A Taxonomy of Attacks and a Survey of Defence Mechanisms for Semantic Social Engineering Attacks

RYAN HEARTFIELD, University of Greenwich
GEORGE LOUKAS, University of Greenwich

Social engineering is used as an umbrella term for a broad spectrum of computer exploitations that employ a variety of attack vectors and strategies to psychologically manipulate a user. Semantic attacks are the specific type of social engineering attacks that bypass technical defences by actively manipulating object characteristics, such as platform or system applications, to deceive rather than directly attack the user. Commonly observed examples include obfuscated URLs, phishing emails, drive-by downloads, spoofed websites and scareware to name a few. This paper presents a taxonomy of semantic attacks, as well as a survey of applicable defences. By contrasting the threat landscape and the associated mitigation techniques in a single comparative matrix, we identify the areas where further research can be particularly beneficial.

Categories and Subject Descriptors: 500 [**Security and Privacy**]: Social Engineering

General Terms: Security, Social Engineering

Additional Key Words and Phrases: Computer Crime, Social Engineering Attacks, Semantic Attacks, Survey

ACM Reference Format:
Ryan Heartfield and George Loukas, 2015. A Taxonomy of Attacks and a Survey of Defense Mechanisms for Semantic Social Engineering Attacks. *ACM Comput. Surv.* 0, 0, Article 0 (2015), 38 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

In information security, the user is often seen as the "weakest link" [Schneier 2011] because even the strongest technical protection systems can be bypassed if the attacker successfully manipulates the user into divulging a password, opening a malicious email attachment or visiting a compromised website. The term most often used for this process is *Social Engineering*, but this does not differentiate attacks that bypass the security of computer systems from those that can be observed in a non-technical arena, such as prize-winning letter scams [Cluley 2011] or physically impersonating an authority figure [SocialEngineer 2013]. To differentiate from these, researchers have proposed the term *Semantic Attack* [Schneier 2000; Jordan and Heather 2005; Thompson 2007]. As semantics is the study of meaning and symbolisation, in the context of social engineering a semantic attack is one that manipulates the user-computer interface to deceive the user and ultimately breach the computer system's security. We propose the following definition:

Semantic Attack. The manipulation of user-computer interfacing with the purpose to breach a computer system's information security through user deception.

Author's addresses: R. Heartfield and G. Loukas, CSAFE Centre, Department of Computing and Mathematical Sciences, University of Greenwich.

DOI: http://dx.doi.org/10.1145/0000000.0000000

Attack Family	Exploits
Phishing	Email, Website, URL, IM, Forums, SMS IRC
File Masquerading	Office Document File, Application File, System File
Application Masquerading	Scareware, Ransomware, Rogueware
Web Pop-Up	Media Plugin, Error Message, Bogus Questionnaire
Malvertisement	Infected Ad, One Click Fraud, Download Button
Social Networking	Friend Injection, Fake Video Links, Game Requests
Removable Media	USB, Flash/SD, CD/DVD
Wireless	Rogue AP, Rogue RFID

Table I: Examples of Semantic Attack Exploits

Semantic attacks have been characterised as a technical enigma for information technology [Abraham and UnduShobha 2010]. That is because the user-computer interface is at the same time their only boundary and their primary window of opportunity. The expanse of the problem space and variation of attacks are extreme (from state-backed Advanced Persistent Threats employing multi-stage/platform attack vectors to script kiddies generating automated phishing emails), obscure (defining what constitutes a semantic attack through its component structure) and arbitrarily difficult to detect (as attack vectors primarily address the user rather than the technical system). They can be technically basic [Jordan and Heather 2005; Dhamija et al. 2006; Drake et al. 2004], highly complex [Huber et al. 2010; Marforio et al. 2011] or a combination of the two [Heartfield and Loukas 2013; Selvaraj and Gutierrez 2010]. Defence mechanisms have been proposed at scientific research level to target exploitations such as website and phishing attacks [Aburrous et al. 2008; Chou et al. 2004; Huang et al. 2009; P. Kumaraguru and Nunge 2007; Kumaraguru 2009] as well as at commercial level [FirstCyberSecurity 2009; Webroot 2013]. However solutions such as these have not addressed the wider semantic attack problem space, and mitigations tend to be specific to given exploits; static and disparate. Without the dynamic capabilities observed in other defence systems such as antivirus and firewalls, they are often overcome by attackers who subtly adjust their approach and render exploit-driven detection mechanisms no longer relevant.

Semantic attacks are typically grouped by type of exploitation into specifically related attack families. For instance, phishing has become synonymous with fake websites and emails [Dhamija et al. 2006], whereby victims are targeted through a baiting mechanism [Atkins and Huang 2013]. In this example, the attack family is phishing, whereas phishing emails and phishing websites assume the exploitation type. Another example would be fake applications posing as legitimate pieces of software (scareware, rogueware [Corrons 2010] etc.), which can be associated with the Application Masquerading attack family. A majority view of currently observed semantic attacks is presented in Table I, which combines terminologies commonly used by information security practitioners to identify semantic attack exploitations into related attack families. This static approach fails to identify common attributes of exploitations that are grouped under different attack families. For example, a phishing URL on a web page or email may share a similar method of obfuscation/deception with the one used in a bogus video URL on a social network site (SNS). So, a defence mechanism that would defeat the particular method would potentially be useful for both types of semantic attacks.

Attack-specific approaches to classification have historically led to development of computer security systems that rely on manually generated attack signatures and definitions [Christodorescu and Jha 2004], which can be circumvented in a number of ways (e.g. by zero-day vulnerabilities, obfuscation, polymorphism [Bilge and Du-

mitras 2012] etc.). Through taxonomic research and development, defence systems have employed techniques that analyse relationships between application behaviour and response (sandboxing [Blasing et al. 2010; Greamo and A.Ghosh 2011], dynamic anomaly based scanning [Tavallaee et al. 2010] etc.). These systems have enabled dynamic and proactive response to security threats on multiple technical platforms, from mobile to desktop operating systems. Yet, this approach has not been replicated or realised for semantic attacks.

A taxonomy of the semantic attack problem space can help researchers evaluate the applicability and scope of proposed solutions for different current and future threats. Some first steps have already been taken in this direction, but still to a limited extent. For instance, Foozy et al. [2011] have classified attacks based solely on the primary attack vector, such as whether it is a website or email, while Ivaturi and Janczewski [2011] have used a hierarchical classification structure that links specific attack vectors to the conditions of their delivery, specifically person-to-person or via online media. These approaches cannot identify attacks and their associated behaviour outside of the criteria specified and therefore may be unable to address future semantic attacks or ones that combine exploitations from more than one categories.

More recently, Tetri and Vuorinen [2013] introduced a dynamic classification framework that addresses high-level attack behaviours involved in the creation of a semantic attack: persuasion, fabrication and data gathering. This approach provides a basis for classifying threats outside of the limited scope of previous taxonomies, but is limited to high-level theoretical dimensions of Social Engineering and it is not clear how it can be used at a technical level by developers of semantic attack defence technologies. On the other hand, the work of Algarni et al. [2013] is highly useful in that respect as it identifies the entities that comprise an attack irrespective of visual representation, but their focus is solely on social networking.

2. TAXONOMY OF SEMANTIC ATTACKS

Our aim is to help researchers and engineers develop technical defence approaches for both current and future semantic attacks by addressing core semantic attack characteristics rather than particular implementations. That is because a defence system designed for a particular set of characteristics can be applicable across all attack types that share it, thus making it a more efficient choice from the perspective of technical development.

Note that current cybercrime operations may have many layers and an adversary may employ composite semantic attacks consisting of multiple phases of individual semantic attacks, in parallel or one after the other [Sood and Enbody 2014]. An example may be a spam email containing a URL to a spoofed website, where the user is forwarded to a drive-by download that results in infection with a scareware application. Our taxonomy classification is designed to classify semantic attacks as individual singular components (e.g., the drive-by download) rather than the possible permutations of these components (spam $\rightarrow$ spoofed website $\rightarrow$ drive-by download $\rightarrow$ scareware). This is an important function of developing a generic taxonomy of attacks that is practically usable at a technical level because it simplifies the objective of a defence mechanism. Assuming that a composite attack requires all its individual component attacks to succeed, mechanisms that address any of the latter would protect against the composite attack too. In the example above, the threat of the scareware application delivered in this manner would be thwarted by any technical mechanism that would effectively block the spam email, spoofed website or drive-by-download attempt.

To be usable in the long-term, these characteristics need to be universally applicable and independent of the platform involved. In particular the platform consolidation driven by cloud computing, the Internet of Things and other recent computing

paradigms is making the distinction between mobile, desktop and embedded system user experience less obvious than in the past. For example, in the near future, fake tyre pressure alerts shown on a car's dashboard [Rouf et al. 2010] may be used to achieve deception in a manner not too dissimilar to current scareware pop-up alerts for mobile and desktop users. A defence approach focusing on the fake warning deception characteristics rather than the platform targeted would potentially be applicable for both.

These characteristics also need to span all stages of an attack. For this, we adopt the definition of the three distinct control stages of orchestration, exploitation and execution suggested by [CESG 2015]. For each stage we pose the questions that we believe would matter the most to developers of technical protection mechanisms. In addition, we have ensured that for each category, the classifications are mutually exclusive.

Based on this constraint we do not by any means claim that the taxonomy is exhaustive, but we do evaluate its practical usefulness by providing for each category an example of how a developer can find it useful.

Control Stage 1: Orchestration

(1) How is the target chosen?
This can help identify the conditions for exposure to an attack. For instance, whether a user is vulnerable due to a specific attribute of theirs or chosen randomly makes a difference for the user and system features that a defence system may have to focus on.
(2) How does the attack reach the target?
This may help identify the platforms that are involved in the attack, which in turn may help a developer choose which remote (hence involving a network) or local system to monitor and potentially where to place a defence mechanism.
(3) Is the attack automated?
The degree of automation in an attack can change fundamentally the response mechanism or the type of data that can be meaningful to collect about it. For instance, a fully automated attack may be possible to fingerprint based on patterns of previously observed behaviour, while a fully manual attack may focus on the attacker's behaviour instead.

Control Stage 2: Exploitation

(1) Is it looks or behaviour that deceive the user?
This can potentially help the developer of a defence mechanism to pinpoint mechanisms by which an attacker can deceive the user into a false expectation by manipulating visual and/or system behaviour aspects of a system.
(2) Is the platform used in the deception only (ab)used or also programmatically modified?
Identifying whether the deception occurs in code (embedded in the system or external), or by abusing intended user space functionality, can help shape the design of a defence system by narrowing down its scope.

Control Stage 3: Execution

(1) Does the attack complete the deception in one step?
An attack that relies on more than one step can be potentially detected more easily than a single-step one and before it completes by looking for traces of its initial steps. It may also be thwarted by preventing even one of the compromising actions that a user needs to be deceived into committing.
(2) Does the deception persist?

Contrary to one-off deception attempts, persistent ones may have a high chance of succeeding in their target but could also help a learning-based defence system to gradually identify its pattern of behaviour and block it.

Each category of answers that correspond to each question helps establish the sections and subsections of the taxonomy, as shown in Fig. 1.

Utilising a single-layer attack model for classification, our taxonomy is able to identify the composition of a semantic attack by using a parameter-based approach, organising the classification criteria in a linear fashion. Unlike previous taxonomy approaches, it captures multiple variables involved in the delivery and execution of a semantic attack by applying criteria that are independent of the attack vectors used. As a result, it can potentially represent both existing and future attacks.

To supplement the taxonomy with produce survey of the current landscape of defence mechanisms, as well as a comparative matrix between the classification criteria and applicable defences, which illustrates areas of semantic attack security most in need of research.

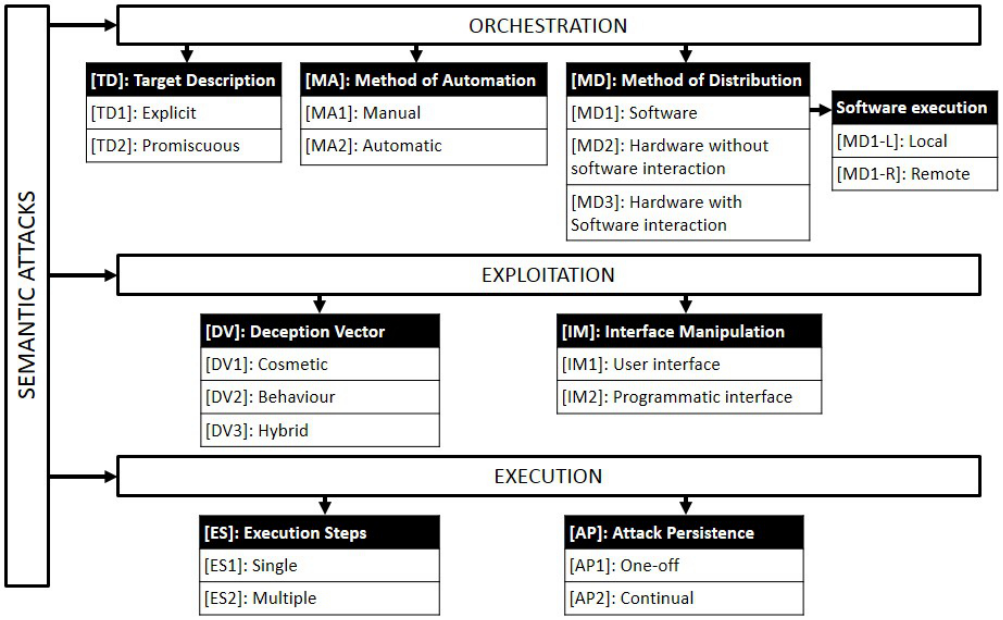

Fig. 1. Taxonomy of Semantic Attack Mechanisms

2.1. Control Stage 1: Orchestration

The arrangement of targeting, distribution and automation in a semantic attack.

TD: Target Description

This refers to the targeting methodology applied in the attack orchestration process: whether the attacker has chosen to target an individual or group randomly (promiscuous targeting) or based on their identity or some other exclusive attribute of theirs (explicit targeting).

TD1: Explicit targeting. By explicit targeting, we refer to the practice of choosing a particular individual or group of users as the targets of an attack based on their specific identity or an exclusive attribute of theirs (company, role, location etc...). Typical examples here are spear-phishing and watering hole. Spear-phishing is the targeted version of phishing, where a carefully crafted phishing email is directed to a specific individual or organisation. This is known as whaling when the target is highly valuable (e.g., a senior executive) [Orman 2009]. Watering hole is the targeted version of the drive-by-download attack, where a malicious script is implanted on the websites that a particular individual or community are known to visit [RSA 2012]. In the same category of explicit targeting based on a common attribute we can include the practice of targeting victims located in a particular country, so as to intentionally cultivate botnets of better quality or to exploit that country's sociopolitical affairs [Chaffin 2014]. For emphasis, let us clarify that we include here only attacks where the target is chosen explicitly for their identity or for an exclusive attribute of theirs, not merely as a second order of consequence. For example, a WiFi phishing attack in a coffee shop is generally not a targeted attack (see TD2: promiscuous targeting). The fact that people that are coffee drinkers are targeted is only a second order of consequence, as the attacker is highly unlikely to have explicitly chosen them because they are coffee drinkers. However, one could conceive a targeted instance of this attack (say a "WiFi spear-phishing" attack) launched specifically when a particular target individual is known to visit that coffee shop. This hypothetical attack would indeed be classified as explicitly targeted.

TD2: Promiscuous targeting. Promiscuous targeting is nondescript and typically focuses on maximum exposure. A typical example would be a large-scale phishing campaign launched by a botnet against any email address the attacker could find [Banday et al. 2009]. Semantic attack worms employing instant messaging and file-sharing usually also aim for maximum exposure through promiscuous targeting [Sharma 2011]. Wifi phishing attacks [Hwang et al. 2008; Briones et al. 2013] are generally promiscuous in their user targeting. The attacker sets up a malicious access point to exploit any incoming client connection regardless of user identity or any other attribute.

MD: Method of Distribution

This refers to the mechanism by which an attack arrives at the target system: through a software interface (Software), executed on the target host (Software-Local) or on a distributed host environment (Software-Remote); or whether it is a physical device (Hardware) that interfaces with the target host via direct hardware access (Hardware without software interaction) or via the host operating system (Hardware with software interaction). Software interfaces are generally more common. Software-based attacks can be highly automated and easy to reuse and replicate with minimal supervision, while embedding a hardware device in a target environment can be manually intensive, high-risk, and difficult to target large number of users.

MD1: Software. This refers to semantic attacks that distribute a deception mechanism through a software medium, usually exploiting built-in functionality in the user interface of legitimate applications rather than technical flaws. They can be classified as local or remote.

MD1-L: Local. Local software attacks are distributed from within the target user's local operating system, as exemplified by attacks that use Trojan horse logic [Emm 2005]. While a Trojan horse carrying a semantic attack payload may itself have been distributed remotely (e.g., unknowingly downloaded from a website), the semantic attack already resides within the target machine when presented to its target [Corrons

2010]. This behaviour is commonly observed in PDF file masquerading [Bindra 2011] and in scareware/rogueware attacks [Hwang et al. 2008; Corrons 2010].

MD1-R: Remote. Here, the semantic attack originates from a host environment other than the target user's system. It may involve web servers or distributed applications, such as cloud (for storage, server hosting or email) and peer-to-peer platforms (P2P networks, instant messaging, torrents). Remote distribution varies between different components in a remote platform. For example, a malicious URL may be presented via an email system on the Internet [Drake et al. 2004] or an instant messaging program [Leavitt 2005; Lauinger et al. 2010]. Malicious files can be served via synchronised cloud storage [Heartfield and Loukas 2013], malvertisements embedded in compromised web pages [Li et al. 2012], automated bots in social network sites [Boshmaf et al. 2011; Coburn and Marra 2008] or a stolen online gaming avatar used to gather information on a computer gaming system [Podhradsky et al. 2013].

MD2: Hardware without software interaction. This refers to attacks distributed to the target through a local hardware interface without interacting with the target host's operating system. Examples include hardware interfaces capable of direct memory access (DMA) or external hardware devices that passively intercept user data. In the case of DMA exploits, attackers plant common peripherals, such as storage devices, employing only hardware interfaces that are capable of creating a direct channel to a system's physical memory, such as firewire [Boileau 2006] or a PCI network interface card [Duflot et al. 2011]. Attackers can also plant man-in-the-middle sniffing devices in the guise of Ethernet network outlet points, or hardware key loggers embedded in keyboards [Keelog 2015].

MD3: Hardware with software interaction. Most hardware-distributed semantic attacks rely on a software component in order to elicit compromising user actions. For example, Anderson and Anderson [2010] describe a series of attacks involving the USB interface, presented as a masqueraded hardware device that includes a software component, such as a malicious file triggering the autorun functionality. In the same category we can include exploitations that depend on WiFi, Bluetooth, as well as other less obvious interfaces for their distribution. An interesting example is near field communication (NFC) phishing. When attached to a legitimate surface (say a Red Cross donation appeal poster), a maliciously modified NFC tag interacts with the NFC software on users' smartphones and directs them to a malicious website rather than the Red Cross's. [Madlmayr et al. 2008].

MA: Mode of Automation

Mode of automation refers to the degree of attacker supervision in the activation and administration of a semantic attack: by manually supervising every element of the attack throughout its lifetime (Manual), pre-configuring every aspect and having no control over the attack after it is launched (Automatic) or by pre-configuring every aspect of it, yet being able to manually adjust it after initiating it through some form of live command and control (Semi-Automatic).

MA1: Manual. This requires intervention from the attacker to facilitate delivery of a semantic attack. The attacker explicitly triggers the deception mechanism by manually placing it in the target environment and, where applicable, explicitly responds to events when user interaction occurs. The attack procedure is flexible, because the attacker has full control over the process and may modify it during the attack. Examples could include an attacker manually sending a phishing email from a spoofed email account rather than an automated botnet, handing out infected DVDs outside a

building, or physically intercepting legitimate media devices and embedding into them a malicious payload before they reach their target [Fisher 2015].

MA2: Automatic. Here, the semantic attack is delivered in an automated manner without requiring communication or intervention from the attacker. The attacker has preprogrammed the procedure for delivering the attack. Automatic mode of automation provides obfuscation of the attacker as the semantic attack operates without external influence, but also limits control and flexibility. For example, in the case of the Anna Kournikova worm, the malicious email and file attachment contained the complete set of instructions for the attack [Chen 2003]; once released the author had no control over its targeting or attack parameters. A more recent example is "Selfmite", a SMS malware for Android devices, which contains a hard-coded procedure to send text messages containing a link to an app. Once installed once, it then replicates the message to the first 20 contacts on that device, effectively masquerading as a text from a known associate in an automated fashion [Ducklin 2014b]. In the same category we can include drive-by download attacks. Once an attacker embeds malicious code in a vulnerable website, the drive-by download runs automatically when a user visits the website [Mavromatis and Monrose 2008; Provos et al. 2009]. In general, full automation can help hide the origin of a semantic attack and reduce the effort required to replicate it, but also limits control over the process. Most automatic semantic attacks feature crude deception mechanisms that can be noticed by an experienced user.

2.2. Control Stage 2: Exploitation

The construction and application of attack vectors designed to bypass system information security.

DV: Deception Vector

The deception vector is a focal point of this taxonomy. It defines the mechanism by which the user is deceived into facilitating a security breach, and can be categorised as cosmetic, behaviour-based or a combination of the two.

DV1: Cosmetic. In the use of graphical user interfaces (GUIs), there is an implicit trust between GUI designer and user. The GUI designer trusts that the GUI will be used as intended, and the user trusts that each component is what it appears to be. Cosmetic semantic attacks exploit this trust by manipulating the appearance of GUI components. For example, a file contains a name, type extension, icon and associated program. Filenames typically appear as readable, clear indicators of what a file does or contains, as is often required by the author. Therefore, official-looking and recognisable filenames command a level of trust. The attacks in [Johnson et al. 2008] and [Shin et al. 2006] are prime examples where cosmetic deception techniques have proven particularly effective in P2P applications, either by matching filenames in a user's shared folder or by generating commonly searched filenames on the network. File extensions carry a presupposed risk associated to their behaviour. For example, ".exe" is a well known executable extension and thus threat to the user's system if the file is malicious. However, the same caution is not exercised for other executable types, such as ".bat", ".ini", ".lnk", or ".scr" [Heartfield and Loukas 2013]. Moreoever, in Microsoft Windows, the file type and appropriate icon to be displayed are determined by extension. As Windows Explorer hides extensions by default and executable programs can be configured to display any icon [Yee 2005], users can be deceived by an icon - extension mismatch if the filename is disguised, as in "document.txt.exe" displayed as "document.txt". In the same manner, emails contain sender addresses, subject titles, body of text containing images and attachments [Drake et al. 2004], and websites use a domain name URL and common page features, such as buttons and images, which

can be manipulated to deceive a user. Attackers can generate phishing websites that assume the identity of their legitimate counterparts by copying the source code and images to appear completely identical [Dhamija et al. 2006]. In typosquatting attacks, simple but effective cosmetic deception is achieved by registering domain names that are similar to popular, legitimate websites, and are likely to be visited by users making spelling mistakes (e.g. twiter.com or facebok.com) [Agten et al. 2015].

Cosmetic deception is prevalent in attacks against web-based user interfaces, such as social media platforms where an attacker posts malicious links that visually imitate videos, enticing users to click on them to view the content [TrendMicro 2014]. In WiFi phishing, the attacker redirects users to an authentication website that looks like the login page for free Internet service [Song et al. 2010]. The GUI components expected for such a page are included to add integrity and conformity to the exploit. Felt and Wagner [2011] provide examples of techniques for developing fake mobile apps that appear like their legitimate counterparts, by duplicating login screen, including the logo, GUI layout and text. The study carried out by J.Corbetta et al. [2014] has demonstrated visual logo spoofing, with manipulated "seal images", which fool the users but are otherwise seen as non-malicious content to a security system monitoring the web platform.

DV2: Behaviour. Here, deception is achieved by mimicking a legitimate system's behaviour rather than looks. Users are duped by supposed functionality convention against the standard approach used in well-known implementations. For example, in the generic rogue access point attack, the user is deceived by merely seeing it on the list of available WiFi networks [Hwang et al. 2008]. Also, in URL phishing attacks on social network sites [Leavitt 2005; N. Wang and Grossklags 2011], the deception relies solely on the accepted behaviour associated to the origin of the attack, as it is received from a user that is on the target's list of friends. In reality, the message is automatically generated by a malicious app installed on that friend's account.

DV3: Hybrid. Hybrid approaches combine aspects of both "looking like" (DV1) and "behaving like" (DV2) to create a convincing deception, as exemplified by phishing websites, which typically copy not only images and text, but also the actual code from the legitimate website [Dhamija et al. 2006]. Corrons [2010] and Giles [2010] describe how scareware applications masquerade as antivirus programs by using similar logos, text and interfaces as their legitimate counterparts, complete with scanning engines and infection removal tools. Paganini [2014] has reported an attack where a wrapper was attached to a legitimate mobile banking app. Once installed, it would load an in-app HTML page mimicking both the looks and the behaviour of the legitimate application's login form. On entering user credentials, an error would be displayed, requesting the user to install the application again. At that point, the legitimate version would indeed be installed and would function as expected from then on.

Other examples of hybrid deception can be seen in removable media attacks, such as USB flash drives. A USB storage device can be preloaded with a legitimate looking Microsoft word file that actually contains a malicious Visual Basic macro or zero-day exploit. Once the user has clicked on the file, Microsoft word opens as the user would expect, whilst also running the malicious code attached to the word document. A recent demonstration by Nohl and Lehl [2014] has shown how USB firmware controllers can be reprogrammed to spoof other devices to take control of a system, exfiltrate data or spy on the user. The exploit shows how a USB flash drive can assume different or multiple identities on a computer, such as a webcam, keyboard, as well as a USB flash drive, by manipulating USB interface identifiers used by operating system to interact with the USB devices functionality (e.g., processing key states sent by a USB keyboard peripheral). Here, the deception vector is both cosmetic and behavioural. The USB

device appears and functions as the user would expect, but in addition it deceives the host system into thinking it is a different device and allows the execution of arbitrary code.

IM: Interface Manipulation

A deception vector can be exploited by only utilising maliciously the existing functionality of the target system (user interface) or by also programmatically modifying it (programmatic interface). For example, an attacker may direct users to an attack server by merely posting a hyperlink using a social network site's user interface or by programmatically injecting malicious javascript code into its HTML code.

IM1: User Interface. Here, we refer to attacks that are limited to abusing existing functionality provided by a user interface as the means of deceiving a user. This includes both hardware and, more commonly, software user interfaces. In malvertisement exploitations [Li et al. 2012], attackers (ab)use the user functionality of an advertisement system on a website, by first posting a legitimate advertisement and then replacing it with a malicious version once it has gained popularity/trust on the host platform. In search-engine poisoning, attackers repeatedly place unrelated phrases in the body of text or URLs on a website, which when crawled by a search engine, return the attackers website in the results of a user search [Sullivan 2008; Howard and Komili 2010]. In man-in-the-middle rogue access points, the attacker adjusts the user configurable service set identifier (SSID) to match that of a legitimate access point in the environment in which it has been installed [Hwang et al. 2008].

IM2: Programmatic Interface. Here, we refer to attacks that are not limited to utilising existing functionality, but may also modify it, typically by exploiting vulnerabilities on the target system. This is prevalent in drive-by download attacks where the attacker injects malicious scripts (e.g. Javascript) into vulnerable websites to redirect users to a malicious download/installation [Egele et al. 2008; Cova et al. 2010].

A technique detailed by [Krishna 2011] involves compressing into a zip file an executable with special hexadecimal characters in the filename, so as to make it look like a Microsoft Word document when accessed with a particular popular zip archive viewer that has this technical flaw. Another example has been presented in [Selvaraj and Gutierrez 2010] and [Bindra 2011], where the PDF language is manipulated to create a file that appears and behaves as a legitimate PDF file would, but when executed can launch a semantic attack requesting login credentials and redirecting the user to a malicious website. This is achieved by using certain special functions in the PDF language that once compiled run on file execution.

2.3. Control Stage 3: Execution

Operational procedure during attack run-time

AP: Attack Persistence

Different semantic attacks implement varied levels of deception that are one-off or persist depending on the intentions of the attacker.

AP1: One-off. Here, once the user has triggered the attack payload, e.g. by performing an action that either grants access privileges or provides sensitive user data to the attacker, the SA ceases any further action. Large-scale spamming is typical of this approach. Attackers mass mail a large address set aiming to capture user details through a spoofed website, which once gathered it redirects users to the legitimate site and disappear [Drake et al. 2004; Dhinakaran et al. 2009]. Similarly, a drive-by-download attack completes on the download of the malware when the user accesses the infected

website. The user remains vulnerable to the same infected website on further attempts to access it, but each time the drive-by-download attack completes on download.

AP2: Continual. In continual semantic attacks, a particular attack instance does not expire upon successful exploitation and the user continues to be exposed to its deception attempts. This may involve reoccurring and direct communication with the target by exploiting the messaging mechanisms provided by email, instant messaging, SNS, P2P networks, IRC forums and other Internet-based services [Drake et al. 2004; Leavitt 2005; Shin et al. 2006]. A characteristic example would be scareware applications, which periodically display fake malware alerts for as long as they reside on the system [Giles 2010]. Continual behaviour is common in semantic attacks that are designed to elicit financial gain from a target.

ES: Execution Steps

A single semantic attack may consist of one or more distinct steps.

ES1: Single-step. Single-step attacks require that the target user carries out only one action to facilitate exploitation (e.g. open a file, run a program, click on a web link or button). Once the initial action from the victim has been carried out, the attack payload is able to gain the necessary information or control for the attacker, at which stage the semantic attack is complete. Variations of the phishing attacks described in [Drake et al. 2004] rely on a single step. The less user interaction required the lower the risk of being detected. This is one of the reasons that can be attributed to the rise of drive-by malware, where the user's system is infected with a single click [Cova et al. 2010]. Other examples of single-step attacks can be identified in typical website and email-based phishing attacks [Dhamija et al. 2006], and more recently in malvertisements [Li et al. 2012]. In the latter, the attackers place legitimate advertisements on trusted websites for as long as needed to gain good reputation and be trusted themselves. At that point, the attackers start placing advertisements that carry malware and infect all users visiting websites that display them.

ES2: Multi-step. In a multi-step attack, a user needs to be deceived more than once for the attack to achieve a meaningful result. Attacks involving multiple steps within the exploitation generally employ one step to gain access and an additional one to steal information or deliver malware. For example, in WiFi phishing, where the attacker generates a spoofed SSID of the local WiFi provider (e.g. a coffee shop's or hotel's), the user need to first connect to this rogue access point (step 1), and then to provide his/her login credentials to the attacker (step 2) after he/she is redirected to a phishing login page for access to the Internet [Song et al. 2010]. Similarly, in the usual typosquatting attack [Szurdi et al. 2014; Agten et al. 2015], the user must first navigate to the spoofed webpage with the mistyped domain name (step 1), and then click on a malformed download link (step 2). In [Jagatic et al. 2007], the phishing attack contains three separate steps for data gathering, payload distribution and deception.

Most semantic attacks that exhibit continual attack persistence (AP2) are multi-step. For instance, scareware [Giles 2010] typically infect the target system via deception (step 1) and then periodically attempt further deceptions through fake alerts of malware detected in the system (a repetitive step 2).

2.4. Taxonomy Examples

Here, let us illustrate the use of the taxonomy with four detailed examples and a table summarising 30 different attacks that have been observed in the wild.

In a drive-by download attack (Figure 2), an attacker needs to first assess a website for vulnerabilities that can be exploited to embed malicious behaviour. Once a weak-

ness is found, the attacker proceeds to insert malicious code into the website that will redirect a user into downloading malware. Attack automation is automatic (MA2) because it requires no intervention from the attacker after the malicious code has been embedded in the vulnerable website weaknesses. The software distribution is remotely presented to the user, on user navigation, via the infected website (MD1-R). Targeting is promiscuous (TD2) as the attacker has planted the exploitation to attack any user who visits the website (but there is also a targeted - TD1 - version of this attack: watering hole). The deception vector can be classified as hybrid because the website itself is legitimate and thus appears and behaves as expected (DV3). The attacker has programmatically manipulated the application platform (IM2) to embed the attack payload. The attack requires only one step of deception, which is that the user performs an action, such as clicking on a button, for the exploitation to occur (ES1). After this, the attack completes (AP1), but, of course, the user can be infected again by performing the same action again.

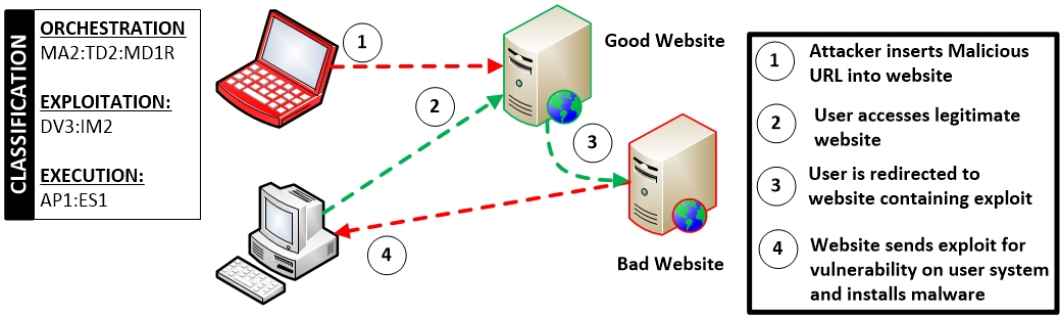

Fig. 2. Taxonomic classification of Drive-By Download

The usual implementation of WiFi Evil Twin phishing (Figure 3) requires two steps of deception (ES2). The attacker configures a rogue access point to advertise an SSID that is identical to a legitimate access point's in the target environment, such as a coffee shop or shopping centre etc., and also programmatically introduces a spoofed captive portal (IM2), which is automatically presented (MA2) to any user who connects to the Internet through it. The rogue access point both looks (identical SSID) and behaves in a manner that mimics the legitimate one (DV3). The rogue access point is distributed via a wireless hardware interface waiting for users to connect to the spoofed SSID. As software interaction is necessary to connect to the wireless network, this is classified as MD3. In the attack's usual implementation, targeting is promiscuous (TD2), directed to any user within range. The deception persists at least for as long as the user is connected (AP2).

Malvertisements in social media (Figure 4) are embedded by attackers after having previously posted a legitimate advertisement that gained in popularity, through user click volumes. Once that advertisement has gained enough reputation, the attacker replaces it with a malicious one, which automatically redirects users to an attack website (MA2). The attack targeting is usually promiscuous (TD2). Attack distribution occurs via the social media website and is forwarded automatically across user communities through in-built recommender and reputation processes on the social media site (MD1-R). The malvertisement deception vector uses cosmetic obfuscation through the social media system own carrier browsing system, which creates the appearance of its legitimacy as an advertisement by placing it in the advertisement pane, displaying

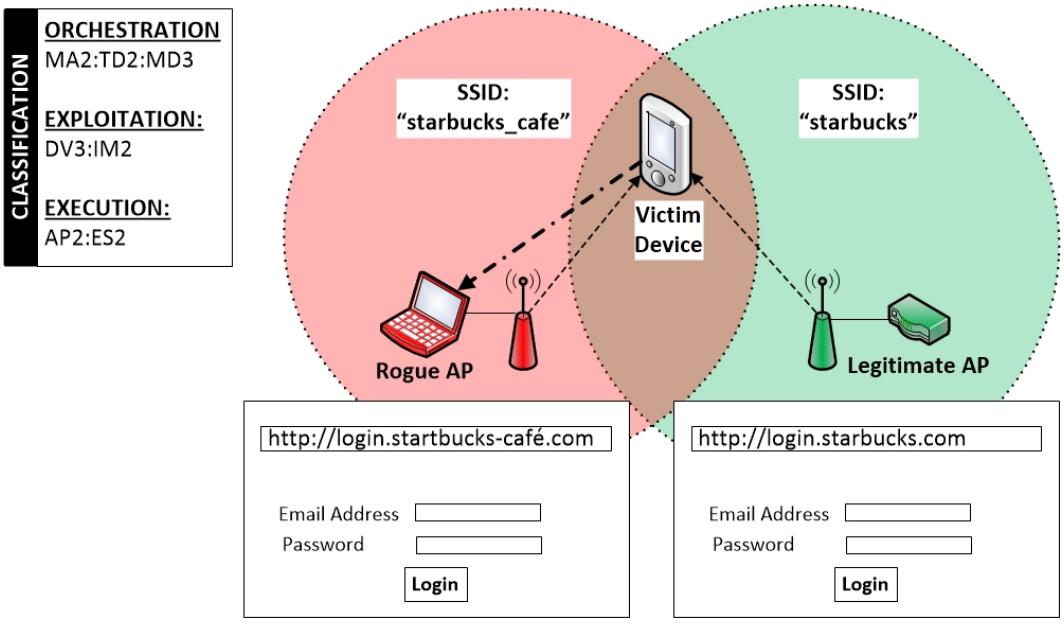

Fig. 3. Taxonomic classification of WiFi Evil Twin Phishing

Fig. 4. Taxonomic classification of SNS Malvertisement

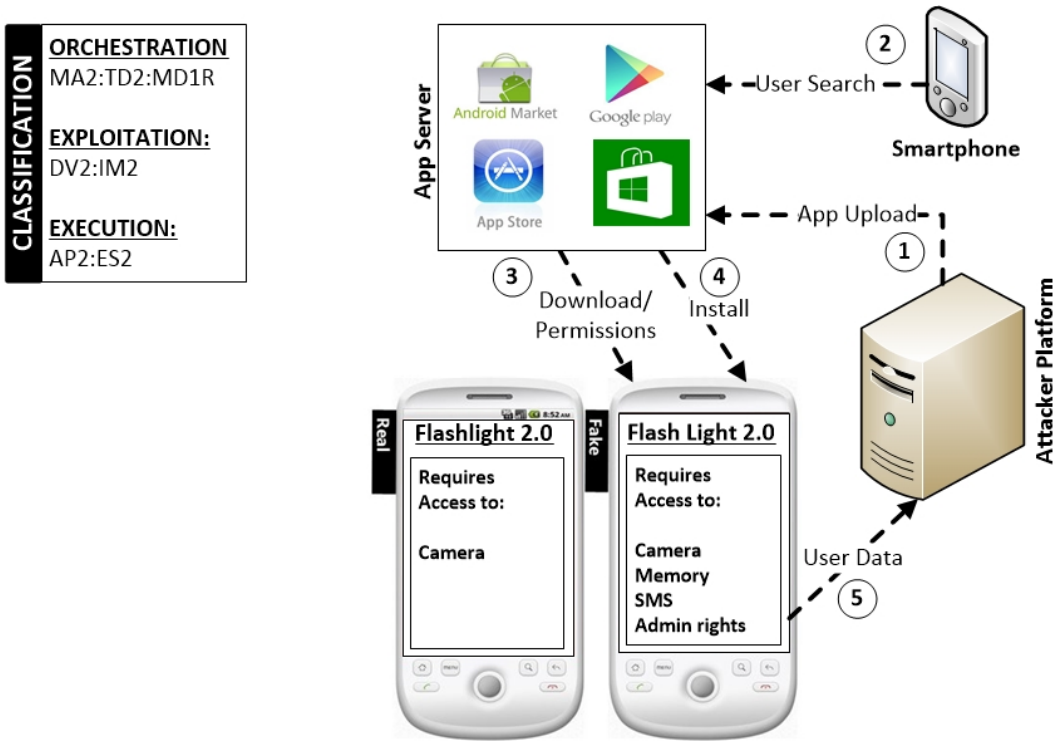

Fig. 5. Taxonomic classification of Fake Mobile App

a graphic and applying recommendation labeling to users (DV1, IM1). The attack persistence is continual (AP2), as after it deceives a user once (ES1), the malvertisement continues appearing in the advertisement sections of their social media profile.

In fake mobile applications (Figure 5), the attacker uploads a spoofed app that masquerades as a legitimate program in a mobile software marketplace (e.g. Google Play, Android, Windows Store, Apple AppStore etc.). The first step is to deceive a user who has searched for related keywords to download it. The second step is to deceive the user into accepting the request for granting excessive system privileges (ES2). Typically, attackers spoof popular types of applications with the purpose of gaining maximum exposure (TD2) before the app is identified as malware and removed. Depending on the implementation, deception can be based on looks, behaviour or both. In the example of Figure 5, a flashlight app downloaded from an app market features non-standard GUI and poor graphics, yet the user may still consider it legitimate if it operates the camera flash as expected (DV2), while in reality, it may be a Trojan horse stealing the user's private data in the background. The whole process is automated (MA2), and attack persistence is continual because the application resides permanently in operation on the mobile device until removed (AP2). The delivery of the deception requires programmatic manipulation (IM2) and distribution from the remote app server (MD1-R).

Having followed a similar analysis for a broad range of semantic attacks that have been discovered or are emerging, we have compiled Table II to summarise semantic attacks based on our taxonomy and observe common characteristics.

Typical Semantic Attacks in the wild	Orch.	Exp.	Exe.
Bluetooth Phishing (Snarfing Attack): This is targeted at any users that leave their mobile devices open to accepting Bluetooth transfer as a default setting. The attacker sends to the target device a malware file with a filename that looks like it corresponds to a popular application (e.g. twitter, facebook, a banking app etc.). When opened by the user, it starts exfiltrating data back to the attacker system [Abu-Nimeh and Nair ; Laurie and Laurie 2003].	MA1, TD2, MD3	DV1, IM1	AP1, ES1
Cryptovirus/Cryptotrojan/Cryptoworm: An attacker spreads a malicious application through email, infected programs (using a wrapper approach) and compromised websites. If the user is deceived into allowing the application to infect the user's machine (e.g., by opening the email attachment), it starts encrypting user data, such as files or the hard drive itself. At that point, it attempts a second deception, demanding payment from the user to release the private key that decrypts the data typically assuming the appearance of an official-looking letter from a government agency [Symantec 2014].	MA2, TD2, MD1-L	DV1, IM2	AP2, ES1
Drive-By Download: An attacker exploits vulnerabilities in a website's code structure (typically JavaScript) and proceeds to embed a malicious payload. The attacker then sends a link to this website, typically containing the website and full URL path to the exploited code, to potential targets (e.g. via phishing email, social media post etc). The machines of the users are infected when they open the link to the website [Mavromatis and Monrose 2008; Provos et al. 2009; Egele et al. 2008; Cova et al. 2010]. The deception takes advantage of the normal-looking cosmetics and behaviour of the affected website.	MA2, TD2, MD1-R	DV3, IM2	AP1, ES1
Fake Mobile App: The attacker uploads a spoofed app that masquerades as a legitimate program in a mobile software marketplace. Upon installation, the application requests full user or admin/root access to the device, such as access to volatile/non volatile memory, sending texts, making calls, or downloading other software. When the fake application runs, it typically exhibits no related functionality to the spoofed legitimate counterpart [Felt and Wagner 2011] (see Figure 5).	MA2, TD2, MD1-R	DV2, IM2	AP2, ES2
Forum phishing - manual: An attacker manually posts a message within a conversation thread on a web forum. This can consist of multiple messages posted in the context of the thread, with some being legitimate but others containing malicious links or images. Typically, the attacker must bypass CAPTCHA controls by manually entering a verification code designed to prevent web bots posting messages.	MA1, TD2, MD1-R	DV2, IM1	AP1, ES1
HTTPS Man-in-the-Middle Adware: An attacker intercepts a user's HTTPS requests and places a spoofed certificate within the root certificate store on the user's system. As a result, when a user send a HTTPS request the attacker redirects them to a phishing website that is incorrectly identified as legitimate through the spoofed certificate. A vulnerability facilitating this attack has recently been discovered to have been inadvertently introduced to Lenovo laptops through poorly written adware [Callegati et al. 2009; US-CERT 2015].	MA2, TD2, MD1-L	DV3, IM2	AP2, ES1
Instant Message phishing - automated: Similar to email phishing, instant messaging phishing includes obfuscated malicious URLs or attachments. The URL redirects the user to a malicious website and the attachment typically infects the instant messaging application or installs malware on the user's machine. In the case of the former, the attack configures the user's instant messaging account to resend the phishing messaging containing the attachment or URL to the user's contact list [Mannan and van Oorschot 2005; Leavitt 2005].	MA2, TD2, MD1-R	DV2, IM1	AP1, ES1
Malicious web pop-up: A malicious pop-up is generated through a browser or website application window that activates on user action, such as navigation, clicking or hovering over buttons, pictures and URLs. The window content typically takes on various approaches to deceive users, such as suggesting that they have won an online lottery and need to enter their bank details to collect the prize, or displaying a fake antivirus scan that identifies bogus infections on the user machine and advises download of a file to remove the infection [Abraham and UnduShobha 2010].	MA2, TD2, MD1-R	DV1, IM2	AP1, ES1
Malvertisement: Malvertisements are embedded by attackers after initially posting a legitimate advertisement that gains in popularity on a website. Once the legitimate advertisement has gained popularity and is trusted by the hosting website, the attacker replaces the advertisement with a malicious one. Once clicked, a user is redirected to a drive-by-download or bogus questionnaire. The malvertisement is distributed through the legitimate advertisement system on the website without needing to programmatically manipulate it [Li et al. 2012].	MA2, TD2, MD1-R	DV1, IM1	AP2, ES1

Multimedia Masquerading: In multimedia masquerading, an attacker embeds a malicious link in the form of a video or audio player, often displaying the video's first frame and title to strengthen the deception. Once clicked, the fake media redirects the user into installing a software application to play the purported media content or to a website requiring the user to input sensitive data, join a premium subscription service or complete a /questionnaire. Often the attack will propagate through social media platforms where the user, on completing a plugin installation to run the media content, gives the malicious adware permissions to re-post on the user's behalf. The fake media is then propagated to friends [Ford et al. 2009].	MA2, TD2, MD1-R	DV1, IM2	AP1, ES1
NFC Phishing: In NFC attacks, the exploitation is embedded in a modified NFC tag. If the user is deceived into scanning it, the user is presented with a request to download an application, which is malicious. The deception relies on the fact that NFC is a relatively new technology, which does not have a reputation of being used maliciously and the interface generally resides in a publicly accessible place governed by a trusted hosting authority, such as airports, shopping centres and schools [Madlmayr et al. 2008].	MA2, TD2, MD3	DV3, IM2	AP1, ES2
P2P Malware: A very simple attack, whereby the attacker shares malicious files over a P2P application, giving them commonly searched file names. A user downloads the malware matching their search criteria and infects their machine by opening the file. Obfuscation is achieved by manipulation of filename, extension and possibly the icon displayed by the proprietary browsing system of the P2P application. Replication occurs by the malware sharing itself through the user's P2P application [Johnson et al. 2008; Kalafut et al. 2006].	MA2, TD2, MD1-R	DV1, IM1	AP1, ES1
PDF File Masquerading: A Portable document format file is manipulated by using special functions in the PDF language that when compiled execute malicious functions, such as requesting access rights and login credentials or redirecting users to an attack website. The appearance and handling behaviour, as represented by the operating system, appear legitimate [Selvaraj and Gutierrez 2010; Bindra 2011].	MA2, TD2, MD1-L	DV3, IM2	AP1, ES1
Peripheral Masquerading - USB: In USB masquerading, an attacker configures a USB compatible device, such as a flash drive, mouse or keyboard, to contain a malicious payload that executes on insertion in a computer system. Typically, this is achieved using the autorun features in the operating system, but zero-day attacks have also exploited vulnerabilities in the way an operating system handles files on USB devices. Importantly, the malicious peripheral functions as intended, so as to avoid raising suspicions while infecting the target system in the background [Jacobs 2011]. In its usual implementation, this is a highly targeted attack (TD1), but a supply chain non-targeted attack (TD2) is also conceivable).	MA1, TD1, MD3	DV3, IM2	AP2, ES2
Peripheral Masquerading - Firewire: A hardware deception attack where the deception mechanism is presented as a physical peripheral utilising the firewire interface. This can be in the form of a memory stick or external media reader. Once the user physically connects the hardware device via the firewire interface to the target host, the malware in the device exploits firewire's direct memory access capability to steal user data or inject malicious data into the system [Boileau 2006].	MA1, TD1, MD2	DV3, IM2	AP2, ES1
Phishing Website: An attacker spoofs a legitimate website for the purposes of stealing user information or duping them into installing malware, usually by duplicating both appearance and behaviour.	MA2, TD2, MD1-R	DV3, IM2	AP1, ES1
Ransomware: Ransomware is malware, similar to cryptovirus (without encrypting files), typically delivered through a Trojan horse, that blocks access to a system by taking control of a browser of the operating system itself until the user pays a sum of money to unblock the machine. The ransomware attempts to convince victims that they are the subject of a criminal inquiry, from a law enforcement authority (using official looking logos and pictures), for ownership of illegal software or pornography. The user is then advised how to unblock the computer by following instructions for paying a fine electronically [Gazet 2010]. The attack is continual for as long as it takes the user to pay the money demanded by the software, after which the access restrictions are lifted.	MA2, TD2, MD1-L	DV1, IM2	AP2, ES2
Rogueware: Similar to a scareware application, rogueware is a standalone application, downloaded from a malicious website or as a wrapper/adware alongside a legitimate file. It masquerades as a well-known program, (such as a media player or Internet connection optimiser, which when executed attempts to steal the user's confidential data [Corrons 2010]. The rogueware continually deceives the user by running as a application on the user host system until removed, whilst	MA2, TD2, MD1-L	DV3, IM2	AP2, ES2
Rogue Access Point: An attacker installs a wireless network access point in public or private environment. Its aim is to harvest connecting users and perform man-in-the-middle attacks through network data sniffing [Hwang et al. 2008].	MA1, TD2, MD3	DV2, IM1	AP2, ES1

Scareware: A malicious computer program designed to convince a user that their system is infected with malware, pressuring the user into buying and downloading further (fake) antivirus applications (which are typically malware, spyware, adware etc.). Commonly, the protection software periodically displays wwarnings for infections and demands payment for licensing in order to remove them.	MA2, TD2, MD1-L	DV3, IM2	AP2, ES2
Search Engine Poisoning (Spamdexing): In search engine poisoning, a search engine is manipulated, so as to include a malicious website among the top-ranked results for given search terms. The users trust that since it appears high on the list of a trusted search engine, that website must be popular and very relevant to their keyword search, hence not suspicious. However, the attacker can force search engines to associate keywords with websites (that are malicious) in several ways, such as flooding websites that accept simple user generation content with specially crafted URLs [Sullivan 2008; Howard and Komili 2010].	MA2, TD2, MD1-R	DV2, IM1	AP1, ES1
SMS Worm (Selfmite): In the "Selfmite" SMS worm an attacker configures a text message to contain a web link to a recommended app that once downloaded by a user proceeds to send the same text, including the obfuscated link, to the first 20 contacts in the user's phonebook; thus spreading effectively by originating from a trusted source [Ducklin 2014a].	MA2, TD2, MD1-L	DV1, IM2	AP1, ES2
Spam Phishing Email (Botnet-generated): An email with a spoofed sender address and an enticing subject title and message requesting the user to download an attachment (which contains malware) or click on a malformed URL. Large-scale phishing attacks are typically distributed via botnets, which crawl the Internet for email addresses and are configured by botmasters to send template-based emails with varied malicious content. The attacker uses official-sounding language and logos to create a cosmetically appealing and authentic communication. Occasionally, spam phishing emails contain payloads for compromising the user's system and adding it to the botnet [Drake et al. 2004; Hong 2012].	MA2, TD2, MD1-R	DV1, IM1	AP1, ES1
Spear-phishing Email: Identical in format to a spam phishing email, however highly targeted toward a specific user, organisation or demographic, whereby the email subject, body of text and sender address are designed to take advantage of an existing relationship with the target. For example, the sender address may appear to originate from a friend and there may be an attachment appearing to be a photo they would be likely to receive from that friend. Spear-phishing is typically a manual process, as the attacker needs to be meticulous in the email construction to create a communication that is highly-specific to the intended target [Hong 2012].	MA1, TD1, MD1-R	DV1, IM1	AP1, ES1
Spear-phishing Email - APT: This is an advanced form of spear-phishing employed by advanced persistent threats (APTs). Highly targeted emails are first sent to harvest login credentials for Microsoft Outlook accounts using dummy User Access Control prompts. A compromised account of one employee, customer or partner is then used to send spear-phishing emails to other company insiders. The attacker periodically injects a malicious message containing previously exchanged Microsoft Office documents that embed hidden malicious macros into an ongoing email discussion among multiple people (to improve the exploit success factor). E-mails are sent from the accounts of people the target knows adding to on going discussions already taking place. The attacker will blind carbon copy (bcc) other recipients to further obfuscate the malicious e-mail [Arstechnica 2014].	MA1, TD1, MD1-R	DV3, IM1	AP1, ES2
Tabnabbing After a user unknowingly navigates to a malicious website, embedded malware on the latter detects when the page has lost its focus and attempts to deceive the user into thinking they had left a Gmail (or other popular website) tab open. For this, the attacker needs to display the Gmail favicon (short for "favourite icon") and a convincing title, such as Gmail: Email from Google, as well as a spoofed Gmail login page by adjusting Javascript code within the website. The spoofed gmail login page is used to steal the login credentials before redirecting the user to the real Gmail website. As it is most likely that the user had never really logged out of Gmail, the direction makes it appear as if the login were successful and, thus not suspicious [Raskin 2011].	MA2, TD2, MD1-R	DV1, IM2	AP1, ES1
Typosquatting (also known as Cybersquatting): User mistypes a domain name (e.g, "twiter.com" instead of "twitter.com") into a browser and lands on a phishing website, which has copied the appearance of the legitimate, intended website. Typically the website appears identical, but does not have the same functionality. An example would be a social media website with a login prompt. Once the user enters their login details into the phishing website, they are redirected to the legitimate website after their personal data have been stolen [Szurdi et al. 2014; Agten et al. 2015].	MA2, TD2, MD1-R	DV1, IM1	AP1, ES2
Visual SSL Spoofing: In SSL spoofing, the attacker configures a website to imitate specific parts of the browser interface, which a user would expect to display SSL/TLS connectivity status. By using official-looking logos or address bars with the typical padlock icon demonstrating encrypted communication, users can be fooled into believing they are on a secure website and entering sensitive information which the user will steal [Adelsbach et al. 2005].	MA2, TD2, MD1-R	DV1, IM1	AP1, ES1

Watering hole: A highly targeted version of the drive-by-download attack, whereby an attacker embeds malware on websites that are visited by a specific individual or group of people. Eventually, the user/group will be exposed to the exploitation in one of the websites they regularly visit and trust. Here, the attacker is persistent in the attempt to exploit the chosen individual/group by infecting multiple websites they are known to visit [RSA 2012].	MA2, TD1, MD1-R	DV3, IM2	AP1, ES1
WiFi Evil Twin: Similar to a rogue access point, on top of the rogue AP the attacker spoofs a nearby SSID of another WiFi service whilst also programmatically duplicating any web-captive portal (typically an authentication page) that exists on the legitimate access point in the environment. Once a user connects to the spoofed access point, the attacker intercepts all traffic, syphoning off credentials, files and sensitive information [Song et al. 2010; Felt and Wagner 2011; Phifer 2000; Briones et al. 2013]. An advanced version of this attack uses the snoopy platform that can be embedded within a UAV drone [SensePost 2014].	MA1, TD2, MD3	DV3, IM2	AP2, ES2

Table II: Taxonomic Classification of Semantic Attacks

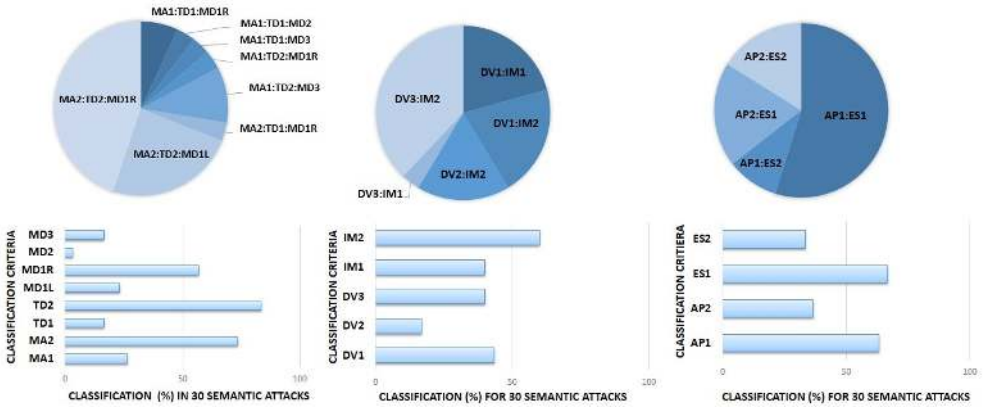

Fig. 6. Taxonomic statistics of Table II

The taxonomic classification of Table II can help identify the characteristics that are shared between different attacks (Figure 6). For example, it is evident that in terms of orchestration most attacks exhibit promiscuous targeting (TD2), software-based remote distribution (MD1-R) and automation (MA2). A defence approach that would thwart this orchestration pattern would be applicable to almost half of the semantic attack types listed here. In terms of exploitation, the vast majority rely on cosmetic deception alone (DV1) or in combination with behavioural deception (DV3). Approaches that would detect or prevent mismatches between the appearance and function of GUI components on websites and applications would reduce considerably the semantic attack surface.

The classification also shows the greatest strength of semantic attacks. Most require no more that one deception step (ES1), which means that a single error in judgement on behalf of the targeted user is sufficient. This is exemplified by spear-phishing attacks used to gain a foothold in target systems with otherwise strong technical security measures, from financial institutions and nuclear facilities [Loukas 2015], to the computers of high-value individuals. Even a prominent cryptographer with several computer security patents in his name can be deceived into clicking the “accept” button on a spoofed LinkedIn invitation from a non-existent employee of the European Patent Office [Eeckhaut and Vanhecke 2014].

3. SURVEY OF SEMANTIC ATTACK DEFENCE MECHANISMS

The impulsive and unpredictable behaviour of human users has been widely characterised as one of the most significant weaknesses in modern computer systems [Hasan and Prajapati 2009]. While security controls against technical exploitations have improved considerably in recent years, the same successes have proven difficult to replicate in user security.

Training and experience can certainly help but are neither easy to acquire nor sufficient [Mitnick and Simon 2001; Hinson 2008]. The landscape of semantic attacks is wide and continuously expanding. While user targeting has been historically limited to emails and websites, the advent of the Internet of Things [Hasan and Prajapati 2009] is expanding the scope of exploitations to even include smart home appliances [Kuipers et al. 2010]. Current systems have limited capability to compare a spoofed file icon with the associated extension or to efficiently highlight the execution similarities of different file extensions to users in a pro-active fashion. Similarly, users rarely inspect their computer environment before performing actions that may compromise their system [Jordan and Heather 2005; Dhamija et al. 2006; Abraham and UnduShobha 2010]. The following survey describes an overview of mechanisms that can protect users against semantic attacks.

3.1. Organisational

3.1.1. Policy and Process Control. Policy and process provide hierarchical control through management frameworks as opposed to technical systems. Designed at lowering exposure to semantic attacks, well maintained policy and organisational procedures help to mitigate and significantly lower the risk of a potential exploit occurring, without relying on the technical capabilities of users [Mitnick and Simon 2001; Hinson 2008; Colwill 2009].

Policy and process control is defined around the business and user environment, as appropriate to their specific needs, for organisational and personal use. However, security frameworks introduced to address semantic attacks have been added as bolt-ons to the wider technical threat, rather than embedded into the governance and strategic development of policy and process control. Mitnick and Simon [2001] reiterates this approach as a major vulnerability, insisting policy should be intrinsically built with people management and therefore embedded in the core of any information security framework. The prevalence of semantic attacks has resulted in a higher degree of applicability within policies that are integrated in internationally standardised frameworks ([Calder and Watkins 2010; Calder and Watkins ; Ali 2014]). Dedicated Governmental Guidelines have also been proposed in ([GOVUK 2015; CPNI 2013; Calder and Watkins 2014; Jansen and Grance 2011]) as well as detailed methodologies and investigations in guide books and research ([Peltier 2013; Frauenstein and Solms 2013]). The book published by Peltier [2013] offers a comprehensive collation of strategies for implementing secure policy and process for IT governance; combining standards and best practice guidelines for integrating information security across organisational domains i.e. user roles and responsibilities to embedded data security tools. Frauenstein and Solms [2013] have developed a user centric, holistic framework specifically targeting phishing attacks. The framework includes a user, "Human Firewall", layer in the information security framework, enabling cross communication between defence dimensions (the user, organisational policy and technical controls) by combining mechanisms such as user awareness and training with policy and linked technology tools.

Current frameworks focus on preemption through layered control mechanisms, as opposed to being proactive and dynamic to change. Policy and Procedures need to be flexible to unknown and unforeseen attacks and therefore appropriate to the chang-

ing threat landscape. Fixed guidelines can quickly become out-of-date as new attack methods are constantly being developed. Furthermore, as a personal tool, policy and procedure can be ineffectual. Rules and regulations designed to enforce and maintain security are too large in scope, almost entirely irrelevant to the personal user and their specific use of computer systems and the internet.

3.1.2. Awareness Training. As user deception is the primary attack vector, susceptibility to semantic attacks can be somewhat reduced with user awareness training. Education is a core component of the defence-in-depth model and in the instance of semantic attacks relates to areas where technical security mechanisms have proven to be inadequate. Interactive training maximises learning and awareness, such as bite-size quizzes, tests or games [Kumaraguru 2009; Cone et al. 2007; Arachchilage et al. 2012; Sheng et al. 2007], which when applied periodically provide education on the characteristics of different attack types. A large proportion of research into awareness training has focused on the approach of content, the methodology of its delivery and how data gathered from testing and formal application can be used to shape security policy and user training programmes [Kruger and Wayne 2006; Kirlappos and Sasse 2012]. Miller and Garfinkel [2006], Schechter et al. [2007] and more recently Lin et al. [2011] have conducted research to measure the effectiveness of awareness systems built into browser security systems, including security toolbars, domain highlighting for suspected phishing and HTTPS/SSL/TLS connection indicators. Their results showed that none of these methods were effective because users would not understand or pay attention to warnings. Similarly, the more recent work of Lee et al. [2015] has shown that security images meant to provide visual confirmation of website authenticity in Internet banking have also proven unreliable, as 74% of the study's participants entered their password after the security images were removed.

An analysis of user awareness indicators in a role-playing survey carried out in [Sheng et al. 2010] demonstrated that the attitude of users against phishing attacks correlated highly with certain demographic characteristics. For example, the age group 18 to 25 was shown to be more susceptible to phishing than other age groups. Furthermore, once educational material was provided against a demographic, users were 40% less likely to enter information into an attack website. Interestingly, this was shown to be counter-productive in a few instances where users avoided legitimate websites too.

A study carried out in [Doupe et al. 2011] suggested using the familiar concept of live "capture the flag" competitions also for measuring and teaching "cyber situational awareness" to users faced with security threats, including semantic attacks. Their competition involved 900 students spread across 16 countries. While undoubtedly interesting and useful, there was no mechanism for measuring which of the students' skills and to what extent were improved by participating in the competition.

Other research has approached awareness training by exploiting the interactivity observed in gaming and its educational impact and value, with development and integration of technology trends such as Social Networking. In [W. A. Labuschagne and Eloff 2011] and [Gulenko 2013] Social Networking Sites are used as a platform to both delivery awareness applications to maximise exposure, exploitable functionality such as automated propagation through friend activity notifications and viral trends. Furthermore SNS were able to provide the means to measure effectiveness of applications in a social media environment by monitoring the response of subsequent user activity such as public/private access to profiles. Other developments have taken awareness efforts further through web-based/portal training systems, even by introducing proxy based training such as "awareness middleware" for access to the internet [Kritzinger and von Solms 2010]. The middleware in this sense is a proxy training portal, responsible for brokering a connection between the user and internet access. Only after

users successfully complete a training quiz/test for improving security education and risk awareness will access to remote resources is permitted. Stewart et al. [2009] and more recently Konak and Bartolacci [2012] have developed training platforms to create awareness and better understanding of the dynamics of semantic threats, by utilising virtualisation as a collaborative and interactive training platform.

A study conducted by Anderson et al. [2013] has used neurological scanning to determine how "habitation" (an automated response based on a reoccurring theme) occurs when users are presented with common, static security warnings, such as HTTPS certificate warnings. The research analyses brain behaviour through functional magnetic resonance imaging, where users were shown to exhibit a drop in visual processing sections of the brain after multiple exposures to a similar security warning. As a solution, the researchers suggested security warnings that are polymorphic, constantly changing appearance and behaviour each time they are generated (e.g., different colour, text, location, size, etc.). This was shown to reduce automatic responses and influence higher cognitive responses.

A recent study by Neupane et al. [2014] identified the neurological indicators associated with user response when exposed to semantic attacks. They showed that users exhibit notable brain activity associated to decision-making, problem solving, attention and comprehension, when security scenarios arise. The study was able to show how individual personality traits can influence decision making results, which can be used to improve training and awareness programmes. For example, training users to respond assertively to malware warnings may prove ineffectual outside a controlled environment for users that exhibit highly impulsive behaviour.

Novel suggestions for creation of a user risk-detection tools [Schaff et al. 2013; Guillaume et al. 2014], can be used to build and define user awareness and develop training policies. The experiment conducted in [Bakhshi et al. 2009] has demonstrated how participating organisations can identify the level of user security awareness, through targeted phishing attacks. Whilst this type of experiments can yield very useful results, they can also be disruptive to an organisation's daily work and may have to be abandoned early (as was the case in the particular study). Furthermore, results need to be followed up through continually improved awareness programs and efficient user risk measurement. These tasks are manually intensive and their long term benefit may not be obvious to organisations. Awareness training approaches certainly have an impact, but the extent of this impact is uncertain, as there is no reliable approach for measuring the lasting effect. Primary evaluation is traditionally based on supervised testing in specific case studies [Eminagaoglu et al. 2009; McCrohan et al. 2010]. The extent to which different awareness training strategies employed by organisations impacts their quarterly or yearly breach statistics is not measured.

3.2. Technical

3.2.1. Sandboxing Mechanisms. Sandboxing is the process of creating an isolated computer environment, typically through virtualisation, to test untrusted operations such as those observed in unverified and untested code or programs. It has been effectively implemented as a security solution in various fields of computing, from specific code platforms [Leroy 2001] to browser systems [Barth et al. 2008], and in the field of smartphone security to improve defence against malicious software [Blasing et al. 2010]. Proprietary applications, such as Adobe Acrobat X, have also implemented their own sandbox engine for enhanced security [Xiao and Zhao 2013].

A sandboxing environment can check and analyse characteristics associated with an attack by categorising behaviour for anomaly detection [Greamo and A.Ghosh 2011]. Further sandboxing methodologies can provide an assured integrity checking mechanism [Peterson et al. 2002; Brickell et al. 2011]; this approach relies heavily on the

accuracy of behavioural analysis algorithms for detection and introduces the issue of increased resource demand. In the context of semantic attacks, sandboxing offers limited control, especially as it handles the execution of code, rather than. A technical attack using visual deception methods instead of technical mechanics to deceive users would need to be understood by the sandbox, but this is a very complex task requiring sampling of graphical interpretation. Web browsers [Chromium 2015; Firefox 2015] have implemented sandbox technologies to prevent websites from manipulating the browsers' own visual and behavioural properties, such as the URL address bar, browser tabs and security indicators. However, these mechanisms are still unable to prevent the execution of spoofing attacks, such as those creating a fake address bar [Drake et al. 2004; Comodo 2015] using code that appears legitimate to the sandbox, but is visually deceiving the user. Instead, the limited research conducted in the application of sandboxing for semantic attacks focuses on its value as a learning tool. Romney et al. [2005] utilised a sandbox environment to gather data used in enhancing user security training.

The Windows and Linux operating systems implement a number of sandbox mechanisms to prevent unauthorised changes being made to a system; acting as a good line of defence even when a user may have unknowingly been deceived by a semantic attack. One such example is the secure-attention-sequence or secure-attention-key as it is better known, which is a secure method of preventing applications spoofing an operating system login process. The login system is sandboxed from potential threats by using the operating system kernel to suspend all running processes before it is initiated. The user access control (UAC) system is another Windows sandbox mechanism that uses the user interface privilege isolation (UIPI) to prevent applications from unauthorised privilege escalation during execution. This is achieved by isolating processes marked with a lower integrity level (e.g. unsigned, untrusted third party application) from sending messages to higher integrity level processes (e.g. to request an action that might require root system access), until granted by an authorised user in the UAC prompt or valid certificate from an approved code-signing authority [Microsoft 2007]. This approach can certainly limit the impact of some semantic attacks, but is specific to the operating system and cannot observe requests made to a remote system that it has no control over.

A system titled "BLADE" (block all drive-by exploits) developed by Lu et al. [2010], provides a sandboxing mechanism to mitigate drive-by download exploitations. BLADE introduces a browser-independent operating system kernel extension, which enforces the rule that executable files must originate from an explicit user action providing consent. In the case of a drive-by download, where there is normally no explicit user action involved, the system redirects to a secure sandboxed location on the user system disk where execution is prevented. This approach is agnostic of the mechanism for deception or obfuscation and whether it is a zero days threat, as it responds only when a download occurs without specific user consent. In testing, BLADE was able to block all drive-by download attacks against a sample of over 1900 active malicious URLs with a minimal effect on system performance.

Invincea [2014] and Pro [2014] offer commercial sandboxing environments that integrate into existing operating systems. These applications install an abstraction layer for handling the execution of programs and files (user initiated or otherwise). Whilst not exclusively aimed at providing protection against semantic attacks, they offer a last line of defence where a malicious application may have found its way on a user system. Requests made by all executing programs are analysed in real-time and categorised as either a threat or legitimate, and are then handled appropriately.

The open source project "Qubes OS" [OS 2015] has developed an operating system that utilises sandboxing to logically separate different applications on a system

from infecting another, should one particular element become compromised. This is achieved by separating different applications into virtual machines that are isolated by security domain, e.g. work/personal. The system also integrates graphical properties, such red app window frames for untrusted and green for trusted, to visually identify applications in different sandboxes. Whilst this functionality does not necessarily block a semantic attack from occurring in the sandbox itself, it can help inform users whether they should trust the behaviour or appearance of application based on the specific sandbox's security domain.

Recently, Bianchi et al. [2015] have developed a tool for the android operating system designed to prevent malicious apps executing cosmetic (DV1), behavioural (DV2) or both cosmetic and behavioural deception (DV3). This can include spoofing an applications appearance or generating misleading visual cues on top of legitimate applications by capturing and analysing application programme interface (API) calls to the android graphical user interface. The researchers employ a method call "static code analysis which interrogates application bytecode, based on a system proposed by Egele et al. [2013], during run-time to scan for requests made to specific Android API's which can enable the execution of visually deceptive components into an android systems graphical interface. The method can be applied for both preemptive and proactive protection of the android interface by vetting applications before release or by detecting this behaviour on the host device during operation. Whilst the approach was shown to be successful at identifying applications that use these APIs to execute cosmetic deception, the researchers also found that there was a significant rate of false positives. In response, they supplement the sandbox mechanism with a visual reputation system to confirm the identity of applications which use these APIs legitimately and those that do not. Crucially, in an evaluation involving 308 users, this approach was shown to significantly improve a users ability to detect malicious applications.

3.2.2. Authorisation, Authentication and Accounting (AAA). Authentication, authorization, and accounting (AAA) is a framework for intelligently controlling access to computer resources, enforcing policies, auditing usage, and providing the information necessary to bill for services. It is usually implemented in controlled environments, especially where there is a varied user landscape posing a risk to the control and protection of data. The framework provides controls for access to resources (Authentication), organisational policy enforcement (Authorisation) and auditing of resource usage i.e. login times, session length, devices accessed (Accounting). Its use aims to ensure that organisations have a detailed level of assurance and control over who has access to a system, based on data on names, roles, skill sets etc. In large deployments, such as organisation domains and networks, AAA can be effectively managed through supervision, ensuring extended identification and delegated access is in place to run executables, access shared folders etc. However, in a home user environment, AAA is less practical because the central supervising authority remains with the user [Motiee et al. 2010]. The capabilities of the individual require correct usage of "least user rights" and administrator privileges where appropriate to protect the system and secure data.

Many technical, autonomous, session based protocols such as Kerberos [Neuman and Ts'o 1994] have been successfully combined with secure centralised authentication and authorisation platforms [Desmond et al. 2008]. Such solutions have proven effective at managing AAA assurances, but governing configuration rules are statically implemented and lack interactivity and dynamic visual interfacing with users once access to a system has been established. On the other hand, security platforms such as Microsoft's User Access Control not only indicate to users that an action may require more privileged rights but it actually displays when a program is trying to gain these rights to run and it asks the users to identify themselves beforehand [Motiee et al.

2010]. One can then make an informed decision. However, the system still relies on the user's due diligence in correctly selecting use access control options. Advancement in identity management, such as Microsoft's Forefront Identity Manager [Turner et al. 2010], have developed comprehensive AAA security that has a wide range of application effect at preventing semantic attacks. Yet, although they provide flexible control over users and access to resources, Identity management systems are not dynamic in their mitigation mechanisms. With regards to the growing threat of SNS-based semantic attacks, Boshmaf et al. [2012] have identified a range of AAA flavoured mechanisms that can provide some protection against automated sybil attacks [Huber et al. 2010] and other current threats. Portable identities, with mutual verification and authentication between open systems (e.g. Facebook, Twitter, LinkedIn), limit the ability to abuse SNS functionality. This is suggested as the main challenge to address as it maintains usability without introducing difficult to use, user facing security controls. This area still lacks working examples and complex semantic attack SNS exploitations remain a major problem. For AAA frameworks to be able to address semantic attacks effectively, there needs to be a shift towards more pro-active measures, rather than relying almost entirely on sound management and individual user security awareness against system generated warnings.

3.2.3. Monitoring. In this context, by monitoring we refer specifically to the observation of computer system behaviour, generated by user/programmable actions, via collection, aggregation and analysis mechanisms. Monitoring is a key security mechanism for semantic attacks as new exploitations can be classified through exposure and effective security controls introduced through understanding of compromising user actions. Historic monitoring has required heavy supervision and forensic investigation, whereas new developments allow semantic attacks to be identified and addressed through use of Honeypots [Lee et al. 2010]. These virtualised environments allow security practitioners to analyse exploitations, classify attack characteristics and identify the user behaviour that can lead to compromise. Research conducted in statistical monitoring focuses on developing semantic attack masquerade aversion by modelling user search behaviour. For example, Salem and Stolfo [2011] have introduced a novel method of monitoring and machine learning in order to identify anomalous user behaviour on a given system. The work conducted by Ruskov et al. [2014] proposes adding dynamic adaptation to crime scripts (a sequence of actions observed in a semantic attack), which can model user/attacker behaviour in a given scenario and can be used in simulation, monitoring data output for collation in order to simplify the process of enumerating attack procedures. A promising piece of recent research in this area by Stringhini et al. [2013] has developed malicious website detection by monitoring the redirection path taken to reach a web destination, instead of analysing physical website features.

Lu et al. [2011] have introduced "SURF", a browser extension designed to detect search engine poisoning by monitoring "search-then-visits" user sessions and classifying redirections into legitimate or malicious. The particular system was able to achieve 99% accuracy in the researchers' empirical real-world evaluation.

Li et al. [2013] have topologically mapped dedicated hosts on malicious web infrastructures, involved in orchestrating redirection paths to malicious websites. The research used a PageRank algorithm to monitor and capture attack hosts responsible for traffic brokering and exchange for malicious activities. This study focuses on the management overlay that cybercriminals implement to conduct and control large-scale malware distribution. As part of the study to identify redirection chains, the researchers used a system called "WarningBird" [Lee and Kim 2012], which is a URL detection system for Twitter, designed to protect against conditional redirection, where the attacker redirects web crawlers to a legitimate landing site, but a user to the at-

tack location. In combination with a real-time classifier, WarningBird exhibits a high degree of accuracy against large samples of Tweets.

3.2.4. Integrity checking. User and application/data integrity is difficult to assure without proof or analysis, especially if the data and subsequent behaviour characteristics originate from an external network, colleague or friend. Integrity checking provides the user with a visual response and technical assurance as to whether the file, website, or data should be trusted [FirstCyberSecurity 2009; Webroot 2013]. Systems such as TripWire implement File Integrity Monitoring that continuously monitors the integrity of a data e.g. a file a based on its properties and behaviour, and further work has shown how this defense mechanism can become dynamic [Abdullah et al. 2011]. A study carried out by Dhanalakshmi and Chellappan [2010] evaluates methods for file type classification, for the purposes of identifying and detection file masquerading attacks. The research evaluates file type detection algorithms such as FHT (File header and trailer), MDA (Multi-Discriminant analysis) and CFD (Compound file detection) in order to identify the true file type. Using this approach the researchers are able to classify fragments of a file, or embedded files to determine a file type; thus providing a key mitigation mechanism for common file masquerading attacks. Research conducted in [Hara et al. 2009] shows how using simple methods to determine integrity, by checking and comparing visual similarity-bsaed phishing can provide accurate and effective results. The researchers demonstrate that by analysing the similarity between visual components in a website, without prior knowledge that it is an attack platform, they are able to automatically determine specific templating of websites that have been spoofed by a large degree of phishes sites.

Recent studies have proposed solutions for detecting the integrity of URLs [Bhardwaj et al. 2014] through optimisations algorithms and image detection techniques to mitigate rogueware [Seifert et al. ; Dietrich 2013]. Integrity tools have also been developed to provide protection against WiFi "Evil Twin" access points, by analysing key components related to the appearance SSID-mac address mappings as well as hashing and cryptographic algorithms being used [Darknet 2015]. Future research addressing semantic attacks can consider digital signatures or certificates that originate from a central authentication system to prove legitimacy of data presented in a file or application (website, email, utility programs), where the data is uploaded, analysed and approved, so that where data are missing from this central authority or using unknown certificates, there is reduced functionality until it is properly verified for secure operation in its respective environment (domain, shared platform - desktop, folder, cloud storage). In social media platforms have been plagued "Sybil" attacks (where a reputation system is subverted by creating multiple fake identifies in a P2P network e.g. Social Network Site), Alvisi et al. [2013] document and evaluate the evolution of defences that have been developed to mitigate this growing threat. The study shows trends in defence mechanisms, that are quickly becoming mathematically complex in nature and sometimes proven to be ineffectual in practice e.g. when applied against a real-world primitive attack. The researchers suggest a defense in depth approach is necessary, such as implementing region-based community detection and social graph topologies to identify sybil attacks where sybil regions (forged identities/communities) are connected to honest nodes (legitimate user profiles). They also indicate that social network operators should implement machine learning algorithms, user profiling tools and user activity monitoring in combination to help identify attacks.

3.2.5. Machine Learning. Research has demonstrated how malware detection through machine learning can be dynamic, where suitable algorithms such as k-nearest neighbours, decision tree learning, support vector machines, bayesian and neural networks, can be applied to profile files against known and potential exploitations and distin-

guish between legitimate and illegitimate data [Chandrasekaran and Upadhyaya 2006; Singhal and Raul 2012; Gavrilut et al. 2009]. Machine learning algorithms have been successfully applied to detect malicious emails, using anomaly classification techniques that demonstrate the potential and capability for further application in other related semantic attack areas [Dong-Her et al. 2011]. Research conducted to tackle the threats of social engineering specifically has seen the employment of neural networks to predict the legitimacy/illegitimacy of phone calls. Using a fabricated dataset, the approach showed positive accuracy in prediction responses, and as such the classification methodology may be suitable across a wider range of semantic attack types. However, dependable datasets generated from real-world data may be essential, as well as the ability to formally integrate algorithm decision processes into suitable access-control systems [Sandouka et al. 2009]. An effort to build credible classifiers for teaching machine learning algorithms has been proposed in [Lee et al. 2010], where bespoke honeypots provide features that can indicate and help filter out social network spammers.

The application of machine learning for defence against semantic attacks can be traced back to the training of support vector machines using binary features for detecting early spam emails [Drucker et al. 1999]. A significant amount of research utilising machine learning algorithms has focused on building efficient detection mechanisms for web and email phishing attacks, with work by Fette et al. [2007] and Basnet et al. [2008] testing a range of clustering and filtering techniques to classify data for accurate prediction of attack detection. Similarly, the research conducted by Garera et al. [2007] highlights the difference between legitimate and illegitimate URLS and proposes a regression filter for constructing classifiers specifically for URL phishing. In a related piece of work, [Bergholz et al. 2008] developed a system to detect phishing emails based on the component features of the email, such as body of text, sender address or embedded images, using combinations of machine learning for classification and class modelling mechanisms for filtering. Bergholz et al. [2010] have continued this research by demonstrating the effectiveness and functionality of a range of filtering approaches that, if combined, can provide effective detection for this attack vector.

Advancements in the application of Ontological Semantics Technology have seen developments that demonstrate applicability semantic attacks too, employing an artificial intelligence approach with an expression based dataset as the prediction mechanisms for building inferences about textual data [Raskin et al. 2010]. This type of detection mechanism indicates useful application against phishing attacks observed in emails and websites. Whilst this technology is not strictly classified as machine learning in regards to the algorithms that are used, there is a learning process coordinating decisions on a piece of data, in this case text-based language, to make predictive inferences about malicious or non-malicious communication. More recent research by Xiang et al. [2011] has introduced CANTINA+, which is a framework for website phishing detection using machine learning algorithms over a range of systems including search engines and third party applications. This research brings together multiple systems, such as search engines, third party services with machine learning mechanisms to accurately classify web pages.

In [Cova et al. 2010], machine learning and emulation techniques have been used to detect anamolous javascript behaviour against established behaviour profiles to prevent drive by downloads. The researches apply 10 different features of an attack process to categorise activities involved in drive-by downloads, emulating website behaviours that match these features in combination with an anamoly detection classifier; implemented in a publicly available tool named JSAND. The researchers presents results of JSAND's application on a large-scale dataset of Javascript code. The study is able to demonstrate through the use of an emulator, detection of malicious code when compared with anamolous features built into a learned classifier (machine learning)

model. Further research aims to build on the JSAND system to provide a browser extension that provides pro-active detecton and mitigation of drive-by download attacks.

Machine learning techniques have been applied to detect phishing attacks in Twitter. In [Aggarwal et al. 2012], a URL phishing detector for Twitter focuses on tweet content, length, hashtags, mentions, age of account, number of tweets combined with URL features (shortened text, non standard domain prefix and suffix structure). Using these variables within machine learning classification techniques (Naive Bayes, Decision Tree and Random Forest algorithms), detecting phishing techniques with an accuracy of 92.52 %. The system has been developed for practical implementation, where the researchers have deployed the system as a Chrome browser extension, working in real-time to classify whether a tweet is a phishing attack or a legitimate communication from a twitter account. Furthermore, the study has been tested on a user community to evaluate its usability and practicality in application; with the researchers claiming its relative simplicity and ease of use. The work represents research with a product outcome that is usable to the internet community against a known vulnerability of a popular social media platform. Other URL detection mechanisms have been demonstrated by Thomas et al. [2011], the study introduces a system that provides real-time URL spam filtering. The system proposed, Monarch, performs real-time filtering of scam, phishing and malware URLs that are submitted to web services through the use of linear classification with the combination of iterative parameter mixing and subgradient L1-regularisation; calculating an overall accuracy of 91% and 0.87% false positive margin.

Stringhini and Thonnard [2015] have recently developed a system for classifying and blocking spear phishing emails from compromised email accounts by training a support vector machine based system with a behavioural model based on a user's email habits. The system collects and profiles behavioural features associated to writing (e.g., character/word frequency, punctuation, stylus), composition and sending (e.g., time/date, URL characterises, email chain content) and interaction (e.g., email contacts). These features are used to create an email behaviour profile linked to the user, which is continually updated whenever a user sends a new email. Testing showed that the bigger the email history of a user the lower the rate of false positives, dropping below 0.05% for 7000 emails and over.

Machine learning algorithms provide an architecture for profiling semantic attacks in a pre-emptive fashion. By utilising characterisation variables with behavioural input datasets (usually gathered via monitoring), precise predictions and indicators can be established as to the significance of a file's or a user's behavioural effect on a system. Whilst machine learning tools have been built, extensively tested and evaluated in research, application has heavily focused on tackling phishing attacks, and most specifically the cosmetic aspects (DV1), covering only a fraction of the wider semantic attack problem space. There are not many actual implementations within deployed and popular email services in the commercial space, and consequently there is a lack of empirical evaluations in real user-facing environments.

4. ATTACK VS. DEFENCE MATRIX

The Attack Vs. Defence matrix shown in table III provides a comparative platform for mapping current defence mechanisms to the taxonomic criteria of semantic attacks. Its benefits are two-fold. It provides researchers with an indication of where the current state of research stands (the majority view) and how to effectively categorise this research (the taxonomy classification criteria). It also highlights attack vectors that have seen underinvestment and may point towards areas where new research may yield useful results, potentially achieving wider coverage by combining approaches from multiple areas into a single, autonomous solution. For clarity we match each of

the survey literature to classification criteria where it has had the most impact or for which is directly or most relevant, and not criteria with an association by second or third order of consequence.

We can observe that target description is the classification category that is one of the least covered across the surveyed defence mechanisms. The usual approaches relate to policy and process, which are typical in the corporate arena, but somewhat surprisingly, we have identified only three related technical approaches, none of which in monitoring or integrity. This highlights an area where further research could be beneficial. For example, semantic attacks that are automated (MA2), remotely distributed (MD1-R) and promiscuously targeted (TD2) would generate the same pattern and possibly noticeable volumes of network traffic, which may be detectable by a network monitoring system. Alternatively, an attack that is manual (MA1) and targeted (TD1) may be more easily identified by an integrity mechanism, for example by authenticating an addressable feature.

Another observation that can be made is that there are several areas that are not adequately addressed by policy and process controls. Examples include the hardware distribution techniques that feature no software interaction (MD2), as well as attacks that persist and may target the user periodically (AP2). From the matrix, it is also evident that awareness and machine learning have the widest and largely overlapping coverage. Yet, we have not identified a single example of research where the two areas have been combined, potentially by incorporating a machine learning component in an awareness system, or by feeding awareness data for the training of a machine learning system.

Interestingly, awareness efforts have primarily focused on exploitation factors, and particularly in understanding the impact of education against the plethora of deception vectors that are used in attacks. However, most have only addressed this through scrutiny of user interfaces (IM1) and not of programmatic interface manipulation (IM2), perhaps because awareness methodologies are typically designed for non-technical audiences. Also interesting is the application of sandbox technologies. They are highly appropriate for the software-based locally distributed (MD1-L), one-off (ES1) attacks, which are generally difficult to defend against, but have seen a relative underinvestment in application against deception techniques (DV1-3). However, as recent research has shown [Bianchi et al. 2015], sandboxing can be effective in identifying semantic attacks that implement both behavioural and cosmetic deception. Therefore, sandboxing mechanisms are likely to offer a viable focus of future research against deception techniques.

A distinctive observation of the matrix relates to defences that incorporate awareness, sandbox and machine learning techniques. In combination, these three defence categories apply to almost all of the classification criteria. Note that we previously demonstrated how research in each classification category will be to a large extent applicable across several attacks that share a characteristic. Therefore, it is conceivable that a defence system combining these features would potentially provide protection against a very wide range of semantic attacks. To the best of our knowledge, a defence mechanism designed to mitigate a multitude of semantic attacks integrating all of these features does not exist yet.

5. DISCUSSION: GRAND CHALLENGES IN SEMANTIC ATTACK RESEARCH

Semantic attack research is fundamentally multidisciplinary. Different aspects of it may require an understanding of personality traits, human-computer interaction, computer and network monitoring, malware dynamics, and access control. In this context, we have identified the following grand challenges.

Attack Vs. Defence		Organisational		Technical				
		Policy/Process	Awareness	Sandbox	AAA	Monitoring	Integrity	Machine Learning
Orchestration	TD1	Mitnick2001, Colwill2009, CPNI2013, Hinson2008	Bakhshi2009		Ruskov2014			Raskin2010, Stringhini2015
	TD2	CPNI2013, Sheng2010, Hinson2008						Drucker1999
	MD1-L	CPNI2013, GOVUK2015		Sandboxie2014, BufferZonePro2014, Peterson2002, Blasing2010, Greamo2011, Xiao2013	Salem2011, Ruskov2014			Singhal2012,
	MD1-R	CPNI2013, GOVUK2015, Frauenstein2013, Janson2011	Labuschagne2011, Sheng2010, Arachilage2012	Lu2010, Barth2008, Blasing2010	Boshmaf2012	Lee2012, Stringhini2013, Li2013, Lu2011, Ruskov2014	Webroot2013, FCS2009, Bhardwaj2014, Dietrich2013	Stringhini2015, Basnet2008, Aggarwal2012, Cova2010, Xiang2011, Bergholz2010, Garera2007, Fette2007, Drucker1999, Dong-Her2004, Lee2010
	MD2		Boileau2006, Duflot2011					
	MD3	GOVUK2015			EvilAP		EvilAP	
	MA1	Mitnick2001, Colwill2009		Salem2011				
	MA2			Lu2010	Schaff2013, Boshmaf2012	Lee2012, Li2013, Lu2011, Stringhini2013	Webroot2013	Aggarwal2012
Exploitation	DV1	Calder2010, Sheng2010, CPNI2013	Bakhshi2009, Gulenko2013, Labuschagne2011, Kirlappos2012, Arachilage2012, Kumaraguru2007, Kumaraguru2009, Lin2011, Wu2006, Lee2015, Sheng2007	Bianchi2015	Lu2011, EvilAP2015		Seirfert2013, Webroot2013, FCS2009, Hara2009, Dietrich2013, EvilAP	Xiang2011, Bergholz2010, Bergholz2008, Garera2007, Basnet2008, Fette2007, Lee2010 Dong-Her2004
	DV2	CPNI2013	Gulenko2013, Schechter2007, Anderson2013, Neupane2014	Bianchi2015	Salem2011		Alvisi2013	Singhal2012, Thomas2011, Cova2010, Raskin2010, Drucker1999, Lee2010
	DV3		Gulenko2013, Lin2011, Lee2015, Anderson2013, Neupane2014, Sheng2007, Wu2006	Bianchi2015	Lee2010		FCS2009, Bhardwaj2014, Dietrich2013	Aggarwal2012, Stringhini2015
	IM1	Calder2010	Bakhshi2009, Arachilage2012, Kumaraguru2007, Kumaraguru2009, Konak2012		Salem2011, Lee2010		FCS2009	Dong-Her2004, Basnet2008, Aggarwal2012, Xiang2011, Bergholz2010, Garera2007, Fette2007, Drucker1999, Lee2010
	IM2		Konak2012	Sandboxie2014, QubesOS2015, Microsoft2007, BufferZonePro2014, Bianchi2015, Barth2008		Lee2012, Li2013, Lu2011, Stringhini2013	Abdullah2011, Dhanalakshmi2010, FCS2009, Bhardwaj2014, Dietrich2013, EvilAP	Chandrasekaran2006, Singhal2012, Thomas2011, Basnet2008, Aggarwal2012, Cova2010, Xiang2011, Bergholz2010, Garera2007, Fette2007
Execution	AP1			Sandboxie2014, BufferZonePro2014, QubesOS2015, Blasing2010, Xiao2013, Barth2008, Lu2010	Neuman1994, Desmond2008, Motiee2010, Turner2010			
	AP2			Sandboxie2014, BufferZonePro2014, QubesOS2015, Blasing2010, Xiao2013, Barth2008, Lu2010, Bianchi2015				
	ES1			Sandboxie2014, BufferZonePro2014, Blasing2010, Xiao2013, Barth2008, Lu2010	Neuman1994, Desmond2008, Motiee2010, Turner2010			Stringhini2015
	ES2	Frauenstein2013			Neuman1994, Desmond2008, Turner2010			

Table III: Attack Vs. Defence Matrix

5.1. An authoritative semantic attack dataset

The lack of publicly available, up to date, relevant and reliable datasets is a pervasive issue in computer security research. In semantic attacks, the human factor makes the acquisition of datasets even more difficult. Researchers often conduct lab-based experiments, which allow for a controlled environment where several socio-technical metrics can be evaluated, but their realism is arguably limited. The alternative, which is to run "social experiments" at the scale required for collecting usable data is problematic. Even if the semantic attacks conducted are innocuous, their potential impact on the users is poorly understood and difficult to monitor. Crucially, debriefing is ethically vital for any deception-based research, but conducting effective debriefing in large-scale social experiments is not straightforward.

The problem is aggravated by the wide range and dynamic nature of deception mechanisms employed. By the time a research group studies a new attack, designs a large-scale experiment, recruits participants, conducts it, analyses and publishes the results, the particular attack may already be gradually abandoned in the wild in favour of newer ones. We propose that this can be somewhat alleviated by focusing not directly to specific attack types, but to the taxonomic characteristics shared between multiple attacks. An authoritative dataset that would be useful in this manner does not exist.

5.2. Predicting susceptibility to semantic attacks in real-time

A challenge related to the acquisition of an authoritative dataset is the identification of measurable semantic attack susceptibility indicators that are practically usable by technical defence developers. For instance, an interesting real-world study by Halevi et al. [2015] has identified conscientiousness as a personality trait that is highly correlated to a user's susceptibility to spear-phishing attacks. This might be helpful to social scientists and designers of awareness programs, but is of little use to a technical protection system's developer, because conscientiousness cannot be measured by a software system in real-time or automatically. What may be measurable, such as typing speed, time to click, typing errors, mouse movement, number of browser tabs open, time logged in, may not necessarily be relevant. A significant challenge is to identify automatically measurable features that do indeed correlate with susceptibility to semantic attacks.

5.3. Patching the user

An important strength of technical security measures is the ability to stay up to date. Following an antivirus update, which is automatically and rapidly delivered, a computer is protected effectively against several new and evolving threats. There is no equivalent concept for semantic attacks. A software application or embedded system may have a serious security flaw, but after this is discovered, the vendor can issue a patch that addresses it. Again, there is no equivalent concept for semantic attacks. The user, which is the entity of the system that is targeted, may take months or years to learn how to recognise and protect from new deception techniques.

The challenge here is to develop a mechanism by which a user's vulnerability to a new semantic attack vector can be reduced in a timely manner. This may involve elements of both dynamic access control and training. For example, a corporate user's access rights that are not core to their job role may be modified dynamically according to the indicators identified in Section 5.2. An access restriction may then be lifted once the user successfully completes a brief e-learning module or a practical assessment on the latest semantic attack methodologies. Of course, this highlights a further challenge, which is how to measure the effectiveness of user training or indeed of any measure designed to protect users and organisations from semantic attacks.

6. CONCLUSION

Technical countermeasures have traditionally struggled to address a good range of semantic attack types, especially the new and more dynamic ones. There is little doubt that policy control, user education through training and awareness campaigns, and other user management countermeasures can be effective, but to what extent is difficult to tell without a formal framework. It would probably be beneficial to combine technical and user management countermeasures into a single autonomous system, where users are proactively profiled in terms of their susceptibility to specific attack types and are allocated permissions accordingly and dynamically. Towards this vision, it is important to have an overall view of the semantic attack threat landscape. The taxonomy provided here is a step in tackling this challenge. It introduces a structured baseline for classifying semantic attacks by breaking them down into their components and thus allowing to identify countermeasures that are applicable to a range of different attacks that share a subset of their characteristics. We have complemented this taxonomy with a survey of defence measures, highlighting their suitability against the taxonomy's categories. In this manner, we aim to facilitate the design of more comprehensive defence mechanisms that can address each criterion of our classification rather than specific attack families.

REFERENCES

Z.H. Abdullah, N. I. Udzir, R. Mahmod, and K. Samsudin. 2011. Towards a dynamic file integrity monitor through a security classification. *Internal Journal of New Computer Architectures and their Applications (IJNCAA)* 3, Article 1 (2011), 766–779 pages.

S. Abraham and C.S. UnduShobha. 2010. An overview of social engineering malware: Trends, tactics, and implications. *Technology in Society* 3, 183–196, Article 32 (2010).

S. Abu-Nimeh and S. Nair. Phishing Attacks in a Mobile Environment. In *SMU HACNet Lab Southern Methodist University Dallas*.

M. Aburrous, M. A. Hossain, F. Thabatah, and K. Dahal. 2008. Intelligent phishing website detection system using fuzzy techniques. In *Information and Communication Technologies: From Theory to Applications (ICTTA 2008. 3rd International Conference on.)*. IEEE.

A. Adelsbach, S. Gajek, and J. Schwenk. 2005. Visual spoofing of SSL protected web sites and effective countermeasures. In *Information Security Practice and Experience*. Springer Berlin Heidelberg, 204–216.

A. Aggarwal, A. Rajadesingan, and P. Kumaraguru. 2012. PhishAri: Automatic realtime phishing detection on twitter. In *eCrime Researchers Summit (eCrime)*. IEEE, 1–12.

P. Agten, W. Joosen, F. Piessens, and N. Nikiforakis. 2015. Seven months' worth of mistakes: A longitudinal study of typosquatting abuse. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. NDSS.

A. Algarni, Y. Xu, T. Chan, and Y. C. Tian. 2013. Social engineering in social networking sites: Affect-based model. In *Internet Technology and Secured Transactions (ICITST) 2013 8th International Conference for*. IEEE, 508–515.

S. M. Ali. 2014. Integration of Information Security Essential Controls into Information Technology Infrastructure Library - A Proposed Framework. *International Journal of Applied* 1, Article 4 (2014).

L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi. 2013. SoK: The evolution of sybil defense via social networks. In *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 382–396.

B. Anderson and B. Anderson. 2010. *Seven deadliest USB attacks*. Syngress.

B. B. Anderson, C. B. Kirwan, J. L. Jenkins, D. Eargle, S. Howard, and A. Vance. 2013. How Polymorphic Warnings Reduce Habituation in the BrainInsights from an fMRI Study. In *Proceedings of of CHI15*.

G. N. A. Arachchilage, S. Love, and M. Scott. 2012. Designing a Mobile Game to Teach Conceptual Knowledge of Avoiding Phishing Attacks. *International Journal for e-Learning Security* 2, Article 2 (2012), 127–132 pages.

Arstechnica. 2014. Phishing scam that penetrated Wall Street just might work against you, too. (2014). http://arstechnica.com/security/2014/12/phishing-scam-that-penetrated-wall-street-just-might-work-against-you-too/

B. Atkins and W. Huang. 2013. A Study of Social Engineering in Online Frauds. *Open Journal of Social Sciences* 3, Article 1 (2013), 23–32 pages.

T. Bakhshi, M. Papadaki, and S. Furnell. 2009. Social engineering: assessing vulnerabilities in practice. *Information management and computer security,* 1, Article 17 (2009), 53–63 pages.

M. T. Banday, J. A. Qadri, and N. A. Shah. 2009. Study of Botnets and their threats to Internet Security. (2009). http://sprouts.aisnet.org/594/1/Botnet_Sprotus.pdf

A. Barth, C. Jackso, C. Reis, and TGC Team. 2008. The security architecture of the Chromium browser. (2008). http://seclah.stanford.edu/websec/chromium

R. Basnet, S. Mukkamala, and A. H. Sung. 2008. Detection of phishing attacks: A machine learning approach. In *Soft Computing Applications in Industry*. Springer Berlin Heidelberg, 373–383.

A. Bergholz, J. De Beer, S. Glahn, M. F. Moens, G. Paa, and S. Strobel. 2010. New filtering approaches for phishing email. *Journal of computer security* 1, Article 18 (2010), 7–35 pages.

A. Bergholz, J. H. Chang, G. Paa, F. Reichartz, and S. Strobel. 2008. Improved Phishing Detection using Model-Based Features. In *CEAS*.

T. Bhardwaj, K. T. Sharma, and M. R. Pandit. 2014. Social Engineering Prevention by Detecting Malicious URLs Using Artificial Bee Colony Algorithm. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving*. Springer India, 355–363.

A. Bianchi, J. Corbetta, L. Invernizzi, Y. Fratantonio, C. Kruegel, and G. Vigna. 2015. What the App is That? Deception and Countermeasures in the Android User Interface. In *36th IEEE Symposium on Security and Privacy*. IEEE.

L. Bilge and T. Dumitras. 2012. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, Vol. 10. ACM, 833–944.

G. S. Bindra. 2011. Masquerading as a trustworthy entity through Portable Document File (PDF) Format. In *Privacy, security, risk and trust (passat)*. IEEE, 784–789.

T. Blasing, L. Batyuk, A. D. Schmidt, S. A. Camtepe, and S. Albayrak. 2010. An android application sandbox system for suspicious software detection. In *Malicious and Unwanted Software (MALWARE) (5th International Conference on)*. IEEE, 55–62.

A. Boileau. 2006. Hit by a Bus: Physical Access Attacks with Firewire. (2006). http://www.security-assessment.com/files/presentations/ab_firewire_rux2k6-final.pdf

Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. 2011. The socialbot network: when bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*. ACM, 93–102.

Y. Boshmaf, I. Muslukhov, and K. Beznosov M. Ripeanu. 2012. Key challenges in defending against malicious socialbots. In *Proceedings of the 5th USENIX Conference on Large-scale Exploits and Emergent Threats, LEET (Vol. 12)*.

E. F. Brickell, J. F. Cihula, C. D. Hall, and R. Uhlig. 2011. Method of improving computer security through sandboxing. US patent No. 7,908,653. (2011).

J. M. Briones, M. A. Coronel, and P. Chavez-Burbano. 2013. Case of study: Identity theft in a university WLAN Evil twin and cloned authentication web interface. In *Computer and Information Technology (WCCIT), 2013 World Congress on*. IEEE, 1–4.

A. Calder and S. Watkins. *COBIT 5 for Information Security*. ITGI.

A. Calder and S. Watkins. 2010. *IT Governance: An International Guide to Data Security and ISO27001/ISO27002*. Kogan Page Publishers.

A. Calder and S. Watkins. 2014. *Framework for Improving Critical Infrastructure Cybersecurity*. National Institute of Standards and Technology (NIST) and United States of America.

F. Callegati, W. Cerroni, and M. Ramilli. 2009. Man-in-the-Middle Attack to the HTTPS Protocol. *IEEE Security and Privacy* 1, Article 7 (2009), 78-81 pages.

CESG. 2015. Common Cyber Attacks: Reducing The Impact. (2015). https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/400106/Common_Cyber_Attacks-Reducing_The_Impact.pdf

B. Chaffin. 2014. Someone Targets Hong Kong Protesters Using Jailbroken iPhones with Malware. (2014). http://www.macobserver.com/tmo/article/someone-targets-hong-kong-protesters-using-jailbroken-iphones-with-malware

M. Chandrasekaran and K. Narayanan S. Upadhyaya. 2006. Phishing email detection based on structural properties. In *NYS Cyber Security Conference*. 1–7.

T. M. Chen. 2003. Trends in viruses and worms. *The Internet Protocol Journal* 3, Article 6 (2003), 23-33 pages.

N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell. 2004. Client-Side Defense Against Web-Based Identity Theft. In *NDSS*.

M. Christodorescu and S. Jha. 2004. Testing malware detectors. *ACM SIGSOFT Software Engineering Notes* 4, Article 29 (2004), 34-44 pages.

Chromium. 2015. The Chromium Projects - Sandbox. (2015). http://www.chromium.org/developers/design-documents/sandbox

G. Cluley. 2011. A 419 scam via snail mail. Naked Security. (2011). Retrieved December 10th, 2013 from http://nakedsecurity.sophos.com/2011/05/30/a-419-scam-via-snail-mail

Z. Coburn and G. Marra. 2008. Realboy Believable twitter bots. (2008). http://ca.olin.edu/2008/realboy/

C. Colwill. 2009. Human factors in information security: The insider threat: Who can you trust these days? *Information security technical report* 4, Article 14 (2009), 186–196 pages.

Comodo. 2015. Demo of a URL-bar spoofing attack. (2015). http://www.contentverification.com/graphic-attacks/demo/

B. D Cone, C. E Irvine, M. F. Thompson, and T. D. Nguyen. 2007. A video game for cyber security training and awareness. *Computer and Security* 1, Article 26 (2007), 63–72 pages.

L. Corrons. 2010. The business of rogueware. In *Web Application Security (72)*, Vol. 10. 7–7.

M. Cova, C. Kruegel, and G. Vigna. 2010. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proceedings of the 19th international conference on World wide web*. ACM, 281–290.

CPNI. 2013. Social Engineering: Understanding the Threat. (2013). http://www.cpni.gov.uk/documents/publications/2013/2013065-social-engineering.pdf?epslanguage=en-gb

Darknet. 2015. EvilAP Defender Detect Evil Twin Attacks. (2015). http://www.darknet.org.uk/2015/04/evilap-defender-detect-evil-twin-attacks/

B. Desmond, J. Richards, R. Allen, and A. G. Lowe-Norris. 2008. *Active Directory: Designing, Deploying, and Running Active Directory*. O'Reilly Media.

R. Dhamija, D. J. Tygar, and M. Hearst. 2006. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM.

R. Dhanalakshmi and C. Chellappan. 2010. Detection and Recognition of File Masquerading for E-mail and Data Security. In *Recent Trends in Network Security and Applications*. Springer Berlin Heidelberg, 253–262.

C. Dhinakaran, J. K. Lee, and D. Nagamalai. 2009. 'Reminder: please update your details': Phishing Trends. In *Networks and Communications (NETCOM'09. First International Conference on)*. IEEE, 295–300.

C. Dietrich. 2013. *Identification and Recognition of Remote-Controlled Malware*. Ph.D. Dissertation. Universitsbibliothek Mannheim.

S. Dong-Her, C. Hsiu-Sen, C. Chun-Yuan, and B. Lin. 2011. Internet security: malicious e-mails detection and protection. *Industrial Management and Data Systems* 7, Article 104 (2011), 613–623 pages.

A. Doupe, M. Egele, B. Caillat, G. Stringhini, G. Yakin, A. Zand, and G. Vigna. 2011. Hit'em where it hurts: a live security exercise on cyber situational awareness. In *Proceedings of the 27th Annual Computer Security Applications Conference*. ACM, 51–61.

C.E. Drake, J. O. Jonathan, and J.K. Eugene. 2004. Anatomy of a Phishing Email. In *CEAS*.

H. Drucker, S. Wu, and V. N. Vapnik. 1999. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on* 5, Article 10 (1999), 1048-0154 pages.

P. Ducklin. 2014a. Anatomy of an Android SMS virus - watch out for text messages, even from your friends! (2014). https://nakedsecurity.sophos.com/2014/06/29/anatomy-of-an-android-sms-virus-watch-out-for-text-messages-even-from-your-friends/

P. Ducklin. 2014b. Return of the Android SMS virus - self-spreading. (2014).

L. Duflot, Y. A. Perez, and B. Morin. 2011. What if you cant trust your network card?. In *Recent Advances in Intrusion Detection*. Springer Berlin Heidelberg, 378–397.

M. Eeckhaut and N. Vanhecke. 2014. De Standaard: Belgian professor in cryptography hacked. (2014). http://www.standaard.be/cnt/dmf20140201_011

M. Egele, D. Brumley Y. Fratantonio, and C. Kruegel. 2013. An empirical study of cryptographic misuse in android applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*. ACM, 73–84.

M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda. 2008. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer Berlin Heidelberg, 88–106.

M. Eminagaoglu, E. Ucar, and S. Eren. 2009. The positive outcomes of information security awareness training in companies A case study. *Information security technical report* 4, Article 14 (2009), 223–229 pages.

D. Emm. 2005. The Changing Face of Malware. In *Proceedings of the IWWST*.

A. P. Felt and D. Wagner. 2011. *Phishing on mobile devices*. na.

I. Fette, N. Sadeh, and A. Tomasic. 2007. Learning to detect phishing emails. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 649–656.

Mozilla Firefox. 2015. Mozilla Wiki - Security/Sandbox. (2015). https://wiki.mozilla.org/Security/Sandbox

FirstCyberSecurity. 2009. Protecting Your Brand Online and Creating Customer Confidence. (2009). http://www.firstcybersecurity.com/main/IPRiskMReview.pdf

Dennis Fisher. 2015. Massive, Decades-Long Cyber espionage Framework Uncovered. (2015). http://threatpost.com/massive-decades-long-cyberespionage-framework-uncovered/111080d

C. Foozy, R. Ahmad, M. Abdollah, R. Yusof, and M. Zaki. 2011. Generic Taxonomy of Social Engineering Attack. In *Malaysian Technical Universities International Conference on Engineering and Technology*. 527–533.

S. Ford, M. Cova, C. Kruegel, and G. Vigna. 2009. Analyzing and detecting malicious flash advertisements. In *Computer Security Applications Conference (ACSAC'09. Annual)*. IEEE.

E.D. Frauenstein and R.von Solms. 2013. An Enterprise Anti-phishing Framework. *Information Assurance and Security Education and Training* (2013), 196–203.

S. Garera, N. Provos, M. Chew, and A. D. Rubin. 2007. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*. ACM, 1.

D. Gavrilut, M. Cimpoesu, D. Anton, and L. Ciortuz. 2009. Malware detection using machine learning. In *Computer Science and Information Technology (IM-CSIT'09. International Multiconference on)*. IEEE, 735–741.

A. Gazet. 2010. Comparative analysis of various ransomware virii. *Journal in computer virology* 1, Article 6 (2010), 77-90 pages.

J. Giles. 2010. Scareware the inside story. *New Scientist* 2753, Article 205 (2010), 38–41 pages.

GOVUK. 2015. 10 Steps to Cyber Security. (2015). https://www.gov.uk/government/publications/cyber-risk-management-a-board-level-responsibility/10-steps-summary

C. Greamo and A.Ghosh. 2011. Sandboxing and virtualisation - Modern tools for combating malware. In *Security and Privacy. (9)*, Vol. 2. IEEE, 79–82.

S. Guillaume, H. Carlo, A. Matthieu, J. Marianne, and M. Romain. 2014. RISK-DET: ICT Security Awareness Aspect Combining Education and Cognitive Sciences. In *ICCGI 2014, The Ninth International Multi-Conference on Computing in the Global Information Technology*. 51–53.

I. Gulenko. 2013. Social against social engineering: Concept and development of a Facebook application to raise security and risk awareness. *Information Management and Computer Security* 2, Article 21 (2013), 91–101 pages.

T. Halevi, N. Memon, and O. Nov. 2015. Spear-Phishing in the Wild: A Real-World Study of Personality, Phishing Self-Efficacy and Vulnerability to Spear-Phishing Attacks. (2015). http://dx.doi.org/10.2139/ssrn.2544742

M. Hara, A. Yamada, and Y. Miyake. 2009. Visual similarity-based phishing detection without victim site information. In *Computational Intelligence in Cyber Security, 2009. CICS'09. IEEE Symposium on*. IEEE, 30–36.

M. Hasan and N.B. Prajapati. 2009. An Attack Vector for Deception Through Persuasion Used by Hackers and Crackers. In *Networks and Communications (NETCOM'09. First International Conference on)*. IEEE, 254–258.

R. Heartfield and G. Loukas. 2013. On the feasibility of automated semantic attacks in the cloud. In *Computer and Information Sciences III*. Springer London, 343–351.

G. Hinson. 2008. Social Engineer Techniques, Risks and Controls. *The EDP Audit, Control and Security Newsletter* (2008), 32–45.

J. Hong. 2012. The state of phishing attacks. (2012).

F. Howard and O. Komili. 2010. Poisoned search results: How hackers have automated search engine poisoning attacks to distribute malware. *Sophos Technical Papers* (2010).

H. Huang, S. Zhong, and J. Tan. 2009. Browser-side countermeasures for deceptive phishing attack. In *Information Assurance and Security (IAS'09. Fifth International Conference on)*, Vol. 1. IEEE.

M. Huber, M. Mulazzani, and E. Weipp. 2010. Who on Earth Is Mr. Cypher: Automated Friend Injection Attacks on Social Networking Sites. In *Security and Privacy Silver Linings in the Cloud*. Springer Berlin Heidelberg, 80–89.

H. Hwang, G. Jung, K. Sohn, and S. Park. 2008. A study on MITM (Man in the Middle) vulnerability in wireless network using 802.1 X and EAP. In *Information Science and Security (ICISS)*. IEEE, 164–170.

Invincea. 2014. Sandboxie. (2014). http://www.sandboxie.com/

K. Ivaturi and L. Janczewski. 2011. A taxonomy for social engineering attacks. In *CONF-IRM Proceedings*.

J. R. Jacobs. 2011. *Measuring the Effectiveness of the USB Flash Drive as a Vector for Social Engineering Attacks on Commercial and Residential Computer Systems*. Master's thesis. Embry-Riddle Aeronautical University.

T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. 2007. Social phishing. *Commun. ACM* 10, Article 50 (2007), 49–51 pages.

W. Jansen and T. Grance. 2011. Guidelines on security and privacy in public cloud computing. *NIST special publication 800* (2011).

J.Corbetta, L. Invernizzi, C. Kruegel, and G. Vigna. 2014. Eyes of a Human, Eyes of a Program: Leveraging Different Views of the Web for Analysis and Detection. In *17th International Symposium, RAID 2014*. Springer, 130–149.

M. E. Johnson, D. McGuire, and N. D. Willey. 2008. The evolution of the peer-to-peer file sharing industry and the security risks for users. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*. IEEE, 383–383.

M. Jordan and G. Heather. 2005. The signs, signifiers and semiotics of the successful semantic attack. In *14th Annual EICAR Conference*. 344–364.

A. Kalafut, A. Acharya, and M. Gupta. 2006. A study of malware in peer-to-peer networks. In *Proceedings of 6th ACM SIGCOMM conference on Internet measurement*. ACM, 327–332.

Keelog. 2015. KeeLog Key Grabber Internal Module PS2 2GB. (2015). https://www.keelog.com/

I. Kirlappos and M.A Sasse. 2012. Security Education against Phishing: A Modest Proposal for a Major Rethink. *IEEE Security and Privacy Magazine* 2, Article 10 (2012), 24–32 pages.

A. Konak and M. Bartolacci. 2012. Broadening E-Commerce Information Security Education Using Virtual Computing Technologies. In *2012 Networking and Electronic Commerce Research Conference*.

B. Krishna. 2011. Malicious emails masquerade as office printer messages. Symantec Connect Blog - Symantec Intelligence.ONLINE. (2011). http://www.symantec.com/connect/blogs/malicious-emails-masquerade-office-printer-messages-0

E. Kritzinger and S. H. von Solms. 2010. Cyber security for home users: A new way of protection through awareness enforcement. *Computer and Security* 8, Article 29 (2010), 840–847 pages.

A. H. Kruger and D. K. Wayne. 2006. A prototype for assessing information security awareness. *Computers and Security* 4, Article 25 (2006), 289–296 pages.

R. Kuipers, E. Starck, and H. Heikkinen. 2010. Smart TV Hacking: Crash Testing Your Home Entertainment. (2010). http://www.codenomicon.com/resources/whitepapers/codenomicon-wp-smart-tv-fuzzing.pdf.

P. Kumaraguru. 2009. *PhishGuru: a system for educating users about Semantic Attacks*. Ph.D. Dissertation. Carnegie Mellon University.

T. Lauinger, V. Pankakoski, D. alzarotti, and E. Kirda. 2010. Honeybot, your man in the middle for automated social engineering. In *LEET10, 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats*.

A. Laurie and B. Laurie. 2003. Serious flaws in bluetooth security lead to disclosure of personal data. (2003).

N. Leavitt. 2005. Instant Messaging: A new target for hackers. *Computer* 7, Article 38 (2005), 20–23 pages.

J. Lee, L. Bauer, and M. L. Mazurek. 2015. Studying the effectiveness of security images in Internet banking. *IEEE Internet Computing* (2015).

K. Lee, J. Caverlee, and S. Webb. 2010. The social honeypot project: protecting online communities from spammers. In *Proceedings of the 19th international conference on World wide web*. ACM.

S. Lee and J. Kim. 2012. WarningBird: Detecting Suspicious URLs in Twitter Stream. In *NDSS*. NDSS.

X. Leroy. 2001. Java bytecode verification: an overview. In *Computer Aided Verification*. Springer Berlin Heidelberg.

Z. Li, S. Alrwais, Y. Xie, F. Yu, and X. Wang. 2013. Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 112–126.

Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang. 2012. Knowing your enemy: understanding and detecting malicious web advertising. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM.

E. Lin, S. Greenberg, E. Trotter, D. Ma, and J. Aycock. 2011. Does domain highlighting help people identify phishing sites?. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2075–2084.

G. Loukas. 2015. *Cyber-Physical Attacks: A Growing Invisible Threat*. Butterworth-Heinemann (Elsevier).

L. Lu, R. Perdisci, and W. Lee. 2011. Surf: detecting and measuring search poisoning. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 467–476.

L. Lu, V. Yegneswaran, P. Porras, and W. Lee. 2010. Blade: an attack-agnostic approach for preventing drive-by malware infections. In *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 440–450.

G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger. 2008. NFC devices: Security and privacy. In *Availability, Reliability and Security (ARES 08)*. IEEE, 642–647.

M. Mannan and P. C. van Oorschot. 2005. On instant messaging worms, analysis and countermeasures. In *Proceedings of the 2005 ACM workshop on Rapid malcode*. ACM, 2–11.

C. Marforio, F. Aurelien, and S. Capkun. 2011. *Application collusion attack on the permission-based security model and its implications for modern smartphone systems. Report 724*. Technical Report.

N. P. P. Mavromatis and M. A. R. F. Monrose. 2008. All your iframes point to us. In *USENIX Security Symposium*. USENIX, 1–16.

K. F. McCrohan, K. Engel, and J. W. Harvey. 2010. Influence of awareness and training on cyber security. *Journal of Internet Commerce* 1, Article 9 (2010), 23–41 pages.

Microsoft. 2007. The Windows Vista and Windows Server 2008 Developer Story: Windows Vista Application Development Requirements for User Account Control. (2007). https://msdn.microsoft.com/en-us/library/aa905330.aspx

M. Wu R. C. Miller and S. L. Garfinkel. 2006. Do security toolbars actually prevent phishing attacks?. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 601–610.

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

S. Motiee, K. Hawkey, and K. Beznosov. 2010. Do windows users follow the principle of least privilege?: investigating user account control practices. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*. ACM.

H. Xu N. Wang and J. Grossklags. 2011. Third-party apps on Facebook: privacy and the illusion of control. In *Proceedings of the 5th ACM Symposium on Computer Human Interaction for Management of Information Technology*. ACM.

B. C. Neuman and T. Ts'o. 1994. Kerberos: An authentication service for computer networks. *Communications Magazine* 9, Article 32 (1994), 33-38 pages.

A. Neupane, N. Saxena, K. Kuruvilla, and M. Georgescu R. Kana. 2014. Neural Signatures of User-Centered Security: An fMRI Study of Phishing, and Malware Warnings. In *Proceedings of the Network and Distributed System Security Symposium*. NDSS, 1–16.

K. Nohl and J. Lehl. 2014. BadUSBOn accessories that turn evil. In *Black Hat USA*.

H. Orman. 2009. The compleat story of phish. *IEEE Internet Computing* 1, Article 17 (2009), 87-91 pages.

Qubes OS. 2015. Qubes OS Project. (2015). https://www.qubes-os.org/

A. Acquisti L. F. Cranor J. Hong P. Kumaraguru, Y. Rhee and E. Nunge. 2007. Protecting People from Phishing: The Design and Evaluation of an Embedded Training Email System. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM.

Pierluigi Paganini. 2014. Phishing goes mobile with cloned banking app into Google Play. (2014). http://securityaffairs.co/wordpress/26134/cyber-crime/phishing-goes-mobile-cloned-banking-app-google-play.html

R. T. Peltier. 2013. *Information Security Policies, Procedures, and Standards: guidelines for effective information security management*. CRC PRess.

D. S. Peterson, M. Bishop, and R. Pandey. 2002. A flexible containment mechanism for executing untrusted code. In *Proceedings of the 11th USENIX Securit Symposium (5th International Conference on)*. IEEE, 207–225.

L. Phifer. 2000. Top Ten Wi-Fi Security Threats. (2000). http://www.esecurityplanet.com/views/article.php/3869221/Top-Ten-WiFi-Security-Threats.htm

A. Podhradsky, R. DOvidio, P. Engebretson, and C. Casey. 2013. Xbox 360 Hoaxes, Social Engineering, and Gamertag Exploits. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*. IEEE, 3239–3250.

BufferZone Pro. 2014. BufferZone-Pro. (2014). http://www.trustware.com/BufferZone-Pro/

N. Provos, M. A. Rajab, and P. Mavrommatis. 2009. Cybercrime 2.0: when the cloud turns dark. *Commun. ACM* 4, Article 52 (2009), 42–47 pages.

A. Raskin. 2011. Tabnabbing: A new type of phishing attack. (2011).

V. Raskin, J. M. Taylor, and C. F. Hempelmann. 2010. Ontological semantic technology for detecting insider threat and social engineering. In *Proceedings of the 2010 workshop on New security paradigms*. ACM.

G. W. Romney, J. K. Jones, B. L. Rogers, and P. MacCabe. 2005. IT security education is enhanced by analyzing Honeynet data. In *Information Technology Based Higher Education and Training (ITHET 2005. 6th International Conference on)*. IEEE.

I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar. 2010. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *19th USENIX Security Symposium*.

RSA. 2012. Lions at the Watering Hole The VOHO Affair. (2012). https://blogs.rsa.com/lions-at-the-watering-hole-the-voho-affair/

M. Ruskov, P Ekblom, and M. A. Sasse. 2014. Towards a Simulation of Information Security Behaviour in Organisations. In *Cyberpatterns*. Springer International Publishing, 177–184.

M. B. Salem and S. J. Stolfo. 2011. Modeling user search behavior for masquerade detection. In *Recent Advances in Intrusion Detection*. Springer Berlin Heidelberg.

H. Sandouka, A. J. Cullen, and I. Mann. 2009. Social Engineering Detection using Neural Networks. In *CyberWorlds CW'09 International Conference on*. IEEE, 273–278.

G. Schaff, C. Harpes, R. Martin, and M. Junger. 2013. An application to estimate the cyber-risk detection skill of mobile device users (IDEA). (2013). http://doc.utwente.nl/87117/1/SCHAFF_itrust-scientific_article_GSC_(3).pdf

S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer. 2007. The emperor's new security indicators. In *IEEE Symposium on Security and Privacy, 2007*. IEEE, 51–65.

B. Schneier. 2000. Inside risks: semantic network attacks. *Commun. ACM* 12, 168, Article 43 (2000).

B. Schneier. 2011. *Secrets and lies: digital security in a networked world*. Wiley.

C. Seifert, J. W. Stokes, C. Colcernian, J. C. Platt, and L. Lu.

K. Selvaraj and N. F. Gutierrez. 2010. The rise of PDF malware. Symantec Security Response. (2010).

SensePost. 2014. Snoopy. (2014). https://github.com/sensepost/Snoopy

V. Sharma. 2011. An analytical survey of recent worm attacks.. In *IJCSNS (11)*, Vol. 11.

S. Sheng, M. Holbrook, P. Kumaraguru, and L. F. Cranor J. Downs. 2010. Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 373–382.

S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti L. F. Cranor, J. Hong, and E. Nunge. 2007. Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish. In *Proceedings of the 3rd symposium on Usable privacy and security*. ACM, 88–99.

S. Shin, J. Jung, and H. Balakrishnan. 2006. Malware prevalence in the KaZaA file-sharing network. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM.

P. Singhal and N. Raul. 2012. Malware detection module using machine learning algorithms to assit in centralized security in enterprise networks. *International Journal 4* (2012).

SocialEngineer. 2013. The Power of the Uniform in Social Engineering. Naked Security. (2013). Retrieved June 22nd, 2013 from https://www.social-engineer.com/the-power-of-the-uniform-in-social-engineering/

Y. Song, C. Yang, and G. Gu. 2010. Who is peeping at your passwords at Starbucks?To catch an evil twin access point. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*. IEEE, 323–332.

A. Sood and R. Enbody. 2014. *Targeted Cyber Attacks: Multi-staged Attacks Driven by Exploits and Malware*. Syngress.

K. E. Stewart, J. W. Humphries, and T. R. Andel. 2009. Developing a virtualization platform for courses in networking, systems administration and cyber security education. In *Proceedings of the 2009 Spring Simulation Multiconference. Society for Computer Simulation International*.

G. Stringhini, C. Kruegel, and G. Vigna. 2013. Shady paths: Leveraging surfing crowds to detect malicious web pages. In *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*. ACM, 133–144.

G. Stringhini and O. Thonnard. 2015. That Aint You: Blocking Spearphishing Through Behavioral Modelling. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 78–97.

D. Sullivan. 2008. What Is Search Engine Spam? The Video Edition, url =. (2008).

Symantec. 2014. Trojan.Ransomcrypt.I. (2014). http://www.symantec.com/security_response/writeup.jsp?docid=2014-051514-5659-99

J. Szurdi, B. Kocso, G. Cseh, J. Spring, M. Felegyhazi, and C.e Kanich. 2014. The long taile of typosquatting domain names. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX, 191–206.

M. Tavallaee, N. Stakhanova, and A. A. Ghorbani. 2010. Toward credible evaluation of anomaly-based intrusion-detection methods. In *Systems, Man, and Cybernetics, Part C: Applications and Reviews (IEEE Transactions on 40)*, Vol. 5. IEEE, 516–524.

P. Tetri and J. Vuorinen. 2013. Dissecting social engineering. In *Behaviour and Information Technology (32)*, Vol. 10. 1014–1023.

K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. 2011. Design and evaluation of a real-time url spam filtering service. In *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 447–462.

P. Thompson. 2007. *Deception as a semantic attack*. Chapman and Hall/CRC, Chapter 2.2, 125–144.

TrendMicro. 2014. Malaysia Airlines Flight 370 News Used To Spread Online Threats. (2014). http://blog.trendmicro.com/trendlabs-security-intelligence/malaysia-airlines-flight-370-news-used-to-spread-online-threats/

B. Turner, D. Lundell, J. Zamora, and C. Calderon. 2010. *Microsoft Forefront Identity Manager 2010 Technical Overview*. Technical Report.

US-CERT. 2015. Lenovo Computers Vulnerable to HTTPS Spoofing. (2015). https://www.us-cert.gov/ncas/current-activity/2015/02/20/Lenovo-Computers-Vulnerable-HTTPS-Spoofing

I. Burke W. A. Labuschagne, N. Veerasamy and M. M. Eloff. 2011. Design of cyber security awareness game utilizing a social media framework. In *Information Security South Africa (ISSA)*. IEEE.

Webroot. 2013. Webroot Real-Time Anti-Phishing Service. (2013). http://www.webroot.com/shared/pdf/WAP-Anti-Phishing-102013.pdf

G. Xiang, J. Hong, C. P. Rose, and L. Cranor. 2011. CANTINA+: a feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)* 2, Article 14 (2011), 21 pages.

H. Xiao and B. Zhao. 2013. Analysis on Sandbox Technology of Adobe Reader X. In *Computational and Information Sciences (ICCIS) (5th International Conference on)*. IEEE.

K. P. Yee. 2005. *Guidelines and strategies for secure interaction design.* Chapter 13, 247–273. http://sid.toolness.org/ch13yee.pdf

Received ; revised ; accepted