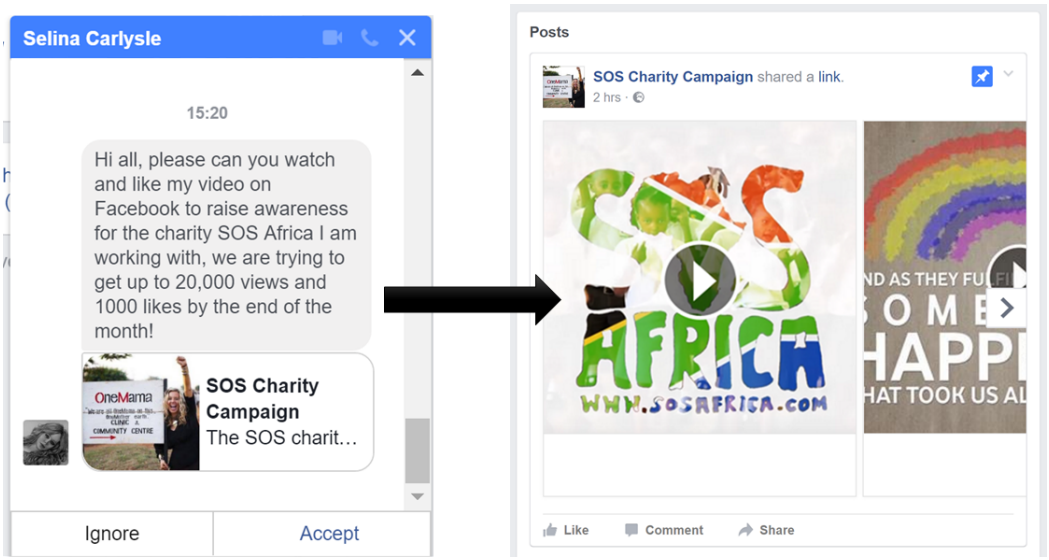


5.2. Laboratory environment: technical defence evaluation

The environment used to evaluate the detection capabilities of the technical defences (listed in Table 7), against each each of the semantic attacks (listed in Table 5), was presented in the form of a virtual machine running a Windows 10 operating system with the latest security updates. For each browser installation, the Internet security web plugins for each specific anti-virus (where applicable) were installed. The default Windows 10, browser, e-mail, anti-virus and platform configuration settings were applied for all testing. Morover, the Windows firewall was left on, but in some cases, Windows Defender was disabled by other anti-virus products.

Technical defence testing functioned by completing the necessary steps required to expose the user and system

Figure 13: Facebook phishing message and URL leading to a fake Facebook charity community page with malicious image link masquerading as a Facebook video post



to an attack's deception vectors and payload. For example, in the case of attack 1.1 (email spear phishing) and 1.2 (Google drive file masquerading), the target e-mail account was accessed on the respective email provider (in this case Gmail), the e-mail read (if not sent to the spam folder) and the URL clicked on. On clicking the link, the Google Drive website would be loaded, providing a view of the file in the Google Drive share. Assuming the file was available to view, the file was downloaded and then opened on the system. If the file was allowed to execute, an attack landing page would open via a browser.

For this experiment, we have selected technical defences which are very likely to be used by both personal users and enterprise organisations on their computer systems. In Table 7, each is evaluated according to the functional capabilities for detecting semantic attacks. Whilst email and browser platforms tend to offer anti-phishing, URL filtering and anti-malware defence, they do not directly employ heuristic scanning as part of this functionality, which as shown, is exclusively provided by the anti-virus software that we have evaluated. This means that in practice, most email providers rely on signature-based attack recognition for email by query through registered attack databases. For a number of the anti-virus products, installation of their full-product suite included browser security add-ons specifically designed for detection of website and email phishing threats and deception-based attacks. Amongst the anti-virus products, Norton, Sophos, Avast and Kaspesrky included and installed browser security add-on software as part of their security suite.

Outside of the technical defences evaluated on the Windows 10 laboratory test bed, the nature of the empirical experiment environment included inherently any technical defences each of the participant reported using on their personal systems. Whilst this increased the breadth of technical defences exposed to the semantic attacks beyond those tested specifically in our tested, the experiment results demonstrated that participants' own technical defences played effectively no part in their ability to detect deception-based threats (see section 5.3.2). In Table 6 the following technical defences were reported by to be installed and used on a number of participants' platforms.

Technical defence	Type
Malwarebytes	AV
McAfee AntiVirus Plus	AV
IBM Trusteer	Endpoint fraud detection
BitDefender	AV
Zenmate VPN	VPN
PIA	VPN
Adblock	Browser add-on
Symantec Endpoint Protection	AV
Norton Internet Security	AV
Windows Defender	AV
ESET Nod 32	AV
Windows Defender	AV
Sophos Endpoint Protection	AV
AVG AntiVirus	AV
Avast AntiVirus	AV

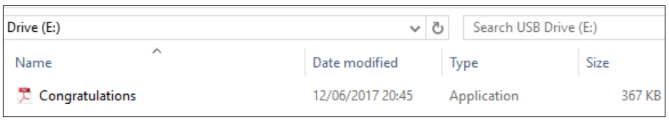
Table 6: Participant HaaSS sensors' on device technical defence platforms

5.3. Case study: Crowd-sourcing HaaSS

In this case study, we have integrated participants and their personal systems directly into *Cogni-Sense*. Here, we use the concept of crowd-sourcing HaaSS sensor detection to utilise attack report telemetry for semantic attack detection. Under this crowd-sourcing scenario, HaaSS sensor

Figure 14: Spear USB attacks (Facebook, Instagram and Blackhat participant profiles)

Figure 15: PDF File masquerading



capabilities can be representative of multiple HaaSS platform structures and deployments, such as HaaSS sensors who are personal computer users subscribed to a public HaaSS system, within a business setting (which would include general business users and security operations users), or even paid human intelligence task workers (HITs) [28, 29] that have been effectively integrated within a hybrid human/machine technical security system [30, 31], drawing from a pool of user profiles qualifying as HaaSS sensors. In the case of the latter, it is reasonable to assume that the highest performing HaaSS sensors, as proven by their track record and detection performance, would likely command a higher fee. In all cases, a key requirement is for HaaSS platform integration directly into different users' computer devices, which in the era of Internet of Things and Bring-Your-Own-Device (BYOD) within working environments means that more often than not these computer devices are the users' own.

5.3.1. Empirical experiment environment

We have developed the experiment environment in adherence with the five design principles for user studies in security and privacy proposed in [32]. All participants were assigned (i) a *primary task* of reporting suspected semantic attacks using the *Cogni-Sense* software; (ii) where the semantic attacks introduce a *realistic risk* by exposing users to real-world deception vectors on their own, personal computer systems, where similar real attacks could be received by the participant before, during or after the experiment; (iii) the participants *were not primed* to the nature or format of the semantic attacks and how, or even if, they will be received to prevent detection bias; (iv) adding element of *double-blinding* as the researchers did not know when participants would be exposed to the emulated experiment attacks or whether their systems security would block the attacks from reaching the participants and (v) ensuring the *terminology and dissemination of the experiment* in relation to the delivery method of the semantic attack threat, security and privacy *was consistent* in order to prevent bias in the participants behaviour and overall reported results.

As the experiment HaaSS sensor report interface *Cogni-Sense* app is installed directly on personal computer devices, the experimental environment was primarily presented in the form of the participant's own system. However, for the HaaSS reporting interface in *Cogni-Sense*, a HaaSS sensor app was developed for Windows operating systems, thus, recruitment was limited to the participants who had a Windows operating system (Windows 7, 8 or 10) installed on their primary computer device. Consequently, this also required participant devices to be a desktop PC or laptop.

The goal for participants was to detect and report any suspected attacks without falling victim to them, by using the *Cogni-Sense* app HaaSS report interface. They were advised to use the application to send a report whenever they detected a suspected attack. As part of the recruitment process, the participants were required to complete a questionnaire, which was used to collect HaaSS features. Each participant was assigned a HaaSS number and reporting user ID to match reports to corresponding HaaSS sensors in the experiment and received a video guide on how to use the reporting tool. During the experiment, if any participant was exploited by one of the semantic attacks, they were either directed to a second attack, as shown in Table 5 or redirected to an attack landing page where the participant was required to enter their experiment ID and name. For both cases, should participants fail to report an attack or submit their details after exploitation, this information would still be available for offline analysis via the activity collection process.

In total, 26 HaaSS sensors were recruited by inviting participants to take part in the experiment with the incentive of a £50 ($67) participation voucher[1] given to each participant at the end of the experiment's six week period. There was no competition element to receiving the voucher. All participants would receive a voucher regardless how much they participated in the experiment (with the exception of those participants which decided to drop

[1]With respect in particular to beneficence, the participants were compensated from their involvement in the experiment. The costs/risks were very low: participants were fully informed of their task and no subterfuge was detected during the experiment, as they were advised they would receive the participation voucher irrespective of performance. Furthermore, no personal information was collected.

ID	Ref	Security Platform	Type	Phish detect	Web Rating	URL block	Heuristic scanning	On access malware
E1	[33]	Yahoo Mail	Email	✓	✗	✓	✗	✓
E2	[34]	Gmail	Email	✓	✗	✓	✗	✓
E3	[35]	Outlook	Email	✓	✗	✓	✗	✓
E4	[36]	ProtonMail	Email	✓	✗	✓	✗	✓
E5	[37]	Yandex	Email	✓	✗	✓	✗	✓
E6	[38]	GMX	Email	✓	✗	✓	✗	✓
E7	[39]	Mail.com	Email	✓	✗	✓	✗	✓
B1	[40]	Firefox	Browser	✓	✓	✓	✗	✓
B2	[41]	Chrome	Browser	✓	✓	✓	✗	✓
B3	[42]	Opera	Browser	✓	✓	✓	✗	✓
B4	[43]	Commodo Dragon	Browser	✓	✓	✓	✗	✓
B5	[44]	Avast Safezone	Browser	✓	✓	✓	✗	✓
B6	[45]	Microsoft Edge	Browser	✓	✓	✓	✗	✓
B7	[46]	Safari	Browser	✓	✓	✓	✗	✓
A1	[47]	Commodo Cloud	AntiVirus	✓	✗	✓	✓	✓
A2	[48]	AVG AntiVirus	AntiVirus	✓	✗	✓	✓	✓
A3	[49]	Avast AntiVirus	AntiVirus	✓	✗	✓	✗	✓
A4	[50]	Windows Defender	AntiVirus	✓	✗	✓	✓	✓
A5	[51]	Norton Security	AntiVirus	✓	✓	✓	✓	✓
A6	[52]	Kaspersky IS2017	AntiVirus	✓	✓	✓	✓	✓
A7	[53]	Sophos Intercept X	AntiVirus	✓	✓	✓	✓	✓
P1	[54]	Facebook	Web platform	✓	✗	✓	✗	✗
P2	[55]	GoogleDrive	Web platform	✓	✗	✗	✗	✓
P3	[56]	Windows10	OS	✗	✗	✗	✗	✗

Table 7: Technical defences tested against semantic attacks in experiment case study

out of the experiment). The participants consisted of a mixture of students, lecturers, working IT professionals (all originating from a range of disciplines) and the wider general public. We have recruited a demographic that varied in terms of computer literacy and computer security awareness, without specifically organising participants based on their individual skills. For example, 38% of the participants were female, the average age was 28, and 40-50% participants had no educational background in computer science or information security. Furthermore, independently of formal training or expertise, analysis of participants' reported computer literacy and security awareness showed the sample is fairly evenly distributed across the self-efficacy scale (on a scale of 0 to 100, 0=Novice and 100=Expert), reporting a standard deviation of 18 and variation of 313 and a standard deviation of 24 and variance of 576, respectively. Therefore, whilst the total number of participants remains relatively small, the sample is shown to be suitably diverse to represent most types of computer user (or equivalently, HaaSS sensor), which allows for greater generalisation of experiment results.

In a HaaSS system, all employees in an organisation or the general public (e.g., from the Internet) are viable and legitimate HaaSS sensors. So, irrespective of whether a participant sample consists of solely of computer experts, novices or a mixture between the two (which is the case with this experiment), it is the application of the H score that distinguishes between sensors, by providing a metric of their expected detection efficacy based on features which have shown to generalise across a wide demographic [5]. With this in mind, it is preferable to be agnostic of the HaaSS sensor from a demographic perspective, instead relying on the H score to provide a unique probability of detection efficacy, which can be dynamic based on time and degree of training, rather than profiling a sensor individually on attributes which are unethical, discriminate (age, gender) or difficult to measure consistently (personality, emotional state); which we have shown to be of limited value and impractical in a technical system [5].

5.3.2. Experiment results

In Tables 8 and 9, the HaaSS sensor participants and technical defences detection results for the experment are shown, respectively.

General observations: Exposure to the experiment attacks was quite varied across HaaSS sensors. Most HaaSS sensors were not exposed to a majority of the attacks, with at least three HaaSS sensors only being exposed to one attack set. However, exposure was also dependent of attack detection efficacy. For example, a HaaSS sensor with perfect detection performance would only have ever been exposed to five of the eleven attacks in total. For Facebook attacks (2.1, 2.2, 4.1 and 4.2), a total of 8 out of 26 HaaSS sensors did not have Facebook accounts or had prevented their profiles from being searchable, meaning at least eight HaaSS sensors would definitely not have been exposed to these semantic attacks.

In total, 17 out of 26 (65%) HaaSS sensors detected at least one semantic attack, with the HaaSS sensor base in the experiment detecting all semantic attacks across all platforms; 2.2 (Facebook video media masquerading) was the only semantic attack not reported by a HaaSS sensor. By comparison, only 3 out of 24 (13 %) technical defences detected a semantic attack in the experiment and these detections were specifically limited to phishing emails only; all of which only identified one of the phishing emails each. Therefore, all other semantics on different platforms went undetected by the technical defences.

In terms of individual attack detection, the HaaSS sensors did not find any specific attack very easy to detect.

	HaaSS sensors																								
A	**H1**	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**	**H10**	**H11**	**H12**	**H13**	**H14**	**H15**	**H16**	**H17**	**H18**	**H19**	**H20**	**H21**	**H22**	**H23**	**H24**	**H25**	**H26**
1.1	.49	.72	.54	.55	-	.59	✗	.50	-	-	.70	.59	.48	-	.53	-	-	.33	.46	-	-	.46	.54	.70	.79	.66
1.2	-	.66	.53	-	-	-	✓	.26	-	-	-	-	.34	-	.18	-	-	.44	-	-	-	-	.17	-	.71	-
2.1A	-	-	-	-	-	.60	✓	-	-	-	.58	-	-	-	.48	-	-	-	-	.49	.30	-	-	-	-	-
2.1B	.51	-	-	-	-	-	✓	.51	-	-	.58	.43	-	-	.48	-	-	.32	-	.49	-	-	-	-	-	-
2.2	.51	-	-	-	-	-	-	-	-	-	-	-	-	-	.48	-	-	.32	-	-	-	-	-	-	-	-
3.1	-	-	.54	.55	-	.60	✗	.63	-	.51	.70	.59	-	.56	.56	.46	.49	.33	.46	-	-	-	.54	.70	-	.66
3.2	-	-	-	-	-	-	.62	-	-	.49	-	-	-	.48	-	-	.26	.44	.46	-	-	-	-	.49	-	.67
4.1	-	-	-	.63	-	.56	-	-	-	.53	.72	.43	-	-	-	-	-	-	-	.49	-	-	-	-	-	-
4.2	-	-	-	-	-	-	-	-	-	.56	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5.1	-	.67	-	.61	✗	.63	✗	.39	.35	-	.65	.52	-	-	.49	-	-	.45	.47	-	.30	.37	.66	.44	-	.68
5.2	-	-	-	-	✗	-	✓	-	.35	-	-	-	-	-	-	-	-	-	.47	-	.57	-	.68	.72	-	.64

Table 8: HaaSS sensor attack detection results. The value in each cell refers to the H score. Note that the HaaSS sensor number shown here is different to the HaaSS sensor ID assigned to participants during the experiment, as the IDs in *Cogni-Sense* were automatically generated by a database and not contiguous. Green indicates detection and report, orange indicates no detection or exploitation, red indication no detection and subsequent exploitation.

	Technical defences																						
A	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**P1**	**P2**	**P3**
1.1	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-	-	✗
1.2	-	-	-	-	-	-	-	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	-	✗	✓
2.1A	-	-	-	-	-	-	-	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-	-
2.1B	-	-	-	-	-	-	-	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-	✗
2.2	-	-	-	-	-	-	-	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-	✗
3.1	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-	-	✗
3.2	-	-	-	-	-	-	-	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-	-	✗
4.1	-	-	-	-	-	-	-	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-	-
4.2	-	-	-	-	-	-	-	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-	-
5.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✗	✗	✗	✗	✗	✗	✗	-	-	✗
5.2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✗	✗	✗	✗	✗	✗	-	-	✗

Table 9: Email provider attack detection results, green indicates detection, orange indicates no detection and or deception prevention, but default behaviour that can help to mitigate payload execution, red indicates no detection and no prevention of deception and execution payload.

For attacks 3.1 (Amazon phishing email) and 4.1 (Facebook phishing message), there was an equal number of HaaSS reports to HaaSS attack exploitation, which was shown to be the highest performance for the HaaSS sensors in the experiment. Nonetheless, attack 1.1 (spear phishing email), was found to be a challenging attack to detect for the HaaSS sensors and in this case was specifically tailored to the HaaSS sensor. The USB spear phishing attack 5.1 also proved equally challenging, with both attacks 1.1 and 5.1 exploiting at least 35% and 38% of HaaSS sensors, respectively. For HaaSS sensors exploited by attack 1.1, 67% were also exploited by attack 1.2 (Google Drive file masquerading), and for attack 5.1, 60% were also exploited by 5.2 (PDF file masquerading). The most difficult attack in the experiment for HaaSS sensors appeared to be the Google Drive PDF file masquerading, whereby 88% of HaaSS sensors exposed to the attack were deceived into downloading the fake file.

For the spear phishing e-mail attack, the H score was the least accurate, predicting high probabilities of detection efficacy for specific HaaSS sensors. However, it is worth highlighting the complexity of these two semantic attacks in particular, compared to the rest of the attacks deployed in the case study. The spear phishing attack was particularly well-crafted, with very little indication of the signs the the e-mail was illegitimate. For example, the *GoogleDrive* URL is in fact legitimate, and the e-mail communication appears at first glance quite credible as the domain name, matches correctly the email sender, which appears to originate from a company called "Iname", whereby the company is in reality completely fictitious. A brief online investigation identifies "@iname.com" as a free email domain provided by *Mail.com* email provider and the company has no public website listed or details available by search engine. However, the detailed target information within the email, good grammar, sincerity and punctuality of the message combine to make the e-mail appear convincing and authentic.

For the spear USB attack, the device was posted directly to HaaSS sensors' home addresses, labelled professionally with logos from platforms and companies based on online profiles they associated most affinity (and usage) to during the participant survey recruitment. The HaaSS sensors needed only to insert the USB into any of their systems for the deception to have been successful, with no technical defence being capable of preventing this physical action from being executed; other than that of the HaaSS sensors themselves. In a real world scenario, it is quite possible that the USB device would contain zero-day malware, which may have compromised a users system immediately, irrespective of whether they used a Windows, Mac or Linux operating system; a malware designed for these platforms could easily have been planted on the device. On the other hand, for this attack in particular, it is

Figure 16: HaaSS sensor report for spear USB through *Cogni-Sense* app

Report No. 107

Reporting User ID: 29

HaaSS Score [61%]

Report Date/Time: 19/06/2017, 21:47

Report Details/Comments:

Unsolicited USB received through post. I am attending BlackHat this year so attack was very targeted. I did not plug the USB into any system I use, I instead put it though a sheepdip process on a standalone system, running the USB

also possible that participant HaaSS sensors did not expect to be exposed to, or even expect to report types of semantic attacks that manifest in cyber-physical form (which is a common form of distribution for semantic attacks [4])

Nevertheless, 23% of HaaSS sensors detected attack 5.1 and generated semantic attack reports for the spear USB attack deception physical space, which demonstrates deception-based attack detection that simply would not have been possible for a technical defence system. One such report is shown in Figure 16, where a picture was taken of the USB and reported via the *Cogni-Sense* app. In this case, the HaaSS sensor titled the image "USB", which allowed for the app to determine their frequency and duration of both USB and removable media through the monitoring functionality. If the title of image was different, the frequency and duration measurements of this report would likely be inaccurate, which would result in a potentially detrimental H score. Thus,further development of *Cogni-Sense* for coping with physical user-interface deception reports is warranted for computing accurately the H score for such attacks more efficiently in the future. Regardless, in the absence of a reliable H score, *Cogni-Sense* continues provides a facility for reporting semantic attacks in physical space.

Performance of the H score: The HaaSS scores generated exhibited a low standard deviation of 0.07 and a sample mean of 0.56, with the most confident correct H score only reporting a 71% detection probability as the highest prediction result. Consequently, no reports qualified for automatic H score report classification threshold and therefore all reports were sent for manual classification in the semantic attack sandbox. The lower H score confidence may be explained by a number of reasons. Firstly, HaaSS sensors auditable (i.e., activity) features were only collected on one device and therefore a large proportion of their platform usage may not have been seen by the *Cogni-Sense* app. Secondly, the recency of HaaSS sensor computer security training records within the HaaSS feature database on *Cogni-Sense* automatically decreased over time, unless it was updated by HaaSS sensors when they reported receiving training.

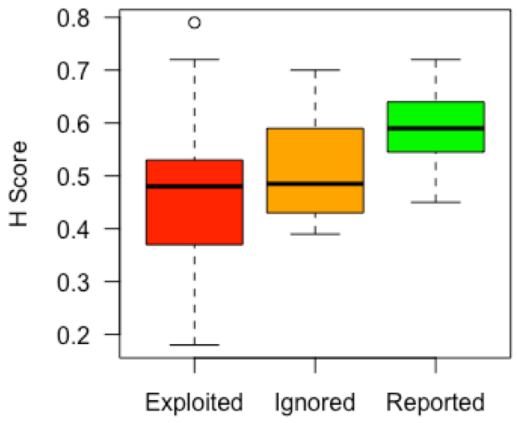
Table 10 shows the model's performance using the default and optimal classification thresholds identified in the experiment. We did not include HaaSS sensor reports or exploitation records for which it was not possible to generate a H score. HaaSS sensors who were not exploited by attacks by performing mitigating actions (orange box in Table 8), but who also failed to report them, were also not included. They were deemed neither exploited or as having "detected" the attack by failing to report it. It was not possible to generate H scores for HaaSS sensors' H5 and H7 exploitation records because their database files used to generate the frequency and duration features via the *Cogni-Sense* app were deleted by the participants at the end of the experiment and therefore not supplied for analysis.

The optimal H score classification threshold was reported at the probability cut-off value of 52% for predicting HaaSS sensors attack reporting and exploitation, which demonstrated a 12% increase in prediction accuracy over the null classifier. In the case of the default classification threshold of 50%, the H score was only 7% more accurate than the null classifier. Overall, the H scores were consistent with the experimental results reported by the susceptibility model testing in [5], reporting an average prediction accuracy of 74% compared to 71%, respectively. Notably, this agreement in prediction accuracy illustrates the H scores ability to generalise beyond initial laboratory conditions to those of empirical, real-world application, whereby the performance of the H score for a sample size of 26 participants was consistent with that reported for the significantly greater sample size of 365 participants used to develop the machine learning model utilised.

Arguably, the most important validation of the H score performance (and its viability as a robust measure of sensor reliability) is observed by contrasting H scores with corresponding HaaSS sensor detection results. In Figure 17, a box-plot of the H scores in the case study shows that as the probability of detection efficacy increased (as predicted by the H score), semantic attack detection efficacy was indeed significantly higher.

HaaSS detection of further semantic attacks: A total of 40 extra HaaSS sensor reports were received for

Figure 17: Overal distribution of HaaSS sensor H scores in experiment case study



Class. Threshold	Accuracy	TP	TN	FP	FN	Precision
.50 (default)	.69	.32	.35	.25	.06	.57
.52 (optimal)	.74	.32	.42	.20	.06	.62
Null classifier	.62	0	.62	0	.62	0

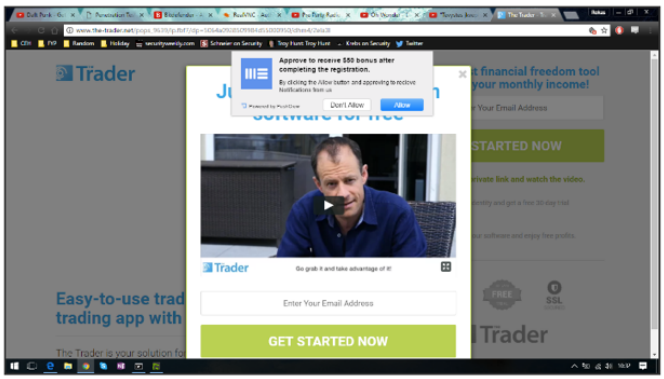
Table 10: H score detection efficacy classification performance for HaaSS sensor reports

suspected attacks that did not originate from the emulated attacks. 26 of these were correctly detected semantic attacks and 23 were incorrectly detected. Interestingly, multiple reports of typosquatting attacks were received, further validating the HaaSS concept for detecting and reporting an even wider range of semantic attacks. In Figure 18, an example typosquatting report is shown from HaaSS sensor 11, with a H score of 67% (who also reported the highest detection rate overall in the experiment).

Expert reviewer sandbox classification (Classification): Within the experiment, for each HaaSS sensor report made by participants, the computed H score was subject to automatic classification if the probability value hit the default defined upper (>.85) or lower (<.1) threshold in *Cogni-Sense*. However, unlike in the lab-based study in [24], where two HaaSS sensor reports for attacks 1.1 and 3.1 were automatically classified based on a 92% H score, during the experiment no reports qualified for automatic classification and therefore all were sent to the semantic attack sandbox for manual classification.

To evaluate the practicality of the *Cogni-Sense* semantic attack sandbox, we have invited an expert reviewer to manually classify each of the HaaSS reports. The expert reviewer recruited was a lead security operations centre engineer, with over ten years experience on security event and information monitoring platforms. Here the aim is to evaluate experimentally whether information supplied by the *Cogni-Sense* system and HaaSS sensors adequately informed accurate classification by report reviewers. From the results, it is clear that the expert reviewer classified each of the HaaSS reports with a high degree of accuracy and excellent precision to distinguish between HaaSS reports that were credible semantic attacks. According to the performance indicators, the semantic attack sandbox has proven its utility as an informative tool for manually classifying HaaSS reports in order to transform HaaSS attack detection into kinetic defence against the reported threats. In the case of the *Cogni-Sense* prototype, each correct detection would have resulted in an e-mail attack alert of the HaaSS report to HaaSS sensors. In a production system, the security enforcement module (SEM) could be expanded to URL and domain blacklisting and blocking specific file names using existing organisational security platforms (e.g., web proxy, anti-virus, firewall etc.).

Figure 18: HaaSS sensor report of typosquatting website received during the experiment



Report No. 66

Reporting User ID: 48

HaaSS Score [67%]

Report Date/Time: 29/05/2017, 10:32

Report Details/Comments:

While navigating to youtube.com I was redirected to scam website. It appears to be "financial" site which asks user for an email address to complete registration. The risk has been mitigated by closing the browser window.

Expert reviewer role	Accuracy	Precision	FP	FN
Security operations centre engineer	.87	.94	.04	.09

Table 11: Expert reviewer semantic attack sandbox report classification performance

HaaSS Score remodelling: During the course of the experiment, the *Cogni-Sense* prototype collected new HaaSS feature data as a result of exploitation records and report classification received via the semantic attack sandbox. In total, only 136 new observations were collected, which had no significant change in the HaaSS sensor detection rate and therefore H score remodelling was not carried out. In general, the data arrival frequency from HaaSS sensor reports is relatively low compared to that

of network alerts from an IDS, but is likely to gradually increase as a HaaSS sensor-base expands. With this in mind, unlike in streaming prediction models, the H score random forest rule-set will remain valid until a significant change in the detection distribution, or until a significant amount of data has been collected with the introduction of new predictor features; which would warrant remodelling.

Conclusion from case study results: The HaaSS sensors comfortably outperformed all technical defences for the attacks evaluated in the experiment. Apart from Outlook and Yandex email providers detecting attack 1.1 and 3.1 respectively, all other technical defences failed to report that a deception-attempt was detected or implement measures to prevent them from executing. An exception is Comodo Antivirus, which failed to detect the PDF file masquerading on the spear USB as an attack, but did run the file through a virtual machine sandbox as part of default behaviour, and as a result prevented execution. The Avast and AVG anti-virus products also scanned the PDF file masquerading executable, presumably because it was an unsigned executable and this is again default product configuration behaviour conducted in most modern-day anti-virus products. In both cases scanning found the file to be legitimate and therefore allowed the file to run (which is also arguably second-order deception from a user's perspective). The file itself should be deemed as malicious code, as it contained a system command to directly open a browser to a malicious URL. More importantly, and as expected, the cosmetic deception vector of the PDF file masquerading attack was completely undetectable and could not be prevented by the host system and or any anti-virus product tested in the experiment. For attacks 2.1A, 4.1 and 4.2, none of the technical defences identified any malicious behaviour or blocked any of the attacks. At no point did any HaaSS sensor report that their personal system's resident technical defences (e.g., anti-virus, browser security, firewalls etc.) either detected or improved their ability to identify any of the semantic attacks that were deployed in the experiment. On the contrary, HaaSS reports generally consisted of detection observation (in the reporting information) that described detection based on HaaSS sensor expertise and knowledge, rather than guidance or support from any available security software they may have had access to. The inadequacy of participants technical defences is further exemplified by the simple fact that many continued to be successful exploited by attack deception vectors, regardless of the security software installed on their device platform.

Overall, the HaaSS sensors were more effective at detecting all threats than the technical defences exposed to the semantic attacks; without prior knowledge of the attacks themselves or any specific semantic attack training provided prior to the experiment. Crucially, the detection performance of the HaaSS sensor base compared to technical defences has significant implications for implementing defence measures to improve semantic attack threat detection capability. For instance, in an organisational context, the 26 participants acting as employees within a HaaSS sensor role would have exhibited a missed detection rate below 10% for the semantic attacks evaluated, compared to a missed detection rate of 81% if only technical security systems had been used. These results strongly suggest that by involving the user within a security platform through active cyber threat detection and reporting, where technical defences struggle to provide adequate protection against deception-based threats, the HaaSS concept is shown to be both a viable and superior method for detecting semantic attacks.

6. Limitations

By its nature, the empirical experiment carried out has some limitations. It is possible that *Cogni-Sense* did not capture the full footprint of a HaaSS sensor's activity, because the the activity collection tool was not installed on other platforms that the participants were using in parallel, such as smartphones and work devices. By including all participant devices, this would increase the accuracy of the activity analysis and would have an impact on the H scores computed for each attack report. However, this was highly impractical for the experiment due to ethics considerations and the substantial cross-platform development required. Furthermore, the activity learning period was naturally limited in time (here, one month), which may not have been a representative or sufficient time period for feature collection and learning for all participants. In principle, the longer the learning period, the more confident the H scores would be. Regarding the prototype HaaSS platform *Cogni-Sense*, implementation limitations were identified by the difficulty in reporting attack 5.1, which initiates a semantic attack distributed through hardware with software interaction [4]. For HaaSS sensors that detected the USB as semantic attack in the physical space, this required HaaSS sensors to take an actual picture of the USB and then send this through image to their device to intiate a report. Therefore, for future HaaSS system platforms it would be essential to allow for capturing reports more naturally in physical space through use of mobile video and image capture capabilities.

In terms of the experiment's relatively small sample size (26 participants), it is quite possible that had the sample been larger the corresponding H score's accuracy could have been higher or indeed lower. However, generally, it is unlikely that a change in H score accuracy based on an increased sample size would unduly affect the utility of the H score as a core component of the HaaSS framework. In fact, this is generally expected because H score prediction performance (e.g., accuracy) and bias will dynamically change between organisations; especially over time. For example, the distribution of the H score will deviate as more reports are received and classified by a HaaSS platform. As a result, the H score will gradually adapt to the bias of different organisations' incumbent

user-base and their specific user profiles and HaaSS reporting telemetry (e.g., types of attacks seen). Therefore, a H Score modelled only on phishing attacks is likely to be less accurate when predicting HaaSS sensor detection efficacy for a file masquerading deception vector. In this experiment, we have shown that a generalised model can work sufficiently well for a participant sample (representing the size of a small company) from which it has not been originally trained; even for attacks is has not previously seen. In practice, an organisation is likely to begin with a base template model for the H score (e.g., the model developed in [5]), which is replaced gradually over time by data from internal HaaSS sensors (i.e., user-base).

7. Future work

Here, we have made progress towards the first technical framework in order to enable the design and implementation of HaaSS platforms, demonstrating the feasibility of the concept as a defence against semantic attacks. Whilst we have evaluated the concept of HaaSS for conventional computer systems, in the space-constrained interfaces of smartphones and embedded systems, the user is afforded a lot less information to spot suspicious activity. For instance, it is difficult to see the full address of a website, and SMS messages in modern smartphones are automatically grouped within one's existing conversations based on the phone number of the sender (even if this has been spoofed). As a result, Cogni-Sense HaaSS sensor reporting mechanisms should be expanded to include a range of system and device interfaces; especially as future semantic attacks will be designed to exploit IoT devices that interface with users through both physical and cyber means. By providing a richer facility for users to report suspected threats across in IoT space, Cogni-Sense can be expanded to address an even wider range of cyber-physical semantic attack threats that are expected to emerge in the near future.

The trustworthiness of HaaSS reporting information has been studied in relation to the reliability of human sensors of semantic attacks in the context of this work. However, malicious modification, prevention or delay of HaaSS reports can also be the result of cyber security breaches affecting the mobile devices and network infrastructure used to deliver HaaSS reports. Examples of these can be denial of service attacks, where the timely delivery of reports is important, and location spoofing attacks, where the accuracy of the location of an incident is important. Future work should aim to introduce the cyber-trustworthiness aspect in HaaSS and propose a mechanism for scoring reports in terms of their cyber-trustworthiness based on features of the HaaSS reporting device. We have previously conducted preliminary research for a mobile device's "cyber trustworthiness" as a HaaS reporting device for police incidents in [57]. Combining both approaches explores the concept of a unified measure of trust and reliability for HaaSS reports based on both the reliability of the HaaSS user and cyber-trustworthiness of their system.

Although our primary aim in this work has been to evaluate the applicability of the HaaSS concept for detecting and mitigating semantic attacks, more generally, HaaSS has practical uses beyond semantic attacks, such as in the detection of physical threats and adverse physical conditions [58, 59, 13]. It would be interesting to investigate the applicability of the paradigm for detection of other cyber threats, such as denial of service, reporting service interruption and degradation to determine user experience in near real-time, or even further exploring the concept of the H score in cyber insurance.

Finally, a direction for research may be to explore whether cyber security can benefit from "super-recognisers" in the same way policing does. These are human sensors employed by the police [2] to rapidly identify suspects through vast volumes of surveillance footage. In the context of our experimentation, this would mean identifying individuals in the population who would detect not only six out of six (e.g., H11 in Table 8), but perhaps hundreds of deception attempts without false positives or false negatives.

8. Conclusion

As building lasting and practical defenses against semantic attacks is a perpetual challenge, for which technical defences are simply not equipped to solve on their own, we have proposed a prototype for utilising the Human-as-a-Security-Sensor paradigm for detecting and mitigating semantic social engineering attacks. The framework proposed provides researchers and developers with a blueprint for the design and development of a technical HaaSS system. We have put the HaaSS paradigm to the test with an empirical case study across a range of semantic social engineering attacks, and compared against technical platforms that claim to provide defence against such attacks, as well as technical defence systems designed specifically to protect against them. In this respect, this first evaluation was successful, as the users in a HaaSS role performed considerably better than all technical defence systems evaluated, and the *Cogni-Sense* application developed for leveraging this ability of users proved fit for the purpose. We also demonstrated experimentally that the application of HaaSS under real-world conditions is indeed viable and practically useful means to dynamically detect semantic attacks.

By involving users as human sensors at the heart of a technical defence platform, we have challenged the concept that users are the weakest link against semantic attacks, instead, empowering them to become one of its strongest links for detecting deception-based threats. It is commonly emphasised that a single user being deceived by one such

[2] The term "super-recogniser" has been coined by the Metropolitan Police Service in the United Kingdom [60]

attack is sufficient to compromise an organisation's security. Here, we have flipped this notion. If a single user detects correctly an attack and can communicate it internally, then the organisation has successfully detected the attack.

References

[1] K. Mitnick and W. L. Simon. *The art of deception: controlling the human element of security*. Wiley, 2001.

[2] B. Schneier. Secrets and lies: digital security in a networked world. 2011.

[3] C. Hadnagy. *Unmasking the social engineer: The human element of security*. John Wiley and Sons, 2014.

[4] R. Heartfield and G. Loukas. A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks. *ACM Computing Surveys*, 48(3), 2016.

[5] R. Heartfield, G. Loukas, and D. Gan. You are probably not the weakest link: Towards practical prediction of susceptibility to semantic social engineering attacks. *IEEE Access*, 4:6910–6928, 2016.

[6] M. A. Sasse, S. Brostoff, and D. Weirich. Transforming the 'weakest link' - a human/computer interaction approach to usable and effective security. *BT technology journal*, 19(3):122–131, 2001.

[7] M. A. Sasse, C. C. Palmer, M. Jakobsson, S. Consolvo, R. Wash, and L. J. Camp. Helping you protect you. *IEEE Security and Privacy*, 12(1):39–42, 2014.

[8] University of Oxford. Information security - report an incident, 2016. URL `https://www.infosec.ox.ac.uk/report-incident`.

[9] BBC News. Fake news: Facebook rolls out new tools to tackle false stories, 2016. URL `http://www.bbc.co.uk/news/world-us-canada-38336212`.

[10] PhishMe. Phishme reporter, 2017. URL `https://phishme.com/product-services/reporter`.

[11] Wombat Security. Wombat security announces new feature to reinforce secure employee behavior against phishing, 2016. URL `https://www.wombatsecurity.com/press-releases/phishalarm-email-add-in`.

[12] Sophos. Sophos phish threat, 2017. URL `https://www.sophos.com/products/phish-threat.aspx`.

[13] M. Avvenuti, M. G. Cimino, S. Cresci, A. Marchetti, and M. Tesconi. A framework for detecting unfolding emergencies using humans as sensors. *SpringerPlus*, 5(1):1–23, 2016.

[14] Y. Zheng, T. Liu, Y. Wang, Y. Zhu, Y. Liu, and E. Chang. Diagnosing new york city's noises with ubiquitous data. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 715–725, Sep. 2014.

[15] E.H. Jürrens, A. Bröring, and S. Jirkai. A human sensor web for water availability monitoring. In *OneSpace*, 2009.

[16] O. Aulov and M. Halem. Human sensor networks for improved modeling of natural disasters. *Proceedings of the IEEE*, 100 (10):2812–2823, 2012.

[17] W. Yuan, D. Guan, E. N. Huh, and S. Lee. Harness human sensor networks for situational awareness in disaster reliefs: a survey. *IETE Technical Review,*, 30(3):240–247, 2013.

[18] D. Wang, M.T. Amin, S. Li, T. Abdelzaher, L. Kaplan, S. Gu, C. Pan, and H. Liu C.C. Aggarwaland R. Gantiand X. Wang. Using humans as sensors: an estimation-theoretic perspective. In *Information Processing in Sensor Networks, IPSN-14 Proceedings of the 13th International Symposium on*, pages 35–46. IEEE, June 2014.

[19] N. Stembert, A. Padmos, S. M. Bargh, S. Choenni, and F. Jansen. A study of preventing email (spear) phishing by enabling human intelligence. In *Intelligence and Security Informatics Conference (EISIC)*, pages 113–120. IEEE, 2015.

[20] L. Malisa, K. Kostiainen, and S. Capkun. Detecting mobile application spoofing attacks by leveraging user visual similarity perception. *IACR Cryptology ePrint Archive*, 2015.

[21] PhishMe. Phishme triage, 2017. URL `https://phishme.com/product-services/triage/`.

[22] PhishTank. Out of the web and into the tank, 2017. URL `https://www.phishtank.com/`.

[23] Millersmiles. Millersmiles anti-phishing services, 2017. URL `www.millersmiles.co.uk`.

[24] R. Heartfield, G. Loukas, and D. Gan. An eye for deception: A case study in utilizing the human-as-a-security-sensor paradigm to detect zero-day semantic social engineering attacks. In *Software Engineering Research, Management and Applications (SERA 2017), IEEE 15th Conference on*. IEEE, June 2017.

[25] M. A. Sasse and M. Smith C. Herley H. Lipford K. Vaniea. Debunking security-usability tradeoff myths. *IEEE Security and Privacy*, 14(5):33–39, 2016.

[26] Birch Grove Software Inc. Activtrak, 2017. URL `https://activtrak.com/`.

[27] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[28] L. Ross, L. Irani, M. Silberman, A. Zaldivar, and B. Tomlinson. Who are the crowdworkers? shifting demographics in mechanical turk. In *CHI'10 extended abstracts on Human factors in computing systems*, pages 2863–2872. ACM, 2010.

[29] M. Marge, S. Banerjee, and A. I. Rudnicky. Using the amazon mechanical turk for transcription of spoken language. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5270–5273. IEEE, 2010.

[30] D. W. Barowy, C. Curtsinger, E. D. Berger, and A. McGregor. Automan: A platform for integrating human-based and digital computation. *Communications of the ACM*, 59(6):102–109, 2016.

[31] W. S. Lasecki, C. D. Miller, I. Naim, R. Kushalnagar, A. Sadilek, D. Gildea, and J. P. Bigham. Scribe: Deep integration of human and machine intelligence to caption speech in real time. *Communications of the ACM*, 60(11), 2017.

[32] K. Krol, J. M. Spring, S. Parkin, and M. A. Sasse. Towards robust experimental design for user studies in security and privacy. In *Learning from Authoritative Security Experiment Results (LASER) Workshop*, 2016.

[33] Yahoo. Secure your inbox, 2017. URL `https://uk.antispam.yahoo.com/`.

[34] Engadget. Google beefs up gmail security to fight phishing attempts, 2017. URL `https://www.engadget.com/2017/05/31/google-gmail-security-fight-phishing/`.

[35] Microsoft. Office 365 email anti-spam protection, 2017. URL `https://support.office.com/en-us/article/Office-365-email-anti-spam-protection-6a601501-a6a8-4559-b2e7-5`

[36] ProtonMail. Effective spam filtering with encrypted email, 2017. URL `https://protonmail.com/blog/encrypted-email-spam-filtering/`.

[37] ESET. Eset anti-phishing, 2017. URL `https://www.eset.com/us/anti-phishing/`.

[38] GMX. Spam filter: The cleanest inbox, 2017. URL `https://www.gmx.com/mail/spam-filter/`.

[39] Mail. Stay safe from phishing: Your worry free email, 2017. URL `https://www.mail.com/mail/spam-filter/499562-stay-safe-phishing-email.html`.

[40] Firefox. How does phishing and malware protection work, 2017. URL `https://support.mozilla.org/en-US/kb/how-does-phishing-and-malware-protection-work`.

[41] Chrome. Google safe browsing, 2017. URL `https://safebrowsing.google.com/`.

[42] Opera. Opera fraud and malware protection, 2017. URL `http://www.opera.com/help/tutorials/security/fraud/`.

[43] Comodo. Dragon internet browser, 2017. URL `https://www.comodo.com/home/browsers-toolbars/browser.php#tab-features`.

[44] Avast. Safezone browser, 2017. URL `https://www.avast.com/f-safezone`.

[45] Microsoft. Secucrity enhancements for microsoft edge, 2017. URL `https://docs.microsoft.com/en-us/microsoft-edge/`

deploy/security-enhancements-microsoft-edge.
[46] Apple. Defending your online privacy and security., 2017. URL https://www.apple.com/uk/safari/.
[47] Comodo. Comodo cloud antivirus, 2017. URL https://antivirus.comodo.com/cloud-antivirus.php.
[48] AVG. Avg anti-virus free, 2017. URL http://www.avg.com/en-gb/free-antivirus-download.
[49] Avast Internet Security. Avast internet security, 2017. URL https://www.avast.com/en-gb/internet-security.
[50] Microsoft. Windows defender smartscreen, 2017. URL https://docs.microsoft.com/en-us/windows/threat-protection/windows-defender-smartscreen/windows-defender-smartscreen-overview.
[51] Symantec. Norton security review 2017: Top antivirus provider with fully furnished internet security suites, 2017. URL https://fatsecurity.com/review/norton.
[52] Kaspersky. Kaspersky internet security 2017, 2017. URL https://www.kaspersky.co.uk/internet-security.
[53] Sophos. Intercept x tech specs, 2017. URL https://www.sophos.com/en-us/products/intercept-x/tech-specs.aspx.
[54] Facebook. What can i do about phishing?, 2017. URL https://www.facebook.com/help/166863010078512?helpref=faq_content.
[55] TipTopSecurity. Is google drive safe to use? how google secures your files online, 2016. URL https://tiptopsecurity.com/is-google-drive-safe-to-use/.
[56] Microsoft. Mitigate threats by using windows 10 security features, 2017. URL https://docs.microsoft.com/en-us/windows/threat-protection/overview-of-threat-mitigations-in-windows-10.
[57] S. S. Rahman, R. Heartfield, W. Oliff, G. Loukas, and A. Filippoupolitis. Assessing the cyber-trustworthiness of human-as-a-sensor reports from mobile devices. In *Software Engineering Research, Management and Applications (SERA 2017), 15th ACIS International Conference on*. IEEE, June 2017.
[58] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 344–353. ACM, 2013.
[59] R. Dave, S. K. Boddhu, M. McCartney, and J. West. Augmenting situational awareness for first responders using social media as a sensor. *IFAC Proceedings Volumes*, 46(15):133–140, 2013.
[60] David J Robertson, Eilidh Noyes, Andrew J Dowsett, Rob Jenkins, and A Mike Burton. Face recognition by metropolitan police super-recognisers. *PloS one*, 11(2):e0150036, 2016.