IEICE TRANS. FUNDAMENTALS, VOL.E104–A, NO.1 JANUARY 2021

PAPER *Special Section on Cryptography and Information Security*

To Get Lost is to Learn the Way: An Analysis of Multi-Step Social Engineering Attacks on the Web*

Takashi KOIDE[†,††a)], ***Nonmember***, **Daiki CHIBA**[†], **Mitsuaki AKIYAMA**[†], **Katsunari YOSHIOKA**[††,†††], ***and*** **Tsutomu MATSUMOTO**[††,†††], ***Members***

SUMMARY Web-based social engineering (SE) attacks manipulate users to perform specific actions, such as downloading malware and exposing personal information. Aiming to effectively lure users, some SE attacks, which we call multi-step SE attacks, constitute a sequence of web pages starting from a landing page and require browser interactions at each web page. Also, different browser interactions executed on a web page often branch to multiple sequences to redirect users to different SE attacks. Although common systems analyze only landing pages or conduct browser interactions limited to a specific attack, little effort has been made to follow such sequences of web pages to collect multi-step SE attacks. We propose StraySheep, a system to automatically crawl a sequence of web pages and detect diverse multi-step SE attacks. We evaluate the effectiveness of StraySheep's three modules (landing-page-collection, web-crawling, and SE-detection) in terms of the rate of collected landing pages leading to SE attacks, efficiency of web crawling to reach more SE attacks, and accuracy in detecting the attacks. Our experimental results indicate that StraySheep can lead to 20% more SE attacks than Alexa top sites and search results of trend words, crawl five times more efficiently than a simple crawling module, and detect SE attacks with 95.5% accuracy. We demonstrate that StraySheep can collect various SE attacks, not limited to a specific attack. We also clarify attackers' techniques for tricking users and browser interactions, redirecting users to attacks.

key words: *social engineering attacks, browser automation, web crawler*

1. Introduction

Attackers use social engineering (SE) techniques to lure users into taking specific actions. Modern web-based attacks leverage SE for malware infections [2], [3] and online frauds [4]–[6], which are called *web-based SE attacks* (or simply *SE attacks*). Attackers skillfully guide a user's browser interaction through attractive web content or warning messages to make users download malware or leak sensitive information. For example, to download pirated games, a user clicks a download button on an illegal downloading web page. Then, a popup window with a virus-infection alert is displayed. A user who believes the fake information clicks a "confirm" button and downloads fake anti-virus software [7].

Common systems to automatically collect SE attacks involve accessing web pages collected from search engines [5], [6], [8]. These systems use a web browser to crawl web pages and identify a particular SE attack by extracting features only from each web page. However, some types of SE attacks constitute a sequence of web pages starting from a landing page and require browser interaction (e.g., clicking an HTML element) at each web page to reach the attacks, which we call *multi-step SE attacks*. This is because each web page gradually convinces a user by using different psychological tactics [3]. Also, different browser interactions executed on a web page often branch to multiple sequences, redirecting users to different SE attacks, because there are multiple attack scenarios corresponding to a user's interests or psychological vulnerabilities. Although current systems analyze only landing pages or conduct browser interactions limited to a specific attack, little effort has been made to follow such sequences of web pages to collect multi-step SE attacks.

We propose StraySheep, a system to automatically crawl the sequence of web pages and detect diverse multi-step SE attacks derived from a landing page. StraySheep is based on two key ideas. The first idea is to simulate the multi-step browsing behaviors of users, that is, intentionally follow the sequence of web pages by selecting possible elements that psychologically attract users to lead them to SE attacks. StraySheep not only follows a single sequence of web pages but also crawls multiple sequences derived from a landing page. The second idea is to extract features from reached web pages as well as an entire sequence of web pages. Unlike previous approaches that extract features from a single web page they have visited [5], [6] or identify malicious URL chains automatically caused without user interactions (i.e., URL redirections) [9]–[11], StraySheep extracts features from the entire sequence of web pages it has actively and recursively followed. That is, StraySheep analyzes image and linguistic characteristics of reached web pages, browser events (e.g., displaying popup windows and alerts) that occurred before reaching the web page, and browser interactions that lead users to SE attacks. These features represent common characteristics of all SE attacks, i.e., persuading and deceiving users. Therefore, by combining these features to classify sequences, StraySheep detects various multi-step SE attacks more accurately. We implemented StraySheep

Manuscript received March 13, 2020.
Manuscript revised September 23, 2020.

[†]The authors are with NTT Secure Platform Laboratories, Tokyo, 180-8585 Japan.
[††]The authors are with Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama-shi, 240-8501 Japan.
[†††]The authors are with Institute of Advanced Sciences, Yokohama National University, Yokohama-shi, 240-8501 Japan.

[*]This paper is the extended version of the paper presented at ASIA CCS '20 [1].

a) E-mail: takashi.koide.fk@hco.ntt.co.jp

DOI: 10.1587/transfun.2020CIP0005

Copyright © 2021 The Institute of Electronics, Information and Communication Engineers

with three distinct modules (*landing-page-collection*, *web-crawling*, and *SE-detection*) to automatically collect landing pages, crawl the web pages branching from them, and detect SE attacks using the results of web crawling, respectively.

To determine the effectiveness of StraySheep's three modules, we conducted three evaluations: the rate of collected landing pages leading to SE attacks, the efficiency of web crawling to reach more SE attacks, and accuracy in detecting the attacks. The first evaluation demonstrated that landing pages gathered by the landing-page-collection module led to 20% more SE attacks than Alexa top sites and search results of trend words. The second evaluation demonstrated that the web-crawling module is five times more efficient at crawling than simple crawling modules. The third evaluation revealed that the SE-detection module identified SE attacks with 95.5% accuracy.

We analyzed collected multi-step SE attacks StraySheep in detail. As a result of categorizing SE attacks, we found that StraySheep reached a variety of SE attacks such as malware downloads, unwanted browser extension installs, survey scams, and technical support scams. We also found that 30% of SE attacks were reached from 25 different advertising providers.

The main contributions of this paper are as follows:

- We propose StraySheep, which detects multi-step SE attacks by automatically and recursively crawling sequences of web pages branching from landing pages. StraySheep can crawl and detect these attacks by simulating multi-step browsing behaviors of users and extracting features from an entire sequence of web pages.
- We evaluated StraySheep's three modules. The landing-page-collection module led to 20% more SE attacks than Alexa top sites and search results of trend words. The web-crawling module was five times more efficient at crawling than a simple crawling module. The SE-detection module identified SE attacks with 95.5% accuracy.
- We conducted a detailed analysis of multi-step SE attacks collected using StraySheep. We found that StraySheep collected various SE attacks, not limited to a specific attack. We analyzed attackers' techniques of luring users and browser interactions leading users to attacks.

2. Background

SE is used to manipulate people into performing a particular action by exploiting their psychology and has been widely used in various types of web-based attacks, such as malware downloads [3], [12], malicious browser extension installs [13]–[15], survey scams [6], and technical support scams [4], [5]. Malware downloads and malicious browser extension installs are achieved by masquerading as legitimate software. Survey scams recruit users attracted by fake survey rewards to trick them into providing sensitive information and accessing web pages controlled by attackers.

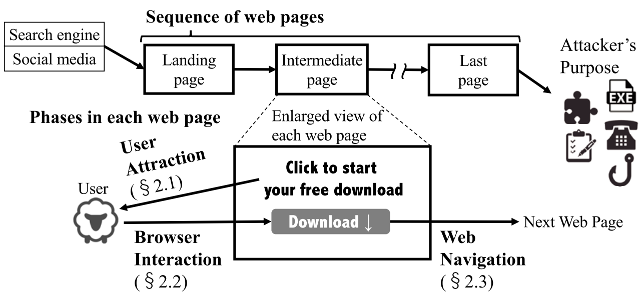


Fig. 1 Sequence of web pages in multi-step SE attacks and phases in each web page.

Technical support scams are carried out by persuading users to make a call to a fake technical support desk and install keystroke loggers, remote access tools, or malware.

Multi-step SE attacks use multiple web pages leveraging different psychological tactics to effectively lure users to the succeeding web page. Figure 1 shows a sequence of web pages in multi-step SE attacks and three simplified phases in each web page: *user attraction*, *browser interaction*, and *web navigation*. Therefore, the three phases can be repeated multiple times, starting from a landing page, which appears in response to clicking on a search-engine result or social-media link. Different user interactions on a single web page also lead to different SE attacks.

2.1 User Attraction

The user-attraction phase attracts a user psychologically by using the content of the web page to deceive and persuade the user to induce browser interaction [3]. For example, these web pages advertise free downloads of video games, threaten users with fake virus warnings, and request bogus software updates. The main purpose of this psychological attraction is to make the user interact with an HTML element (e.g., `a` and `div`) that navigates to malware downloads or a web page controlled by an attacker. We call such HTML elements *lure elements*. What is common with lure elements is that they contain words or shapes indicating the behavior or category of an element. A lure element is characterized by its visual effects, such as easily understandable download buttons containing "Click here to download" and movie play buttons containing "WATCH NOW" or a triangle pointing right. A lure element is also characterized by containing words such as "`download-btn`" and "`video-play-link`" in their text content and document object model (DOM) attributes such as `id`, `class`, and `alt`. Multiple lure elements may be arranged on a single web page. In this case, clicking these lure elements results in different SE attacks.

2.2 Browser Interaction

Users who are acted upon by the previous user-attraction phase are guided to interact with lure elements on the web page. This browser-interaction phase is mainly an explicit click on the lure element but also includes an unintended click [8]. For example, unintended clicks include clicking an overlay on the entire web page, context menu, and the browser's back button. These clicks are forcibly generated by JavaScript to redirect a user to a new web page or show a popup window against the user's intention.

2.3 Web Navigation

In the web-navigation phase, browser events occur as a result of browser interaction. These browser events redirect to another web page in the current window or a new window (popup), display alert dialogs, and download files. Web-page redirection occurs in an intermediate step of a multi-step SE attack, guiding the user to the next web page. On the next page, another user attraction, browser interaction, and web navigation could occur again. The purpose of repeatedly making a user reach multiple web pages without completing the attack on one web page is to gradually convince the user and increase the success rate of the attack. For example, to increase the attack-success rate of a user who watches a movie on an illegal streaming site, attackers display a popup that offers a dedicated video player with an alert dialog such as "Please install HD Player to continue." instead of providing an automatic software download on the first web page. Also, the multiple sequences of web pages in a multi-step SE attack often branch from the landing or intermediate web pages because such web pages contain two or more lure elements leading to different pages.

2.4 Problems on Collecting SE Attacks

There are three approaches to automatically collect SE attacks: tracing web traffic, archiving with a crawler, and crawling with a web browser. We give a brief introduction of these approaches and their limitations then present the requirements of collecting SE attacks.

The first approach is of reconstructing SE attacks from web traffic obtained through passive network monitoring [3]. To take measures against SE attacks, revealing a single SE attack reached from the landing page is useful, but uncovering all attacks that branch from web pages is more critical. However, this approach is used to observe only a single sequence of web pages accessed by the user. Also, it cannot be used to observe SE attacks starting from arbitrary web pages. That is, it cannot be used to observe attacks from which a user was not affected but another user could be affected.

The second approach is to visit each web page using crawlers such as Heritrix [16] and GNU Wget. Such crawlers extract links from a downloaded HTML source code of a web page and crawl them recursively. This approach can solve the problem with the first approach, in which it cannot collect SE attacks that the user did not reach because it can input an arbitrary URL. However, these crawlers can only execute simple content downloads and static content parsing. SE attacks often use web content dynamically generated by JavaScript, which require user interactions to navigate to the next pages; thus, these types of crawlers cannot collect most SE attacks.

The third approach is of web-browser automation using a tool such as Selenium [17]. Web-browser automation enables us to simulate user interaction to all elements on each web page. With this approach, we can solve the problems with the second approach. If we apply the idea of following all links with the second approach to web-browser automation, that is, clicking all elements on each web page, we can ideally collect all multi-step SE attacks derived from a landing page. However, recursively following all elements takes a significant amount of time because the browser requires time to run JavaScript and render web pages.

In summary, to efficiently observe multi-step SE attacks in a short time, the number of elements to crawl must be reduced by selecting possible lure elements from thousands of HTML elements on each web page. To analyze multi-step SE attacks in detail, it is also necessary to recursively follow multiple sequences of web pages that lead to SE attacks derived from a landing page rather than tracing only a single sequence of web pages. Therefore, requirements for collecting and analyzing multi-step SE attacks are crawling with the web-browser-automation approach, selecting lure elements that will likely lead to SE attacks, and recursively interacting with lure elements.

3. StraySheep

We propose a system called StraySheep that automatically collects landing pages that lead to SE attacks, crawls web pages, and detects multi-step SE attacks. StraySheep consists of three modules: *landing-page-collection*, *web-crawling*, and *SE-detection*. An overview of StraySheep is shown in Fig. 2. The landing-page-collection module gathers URLs of web pages leading to SE attacks by leveraging search engines and social media. The web-crawling module starts recursive web crawling from the URLs collected by the landing-page-collection module, selects and clicks on lure elements, and outputs a WebTree. A WebTree consists of tree-like abstract data, including logs such as web navigation, browser interaction, and snapshot (screenshot and HTML source code) observed at each web page branching from a landing page. The SE-detection module extracts features from a WebTree and identifies the multi-step SE attack using a classification model.

3.1 Landing-Page-Collection Module

The landing-page-collection module leverages search engines and social media to find landing pages as input for the web-crawling module. Many SE attacks use web pages that have copyright infringement, such as illegal downloads and free video streaming, to draw the attention of incautious users [8], [18]. To induce a user to access such web pages, attackers use search-engine-optimization techniques [19]–[21] and post messages on social media, which include links to the landing pages [22]–[24]. Examples of

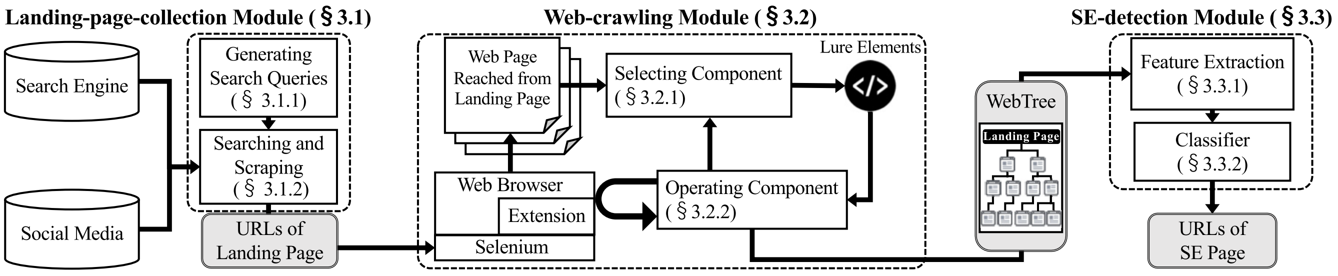

Landing-page-collection Module (§ 3.1)
Web-crawling Module (§ 3.2)
SE-detection Module (§ 3.3)
Search Engine
Social Media
Generating Search Queries (§ 3.1.1)
Searching and Scraping (§ 3.1.2)
URLs of Landing Page
Web Page Reached from Landing Page
Selecting Component (§ 3.2.1)
Lure Elements
Web Browser
Extension
Selenium
Operating Component (§ 3.2.2)
WebTree
Landing Page
Feature Extraction (§ 3.3.1)
Classifier (§ 3.3.2)
URLs of SE Page

Fig. 2 StraySheep overview.

such social-media postings are an instruction video for illegally installing software and a message introducing a free game download site. To collect such landing pages effectively, the landing-page-collection module uses a web-search-based approach consisting of two steps: *generating search queries* and *searching and scraping*.

3.1.1 Generating Search Queries

The landing-page-collection module generates search queries to search the URLs of possible landing pages leading to SE attacks. To generate the search queries, we design the module so that it collects *core keywords*, which stand for a title or name of paid content (e.g., “Godzilla” and “Microsoft Office”) and concatenates them with predefined *qualifiers* (e.g., “free download,” “crack,” and “stream online”). To collect core keywords, the module automatically scrapes popular electronic commerce (EC) sites and online database sites by using predefined scraping logic in accordance with each site and groups the core keywords by content category (e.g., video, software, and music). These core keywords can regularly be updated by recollecting ranking and new release information.

The aim of using qualifiers is (1) limiting the coverage of search results including illegal downloads and streaming, not legitimate sites, and (2) increasing the variation in search results. We manually prepare qualifiers in advance using autosuggest/related search functions on a search engine. When a user queries a certain word in a search engine, these search functions provide a list of corresponding keyword predictions. We input some titles of paid content to the search engine and collect qualifiers for each category because the qualifiers we require vary depending on the core keyword’s category. For example, qualifiers of video are “stream”, “movie”, and “online”. For another example, qualifiers of the software category are “download”, “crack”, and “key”.

3.1.2 Searching and Scraping

This module retrieves URLs from a search engine or social media by using the generated search queries. It inputs them into the search engine and search forms on social media to widely collect corresponding URLs. Some social media do not always provide comprehensive search results due to a minimum required search function; thus, the module also uses a search engine to collect social-media postings. Finally, it outputs the URLs collected from the search results and links scraped from social media postings as input for the web-crawling module.

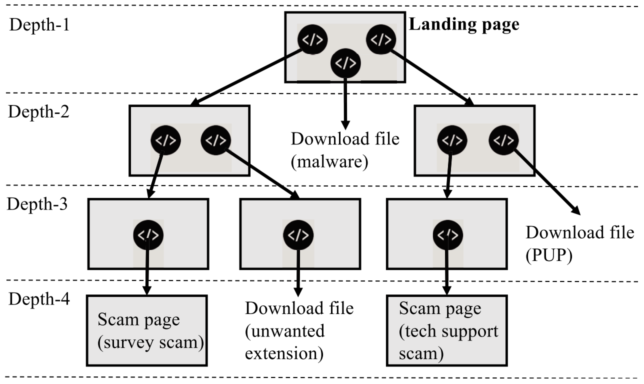


Fig. 3 Conceptual model of WebTree.

3.2 Web-Crawling Module

The web-crawling module automates a web browser to recursively crawl a URL collected by the landing-page-collection module and outputs a WebTree as a crawling result. Figure 3 shows a conceptual model of a WebTree representing sequences of web pages derived from the landing page and visited by the web-clawing module. The web-crawling module starts from the landing page, clicks on multiple lure elements on the web pages, and recursively follows multiple web pages derived from the landing page. The depth indicates the recursion count of web crawls. The depth increases when this module reaches a web page that completes loading and is waiting for browser interaction. This module uses Selenium and our original browser extension to automatically control and monitor a web browser. For the prototype of our system, we chose Google Chrome as a browser, but Selenium can also control other web browsers; thus, the web-crawling module can use different browsers. In the following section, we describe two components of the web-crawling module: *selecting* and *operating*.

3.2.1 Selecting Component

The selecting component collects a lure element that causes web navigation leading to SE attacks by analyzing an HTML source code and a screenshot of a web page. As mentioned in Sect. 2.1, a word representing the category or action of an element tends to be used for the lure element's DOM attributes, text content, and the text drawn inside the button graphic, for example, "download" in "`download-btn`" of the `class` attribute and "click" in "Click Now" of the text drawn inside a clickable button. To select elements containing such keywords as lure elements, we design the selecting component so that it parses an HTML source code and executes image processing of a web page's screenshot. The purpose of the selecting component is not to accurately detect elements leading to SE attacks but to select possible lure elements to reduce the number of elements with which to interact. By following only selected elements, the web-crawling module can efficiently reach diverse SE attacks. Note that there could be multiple lure elements on the same web page; thus, this component analyzes all elements on the web page. The reason the selecting component also executes image processing is to complement the acquisition of character strings drawn in the button image (i.e., `img` element), which cannot be acquired from the HTML source code. This component also identifies lure elements by their shape such as the triangular video play button.

We explain a statistical method of preparing keywords for selecting lure elements. We compare elements that have actually redirected users to SE attacks (lure elements) with other elements that have not redirected users to any SE attacks (*non lure elements*) and extract words specific to lure elements. More specifically, we extract attribute, text content, and strings drawn on buttons from the collected elements and divide these words into two documents: a document of lure elements and one of non lure elements. We then calculate the term frequency-inverse document frequency (tf-idf) of the two documents and manually choose words that have high tf-idf values from the lure-element document. The process of keyword selection is shown in Sect. 4.2.

In the analysis of HTML source codes, if an element matches at least one of the following four rules, this component determines it to be a lure element.

- One of the keywords is used in the element's text content.
- A keyword is set in `id`, `class`, or `alt` DOM attributes.
- A keyword is used as the file name indicated by the URL of the link (`a` element) or image (`img` element).
- An executable file (e.g., `.exe` or `.dmg`) or a compressed file (e.g., `.zip` or `.rar`) is used as a link extension.

The purpose of the analysis of image processing is to find rectangular buttons and video play buttons. This component extracts character strings written in each element from the screenshot and matches keywords used in the HTML source code analysis. This component leverages OpenCV to find rectangle contours representing the button areas in the screenshot and identify the coordinates and size of buttons. It also uses optical character recognition (OCR) using Tesseract OCR [25] to extract character strings from the rectangles the component found. This component executes keyword matching with extracted character strings and determines an element containing one of the keywords in the area to be a lure element. To acquire video play buttons as lure elements, the module also finds a triangle contour pointing right. Finally, the component outputs multiple lure elements that may lead to SE attacks from the web page.

3.2.2 Operating Component

The operating component executes browser interactions (i.e., clicking on lure elements), monitors web navigation, and constructs a WebTree. It simulates clicking on lure elements with the `CTRL` key pressed to open the web page in a new browser tab because the current page may be transferred to another web page by a simple clicking. As a result, links or popup windows can be opened in new tabs without changing the original tab. The operating module also clicks a `body` element, `body` element with context click, and the browser's back button to simulate unintended clicks described in Sect. 2.2. When the new tab is opened, the selecting component finds lure elements again, and the operating component executes browser interactions with a depth-first order, unless it reaches a predetermined maximum depth. We explain the maximum depth we used in the following experiment in Sect. 4.2.

The operating component also monitors web navigation. For monitoring JavaScript function calls, this component hooks the existing JavaScript function to detect the executed JavaScript function name and its argument. The JavaScript functions to be monitored by this component are `alert()`, `window.open()`, and the installation function of the browser extension (e.g., `chrome.webstore.install()`). The function `alert()` is frequently used in SE attacks that threaten a user by suddenly displaying dialog with messages inducing user anxiety. The `window.open()` function opens a new browser window and is used for popup advertisements. This component also hooks the installation function of the browser extension and detects what type of browser extension was installed from the argument. This component also monitors URL redirection, which navigates a user to another URL. URL redirection is divided into client-side redirection and server-side redirection. A web browser may conduct client-side redirection such as JavaScript function `location.href` when this component clicks the lure element. On the other hand, a web server conducts server-side redirection to navigate to another web page before loading a web page. This component monitors the URLs the browser passed during server-side redirection to identify the server that navigates users to SE attacks, such as advertising providers.

The operating component conducts browser interactions and monitors web navigation until it finishes clicking on all selected lure elements. This component aggregates information from sequences of web pages (i.e., screenshots, the HTML source codes of web pages, browser interactions, and web navigation) and finally outputs a WebTree as input for the SE-detection module.

3.3 SE-Detection Module

The SE-detection module extracts features from a WebTree output by the web-crawling module and identifies multi-step SE attacks using a classification model. This module first extracts *sequences* from the WebTree. A sequence is defined as a series of rendered web pages from the landing page (a root node) to the last pages (leaf nodes). Note that the sequence does not represent a URL redirection chain (an automatic process of forwarding a user to another URL multiple times) but a series of displayed web pages through user interaction. This module then extracts features from each sequence that reaches web pages of depth of two or more. Unlike conventional methods that examine structural similarity of URL redirection chains [9]–[11], this module extracts features specific to multi-step SE attacks from the entire sequence: contents of web pages, browser interactions that trigger page transitions, and web navigation. Finally, it identifies whether the last page of each sequence is the SE page using a classifier and outputs URLs of the detected web pages. Ground truth data for identifying SE attacks is explained in Sect. 4.3.

3.3.1 Feature Extraction

To classify web pages that trick users into interacting, it is common to use information that can be acquired after visiting the web page, such as image and HTML features [6], [8]. However, if a classifier uses such features, it cannot detect an SE page similar to the legitimate page, such as a fake software-update web page that closely resembles a legitimate Flash update page or fake infection-alert page using the logo of security vendors. Therefore, we designed feature vectors using not only features extracted from a single web page but also all features extracted from the entire sequence. Specifically, it analyzes the last page of the sequence, page before the last page (previous page), and the entire sequence, as shown in Fig. 4. Table 1 shows features extracted from each sequence and grouped into the three phases of SE attacks: user attraction, browser interaction, and web navigation. To the best of our knowledge, StraySheep is the first system that automatically collects these features from the entire sequence by recursively crawling web pages from the landing page. In terms of the user-attraction-based features, StraySheep extracts appearance, meaning of a document, and structure of HTML from the last and previous pages.

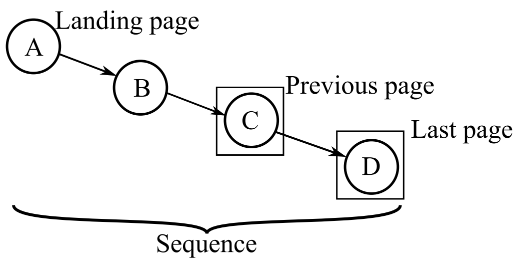


Fig. 4 Example of extracting features from a sequence.

Table 1 List of features SE-detection module uses.

	Target	Feature	# of dimensions
User attraction	Last page	Image features (Bag-of-visual-words)	128
	Last page	Color histogram	30
	Last page	Linguistic features (Doc2Vec)	300
	Last page	HTML tag histogram	40
	Last page	Length of text field	1
	Previous page	Image features (Bag-of-visual-words)	128
	Previous page	Color histogram	30
	Previous page	Linguistic features (Doc2Vec)	300
	Previous page	HTML tag histogram	40
	Previous page	Length of text field	1
Browser interaction	Sequence	# body clicks	1
	Sequence	# body context clicks	1
	Sequence	# left clicks	1
	Sequence	# back button clicks	1
	Sequence	# <a>tags clicked	1
	Sequence	# <iframe>tags clicked	1
	Previous page	Coordinates (x,y)	2
	Previous page	Size (width, height)	2
Web navigation	Sequence	Depth	1
	Sequence	# alert dialogues	1
	Sequence	# popup windows	1
	Sequence	# server side redirections	1
	Sequence	# client side redirections	1
	Last page	File downloads	1
	Last page	Extension installs	1
	Previous page	File downloads	1
	Previous page	Extension installs	1

It then finds features based on browser interaction, such as actions performed on the web pages and lure elements from the previous page and entire sequence. The SE-detection module also analyzes web navigation that occurred on the last page, the previous page, and the entire sequence. We explain a feature extraction method for each SE-attack phase below in detail.

User Attraction The appearance of a web page and the semantic properties of text content include the intention of the attacker to trick a user. The HTML document structure is also an important indicator for analyzing the similarity of web pages using the same document template. The SE-detection module extracts image and linguistic features from the last and previous pages of the sequence. It also calculates an HTML tag histogram, RGB color histogram, and the length of the text field from both the last and previous pages. These features are useful for identifying web pages that use the same page templates and images as other malicious web pages. To extract image features, we use AKAZE [26], which is a bag-of-visual words algorithm that detects local image features. The SE-detection module extracts 128-dimensional image features from the screenshots of the last and previous pages using a trained model we previously constructed. We use Doc2Vec [27] as a document-modelling algorithm to extract linguistic features. The purpose of this is to capture attackers' intentions, such as deceiving or threatening users, based on linguistic characteristics. The module extracts the 300-dimensional features from the text content of the last and previous pages by using a doc2vec model trained beforehand. The text content of a web-page document is extracted by cleaning out HTML tags from an HTML source code. The SE-detection module also calculates a histogram of the RGB (red, green, and blue) values of the screenshot with ten bins for each color and a histogram of HTML tags of the text content. This module uses up to 40 HTML tags (e.g., `a` `div`, and `img`) frequently appearing on the web pages we collected in advance. It counts the number of characters in the text content.

Browser Interaction The SE-detection module analyzes lure elements and actions that caused SE attacks. Browser interaction is an important indicator that characterizes multi-step SE attacks because the destination web pages change depending on the types of actions taken by users and clicked elements. To extract features from browser interactions, we design this module so that it counts the number of left clicks and unintended clicks (body clicks, body-context clicks, and back-button clicks) the web-crawling module performed in the sequence. This module also counts the types of clicked lure elements (`a` and `iframe`) in the sequence and determines the size (x,y) and coordinates (width, height) of lure elements on the previous page.

Web Navigation The SE-detection module analyzes browser events that occurred as a result of browser interaction. File downloads and extension install indicate events that are directly related to SE attacks such as malware downloads and unwanted extension installs. Since SE attacks are often delivered via advertising providers, redirection has characteristics unique to SE attacks. The method of navigation (e.g., redirection and popup window) is important for analyzing SE attacks. This module determines whether file downloads and extension installs occurred on the last and previous pages. It counts the times popup windows were displayed and the number of URLs observed during server-side and client-side redirection. It also checks the number of displayed alert dialogues and the length of the sequence, i.e., crawling depth.

3.3.2 Classifier

We combine the features extracted from sequences to create features vectors and construct a binary classifier to identify SE web pages. We use Random Forest as a learning algorithm because we can measure the importance of each feature that contributes to the classification. Evaluation results compared with other algorithms are given in Sect. 4.5.

4. Evaluation

We evaluated the three modules of StraySheep (landing-page-collection, web-crawling, and SE-detection). We first evaluated the qualitative advantage of StraySheep by comparing it with previous systems for collecting SE attacks. We then evaluated the effectiveness of the landing-page-collection module by comparing its two collection methods (search engine and social media) to three baseline URL-collection methods in terms of the number of landing pages leading to SE attacks and total visited malicious pages and domain names. Also, we conducted a crawling experiment to determine the efficiency of the web-crawling module by comparing its crawling method with two baseline crawling methods in terms of the number of malicious domain names reached per unit of time. Finally, we confirmed the effectiveness of the SE-detection module in terms of detection accuracy.

4.1 Qualitative Evaluation

We qualitatively compared StraySheep with the previous systems to collect SE attacks from five perspectives. Table 2 summarizes the results.

Collecting method. The previous systems [2], [3] for passively observing HTTP traffic to analyze SE attacks, can only collect attacks triggered by users' real download events. On the other hand, actively crawling arbitrary web pages with StraySheep enables us to proactively detect SE attacks before many users reach the web pages.

Interacting with elements. To observe multi-step SE attacks, we need to interact with HTML elements and recursively follow page transitions. Surveylance [6] is a system to detect survey gateways, which are landing pages displaying survey requests, and interact with their survey content and survey publisher sites. A system proposed by Rafique et al. [8] detects free live streaming (FLIS) pages and interacts with overlay video ads on them. While these sys-

Table 2 Comparison between proposed and previous systems.

	StraySheep	Surveylance [6]	Rafique et al. [8]	ROBOVIC [4]	Srinivasan et al. [5]	TrueClick [18]	WebWitness [2], [3]
Collecting method	Active	Active	Active	Active	Active	Active	Passive
Interacting with HTML elements	●	◐ (survey filling)	◐ (overlay video ad)	○	○	○	○
Following multiple lure elements on web page	●	○	○	○	○	○	○
Features (image)	●	●	●	○	○	●	○
Features (HTML)	●	●	●	○	○	○	○
Features (linguistic)	●	●	●	◐ (heuristic)	●	○	○
Features (sequence)	●	○	○	◐ (heuristic)	○	○	◐ (network level)
Source of landing-page collection	Search engine, **social media**	Search engine	Search engine	Parked domain, URL shortener	Search engine	File sharing site	Depending on users
Type of SE attacks to collect	**All multi-step SE attacks**	Survey scams	FLIS aggregator pages	Tech support scams	Tech support scams	Trick banners	SE downloads

●: Fully Covered, ◐: Partially Covered, ○: Not Covered

tems focus on survey scams or FLIS services, StraySheep can collect various SE attacks and observe different types of survey scams originating from web pages deeper than the landing pages (see Sect. 5.1).

Extracting features. As stated in Sect. 3.3, StraySheep extracts features such as images, HTML structures, and linguistic context from reached web pages and analyzes sequences to accurately detect multi-step SE attacks. As shown in Table 2, none of the previous systems use all the features used in StraySheep.

Source of landing-page collection. StraySheep collects landing pages from two common platforms: search engines and social media. StraySheep is the only system that uses both platforms.

Type of SE attacks to collect. While the previous systems are limited to detecting a specific attack, StraySheep collects various multi-step SE attacks by following lure elements on each web page.

In summary, StraySheep is the first system to collect multi-step SE attacks not limited to specific attacks by recursively following multiple lure elements on web pages. StraySheep also detects multi-step SE attacks by extracting various types of features from reached web pages and sequences.

4.2 Experimental Setup

We implemented StraySheep for Google Chrome 69 with Ubuntu 16.04. It simultaneously ran up to 32 instances on a virtual machine assigned with Intel Xeon 32 logical processors and 256-GB RAM. For the browser setting, a user agent was set as Google Chrome of Windows 7, and browser cookies were reset for every landing-page access. Our crawling experiment spanned from November to December 2018, and StraySheep used a single IP address. We need to set a timeout for performance evaluation because the two baseline web-crawling modules mentioned in Sect. 4.4 require an enormous amount of time (a few weeks at most) to complete web crawling. About 90% of web crawling conducted with StraySheep finished within an hour in our preliminary experiment (similar results are shown in Fig. 5); therefore, we set the timeout to one hour. To find the best maximum depth for collecting the most malicious domain names when we used the timeout, we changed the depth from two to six. The number of malicious domain names monotonically increased up to depth four and decreased as the depth increased. Therefore, we set the maximum depth to four in the following experiments.

To determine keywords for selecting lure elements, we followed the statistical method described in Sect. 3.2.1. First, we manually browsed landing pages (e.g., game download, movie streaming, and torrent sites) and clicked on various HTML elements. We also browsed intermediate pages navigated from them, such as fake virus alerts, file downloading, and advertising pages served by URL shorteners. We then gathered 1,447 lure elements from 978 web pages, which we confirmed finally led to SE attacks. To determine if the reached web pages contained SE attacks, we used URL/domain blacklists (Google Safe Browsing, Symantec DeepSight [28], and hpHosts [29]) to match visited web pages and checked the MD5 hash values of the downloaded binaries with VirusTotal. We defined an *SE page*, which matched the blacklist whose label was associated with SE attacks (e.g., phishing, tech support scam, and survey scam) or started downloading malware or potentially unwanted programs (PUPs) [30], [31]. We used the same method of checking SE pages in the following experiments. We randomly selected 5,000 non-lure elements that did not redirect to any SE pages from the landing and intermediate pages. We created lure and non-lure elements' documents containing words extracted from attributes and text content to calculate tf-idf. Finally, we chose 31 keywords specific to the lure elements by excluding proper nouns (e.g., game and movie titles) and words with zero tf-idf values.

4.3 Effectiveness of URL Collection

To show the effectiveness of StraySheep's landing-page-collection module, we validated landing pages collected by this module; thus, we used the web-crawling module to recursively crawl the landing pages and identified whether visited web pages caused SE attacks. We compared the number of collected landing pages that led to SE attacks across the five methods, i.e., the landing-page-collection module's two methods (search engine and social media) and three baseline methods (Alexa top sites, trend words, and core keywords). We collected 5k landing pages for each method.

Search Engine (StraySheep's Method) This method collected a total of 3k core keywords from EC/database sites, such as amazon.com, steampowered.com, billboard.com,

Table 3 Results of web crawling starting from landing page collected with each method.

	Search Engine (StraySheep)	Social Media (StraySheep)	Alexa Top Sites (Baseline)	Trend Words (Baseline)	Core Keywords (Baseline)
# of landing pages	5,000	5,000	5,000	5,000	5,000
# of landing pages lead to SE attacks	**1,060 (21.2%)**	**808 (16.2%)**	33 (0.7%)	65 (1.3%)	46 (0.9%)
# of unique visited URLs	50,587	25,722	27,818	16,240	30,133
# of unique URLs of visited SE pages	**4,716 (9.3%)**	**1,633 (6.3%)**	80 (0.3%)	105 (0.6%)	628 (2.1%)
# of unique visited domains	4,984	3,619	8,046	3,537	4,507
# of unique domains of visited SE pages	**446 (8.9%)**	**151 (4.2%)**	42 (0.5%)	27 (0.6%)	95 (2.1%)
# of unique downloaded Malware samples	160	186	50	3	41

and imdb.com, which we chose from Alexa top 500 sites. These core keywords were divided into five categories: software (game and applications), video (movie, animation, and TV series), music, eBook, and comic. We can increase the variety of landing pages by collecting different types of core keywords, which are often used in illegal sites to lure users. This method generated 90k search queries by concatenating the core keywords with an average of 30 predefined qualifiers for each category. When we search for only core keywords, many legitimate sites, such as official sites of movies or games, are included in the search results. However, we can collect more landings pages leading to SE attacks, including illegal sites, by adding qualifiers to core keywords. It searched the queries using Microsoft Bing Web Search API [32] (Bing API) and collected about 1M unique URLs. In that web search, it gathered URLs from up to 30 search results for each search query. Note that the search queries containing the same core keywords with different qualifiers sometimes returned duplicate search results, and some search queries returned less than 30 search results. Finally, we randomly sampled 5k URLs from the collected 1M URLs to crawl for the crawling experiment.

Social Media (StraySheep's Method) This method also searched seven social-media platforms (Facebook, Twitter, Youtube, Dailymotion, Vimeo, Flickr, and GoogleMap) using the same search queries as the above search-engine experiment. Attackers post fake messages on social media such as free downloads of games and streaming of movies to lure users into accessing their links. By collecting such social media posts, we can also gather landing pages that do not appear in search engine results. This method extracted links from posting messages (from Facebook, Twitter, and Flickr), descriptions of uploaded video (from Youtube, Dailymotion, and Vimeo), and descriptions of GoogleMap's My Maps. It used search forms on Youtube, Dailymotion, and Facebook because they have flexible search mechanisms and searched Bing API for the other social-media platforms to gather up to 30 social media postings for each search query. It searched for 10k search queries (sampled from 90k search queries) for each social-media platform and found a total of 130k unique social-media postings. These search queries often returned less than 30 search queries. This method then gathered 45k unique links by scraping these 130k social-media postings. Some social-media postings did not include any links or included multiple links. Finally, we randomly sampled 5k URLs from the 45k links for the crawling experiment.

Alexa Top Sites (Baseline Method) We gathered the top 5k domain names from Alexa top sites and converted them to 5k URLs by adding "`http://`" to the domain names.

Trend Words (Baseline Method) We searched the top 1k trend words collected from Google Trends using Bing API and randomly selected 5k URLs from the 30k search results (retrieved 30 results per query).

Core Keywords (Baseline Method) We simply searched the same set of 3k core keywords we used for the above Search Engine method and randomly sampled 5k URLs from the 90k search results (retrieved 30 results per query).

Table 3 lists the results of web crawling for each method. The landing pages that led to SE attacks and collected with the search-engine and social-media methods accounted for 21.2 and 16.2% for each 5k landing pages. While, those of the three baseline methods (Alexa top sites, trend words, and core keywords) were much smaller, 0.7, 1.3, and 0.9%, respectively. From the results of the search-engine and social-media methods, the numbers of unique visited URLs and domain names were larger than those of the three baseline methods. Since StraySheep's methods, which use qualifiers, collected about 20 times as many landing pages lead to SE attacks as the baseline method (Core Keywords) when using the same set of core keywords, qualifiers are effective in collecting landing pages. The number of malware samples reached from the URLs collected with the search-engine and social-media methods was also larger than that of the other three methods.

4.4 Efficiency of Web Crawling

To evaluate the efficiency of StraySheep's web-crawling module, especially the function to follow lure elements selected by the selecting component, we compared the ratio of SE pages in visited web pages and the time to reach SE attacks among three web-crawling modules: that of StraySheep's web-crawling module and two baseline web-crawling modules. Then, we compared the crawling performance of StraySheep with that of TrueClick [18].

Comparison of crawling performance with baseline web-crawling modules and StraySheep We implemented the two baseline modules: *ElementCrawler*, which extracts all visible elements on the web pages and simply clicks them, and *LinkCrawler*, which purely selects all the link elements (HTML `a` tag with `href` attribute) and clicks them. Note that elements selected by ElementCrawler contain all those selected by LinkCrawler or StraySheep's web-

Table 4 Results for each web-crawling module.

	StraySheep		ElementCrawler		LinkCrawler	
	SE pages	Total	SE pages	Total	SE pages	Total
# of Total pages	9,374 (**5.4%**)	173,060	13,559 (2.4%)	562,708	19,241 (3.6%)	540,822
# of Unique visited pages	**6,283 (8.5%)**	73,906	5,998 (3.1%)	191,901	5,445 (3.0%)	180,920
# of Unique visited domains	**513 (6.7%)**	7,660	437 (3.2%)	13,545	335 (3.4%)	9,734

crawling module. ElementCrawler and LinkCrawler are alternative implementations of StraySheep's web-crawling module, which are implemented by replacing the selecting component (see Sect. 3.2.1) with the function of selecting all elements or all links from an HTML source code. The landing pages we input to the three modules were the same 10k URLs as those collected by the landing-page-collection module, as mentioned in Sect. 4.3, which are the 5k URLs collected from a search engine and another 5k URLs collected from social media. We newly crawled the 10k landing pages using ElementCrawler and LinkCrawler under the same condition mentioned in Sect. 4.3. We compared these crawling results with those of the above experiment in which StraySheep's web-crawling module crawled the 10k landing pages. In the same manner as the above experiment, we identified SE pages using blacklists and VirusTotal.

Table 4 shows the number of total pages, unique visited pages, and domain names for each web-crawling module. The numbers of unique visited pages and domain names of SE pages visited with StraySheep's web-crawling module were 6,283 pages and 513 domain names, which were larger than those of the baseline modules, and the percentages of pages and domain names of SE pages were also larger than those of the baseline modules (8.5 and 6.7%, respectively). Although the numbers of total pages of ElementCrawler and LinkCrawler were three times larger than that with StraySheep's web-crawling module, StraySheep's web-crawling module had the best percentage (5.4%) for all SE pages. This is because StraySheep's web-crawling module selected lure elements from thousands of elements to crawl web pages likely to cause SE attacks, while ElementCrawler and LinkCrawler simply took turns to click elements and reached many benign web pages. In short, ElementCrawler may crawl all potential SE attacks by taking an enormous amount of time; however, StraySheep can reach SE attacks in a shorter time by selecting lure elements.

Next, we analyzed the efficiency of each web-crawling module by comparing the time taken to complete visiting web pages branching from the landing page. Figure 5 is a cumulative distribution function (CDF) of the time for each web-crawling module, which shows the percentage of web crawling finished at a certain time out of all web crawling starting from 10k landing pages. We found that 88.5% of StraySheep's web crawling module finished within one-hour timeout. In contrast, ElementCrawler finished only 22.8% of web crawling within the timeout, and LinkCrawler finished 29.9%. The average time to complete the web crawling for each landing page was 14 minutes for StraySheep's web-crawling module, 49 minutes for ElementCrawler, and 47 minutes for LinkCrawler.

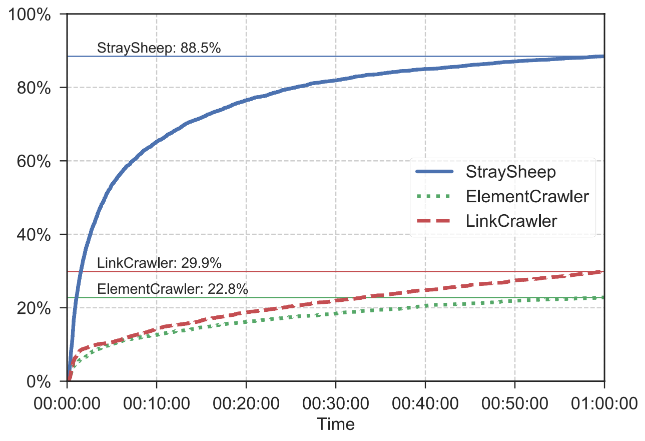


Fig. 5 CDF of time taken to complete web crawling for each landing page within a 1-hour timeout. Horizontal lines mean the percentage of web crawling completed before timeout.

To measure the web-crawling modules' ability to reach SE attacks per total crawling time, we calculated *crawling efficiency*.

$$\text{Crawling Efficiency [/sec]} = \frac{\text{\# Unique domains of visited SE pages}}{\text{Total crawling time [sec]}}.$$

Crawling efficiency indicates the ability to reach the unique domain names of SE pages per unit of time. Higher crawling efficiency implies that the module can efficiently reach new SE pages.

We show the crawling efficiency for each web-crawling module in Table 5. Total crawling time in Table 5 represents the sum of the times to complete crawling 10k landing pages. The crawling efficiency of StraySheep's web-crawling module was 4.1 times higher than that of ElementCrawler and 5.1 times higher than that of LinkCrawler, making it the most efficient module to reach SE attacks. As described in Sect. 3.2.1, since StraySheep's web-crawling module detected lure elements that led to SE pages by using the selecting component, it visited more SE pages in less time than the two baseline modules.

We also examined the ability to visit SE attacks that can be reached via multiple web pages. Table 6 shows the number of unique domain names observed at each depth. Note that each depth may have duplicate domains because the web-crawling modules visited the same domains at different depths. Also, the number of domain names observed at a depth of 1 was the same because each module visited the same landing pages. The number of domains of SE pages show that StraySheep's web-crawling module efficiently

Table 5 Crawling efficiency of each web-crawling module.

	StraySheep	ElementCrawler	LinkCrawler
# of unique domains of visited SE pages	**513**	437	335
Total crawling time [sec]	**8,429,288**	29,698,118	28,421,460
Crawling efficiency [/sec]	$\mathbf{6.1 \cdot 10^{-5}}$	$1.5 \cdot 10^{-5}$	$1.2 \cdot 10^{-5}$

Table 6 Unique domain names observed at each depth.

Depth	StraySheep		ElementCrawler		LinkCrawler	
	SE	Total	SE	Total	SE	Total
1	91 (2.2%)	4,187	91(2.2%)	4,187	91(2.2%)	4,187
2	**223 (5.5%)**	4,043	159 (4.1%)	3,882	126 (4.6%)	2,726
3	**231 (6.3%)**	3,692	171 (3.5%)	4,895	148 (3.9%)	3,844
4	**348 (9.4%)**	3,685	299 (2.8%)	10,694	303 (3.4%)	8,939

visited more domains of SE attacks at every depth than the baseline modules. As the depth became deeper, the percentages of an SE page's domains that ElementCrawler and LinkCrawler detected decreased. On the contrary, the percentages of an SE page's domains that StraySheep's web-crawling module visited were 5.5% at a depth of 2, 6.3% at a depth of 3, and 9.4% at a depth of 4; thus, the deeper StraySheep's web-crawling module crawled, the more it efficiently visited SE pages. As described in Sect. 3.2.1, StraySheep selects lure elements that lead to SE attacks so that the web-crawling module can reach more of an SE page's domains even though it crawls deeper.

Comparison of crawling performance with TrueClick and StraySheep We also conducted an additional experiment comparing the crawling performance of StraySheep with that of TrueClick in terms of the ability to reach SE pages and collect malware executables. TrueClick is a tool that distinguishes fake advertisement banners (trick banners) from genuine download links. TrueClick has the similar purpose as StraySheep for finding HTML elements that are made to deceive users and direct to a malicious site or malware executable, but it only finds elements displayed by advertising providers regardless of the web site owner's intention.

Since the source code of TrueClick has not been published, we re-implemented TrueClick based on the implementation details of the paper [18] using a manually collected dataset containing 87 trick banners and 51 genuine banners, which is almost equivalent to the amount of the original dataset (165 trick banners and 94 genuine download links), to train a machine learning model. The trained model identifies trick banners with 98.6% accuracy. We then created a baseline crawling module by replacing StraySheep's selecting component (Sect. 3.2.1) with TrueClick implementation.

To equivalently compare the crawling results under the same experimental condition in terms of the period of landing-page collection and web crawling, we have collected 5k URLs in the same manner as that mentioned in Sect. 4.4 and crawled them using both StraySheep's web-crawling module and the baseline module as of November 2019. Since this experiment was conducted at a different period than the one explained above, we newly collected 2.5k landing pages each from a search engine and social media as input URLs. Table 7 summarizes the results. StraySheep visited more SE pages than TrueClick because it follows not only trick banners but also buttons and links intentionally placed by web site owners to lead to SE attacks. While StraySheep successfully downloaded 266 malware samples, TrueClick downloaded only 1 malware sample. This is because, in most cases, genuine download links distribute malware samples instead of trick banners on web pages redirected from the first trick banners on landing pages. Table 8 shows the number of unique SE pages observed at each depth. Similar to the results in Table 6, Straysheep reached more SE attacks as it crawled deeper. Conversely, the number of SE pages that TrueClick reached considerably decreased deeper than depth three. The reason for this is that as we crawl deeper from the landing page, the number of trick banners decreases. Additionally, intentionally placed lure elements including genuine download links mainly lead to SE attacks at deeper depths. Figure 6 shows the overlap of SE pages' domain names observed using each crawler. Although StraySheep did not visit a small number of SE pages dynamically served by ads, it covered most of the SE pages observed by TrueClick. In summary, to collect more multi-step SE attacks, we need not only to detect trick banners but also follow lure elements.

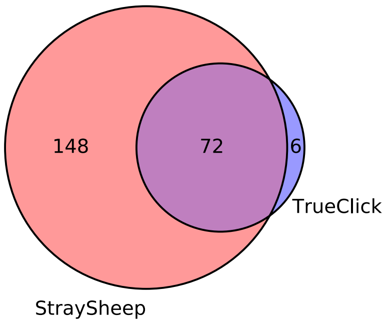


Fig. 6 Overlap of SE pages' domain names observed using StraySheep and TrueClick.

4.5 Evaluating the SE Detection Module

We evaluated the effectiveness of StraySheep's SE-detection module using WebTrees, which are the outputs of StraySheep's web-crawling module. We used 30k WebTrees constructed from the results of web crawling starting from 30k landing pages. These WebTrees consisted of the 10k landing pages crawled by StraySheep's web-crawling module (Sect. 4.4) and additional 20k landing pages. The

Table 7 Results of web crawling using StraySheep and TrueClick.

	Unique visited pages (domain names)	Unique visited SE pages (domain names)	Unique malware samples
StraySheep	48,524 (5,809)	3,897 (219)	266
TrueClick	7,917 (2,978)	523 (78)	1

Table 8 Unique SE pages observed at each depth by using StraySheep and TrueClick.

depth	SE pages crawled using StraySheep (domain names)	SE pages crawled by TrueClick (domain names)
1	97 (44)	97 (44)
2	845 (86)	356 (35)
3	1068 (104)	48 (12)
4	2302 (106)	25 (12)
Unique SE pages	3,897 (219)	523 (78)

20k landing pages were collected and randomly sampled in the same manner as for the 10k landing pages mentioned in Sect. 4.3. We carried out the web crawling in the same environment in the same period to output additional 20k WebTrees.

To create datasets for evaluation, we extracted malicious and benign sequences from the 30k WebTrees. The 30k WebTrees contained a total of 243,914 unique web pages (13,415 unique domains) of visited web pages. To label these web pages as SE pages, we used blacklists (same as in Sects. 4.3 and 4.4) and VirusTotal. We labeled 51,501 unique web pages (unique 1,066 domains) as SE pages and extracted 1,066 sequences, which reached 1,066 different domain names from distinct landing pages. We excluded unreachable or parking domain pages and created 1,045 sequences as the malicious dataset. To create a benign dataset, we randomly sampled 1,045 sequences that did not visit SE pages.

To evaluate the detection accuracy of the SE-detection module, we conducted a 10-fold cross-validation (CV) on the labeled dataset. The SE-detection module classified our dataset with a precision of 97.4%, recall of 93.5%, and accuracy of 95.5%. When we changed the learning algorithm from random forest to support vector machine, logistic regression, and decision tree, their accuracies were 93.6%, 90.8%, and 90.7%, respectively. The percentage of feature importance accounted for 65.2% of features extracted from the last page (*last page features*), 28.2% of features extracted from the previous page (*previous page features*), and 6.6% of features extracted from the entire sequence (*sequence features*), as shown in Table 1.

Although the SE-detection module can accurately identify multi-step SE attacks, the evaluation result contained some false positives and false negatives. We discuss ideas for reducing these false positives and false negatives. The false positives included popular shopping and casino sites that were redirected from pop-up ads triggered by unintended click. We can reduce these false positives by extracting long-term stable and popular domain names from domain lists such as Alexa to create a white list. We also found false negatives that were listed on blacklists but not

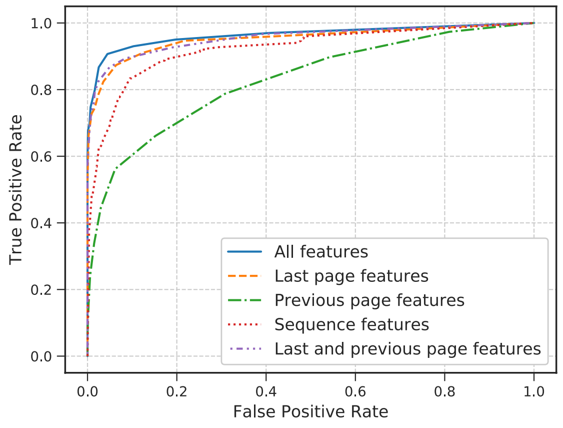


Fig. 7 ROC curves of SE detection results for each feature set.

detected by the SE-detection module. Since we only trained web pages written in English in this experiment, Some web pages written in non-English languages were included in false negatives. Ad providers may change web pages to serve depending on the region of a source IP address. Therefore, we can accurately detect multi-step SE attacks by automatically translating web pages to English or by training web pages written in a specific language corresponding to the region of the source IP address.

To show the relationship between detection accuracy and features, we divided the features into four feature sets: last page, previous page, sequence, and the combination of last and previous page (feature sets without our proposed sequence features). We conducted 10-fold CVs using all feature sets and four divided feature sets with the same dataset discussed in Sect. 4.5. Figure 7 shows the receiver operating characteristic (ROC) curves for the classification results. The most accurate result was the CV using all features in order of the combination of last and previous page, last page, sequence, and previous page feature sets. The area under the curve (AUC) for each result was 0.965, 0.955, 0.948, 0.923, and 0.829. This experiment revealed that our original page-level features that analyzed linguistic, image, and HTML characteristics were useful in detecting various types of SE attacks, i.e., not limited to a specific SE attack. However, we can classify more accurately by using the features of a previous page and sequence together that StraySheep automatically collects.

Some web pages had similar appearances to known SE pages but were not blacklisted. To find such potentially unknown SE pages, we leveraged the SE-detection module to classify the remaining 192,620 sequences of the 11,304 domain names not used in the evaluation. As a result of manually excluding false positives (27 domains) from the classification results, we found 359 unknown domain names associated with SE attacks. We not only detected web pages where page contents were shared across multiple domain names to expand attack campaigns (e.g., Fig. 8 and Fig. 9), but also discovered unreported domain names associated with technical support scams and survey scams. This process was conducted by analyzing screenshots to check whether suggested software and extensions or login pages are associated with legitimate services. One example of the false positives was a Facebook login page opened by a popup that redirected from an illegal software-download blog by clicking a share button. Another example was a download page of legitimate anti-virus products that transferred by clicking advertising in an iframe. We finally found a total of 1,404 unique domain names (the 1,045 blacklisted domain names and newly detected 359 domain names), and 56,922 sequences reached the 1,404 domain names. The number of sequences' steps (i.e., the number of page transitions) from one to three is 11,855 (20.8%), 13,813 (24.3%), and 31,254 (54.9%), respectively.

5. Detailed Analysis of Detected Multi-Step SE Attacks

We conducted a detailed analysis of the collected multi-step SE attacks mentioned in Sect. 4.5 (1,404 domain names and 56,992 sequences). To show that StraySheep found a wide variety of SE attacks, we categorized the observed SE page's domain names and investigated the attacker techniques to deceive and persuade users for each SE attack category. We then analyzed the browser interactions and advertising providers that led to SE pages to clarify the cause of SE attacks. Finally, we investigated network infrastructures hosting SE attacks.

5.1 SE Attack Categories

To clarify the types of multi-step SE attacks detected by StraySheep, we categorized the 1,404 domain names into 11 categories, as shown in Table 9. We used labels of blacklists (Google Safe Browsing, Symantec DeepSight, hpHosts) and virus scan results of VirusTotal to categorize the attacks. We leveraged AVClass [33] to classify detected binaries as PUPs or malware. We also checked the appearance of these domain names' web pages to complement categorization.

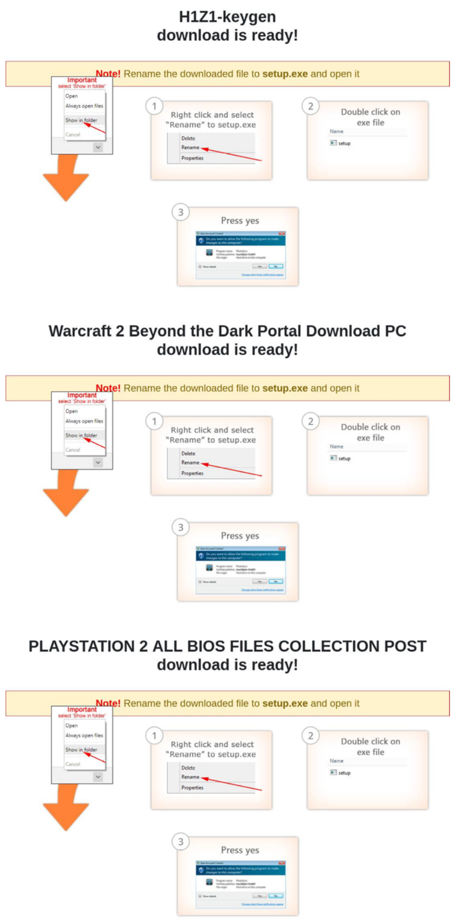
PUP and Malware The most common categories we identified were PUP (566 domain names) and malware (310 domain names). These categories are SE attacks where PUPs and malware were downloaded due to browser interactions. StraySheep downloaded 6,924 unique binary executable files (e.g., `.exe` or `.dmg`). For example, we found that these binaries were disguised as fake game installers, fake anti-virus software, and fake Java/Flash updaters. Out of the 6,924 binaries, we detected 1,591 unique binaries including 1,090 malware samples and 501 PUPs by checking their MD5 hash in VirusTotal and using AVClass. We confirmed that 3,336 unique binaries were never uploaded to VirusTotal. Although the remaining 1,997 unique binaries were already uploaded, they were not detected by any anti-virus software in VirusTotal.

Table 9 SE attack categories.

Category	SE domain names
PUP	566 (40.2%)
Malware	310 (22.1%)
Unwanted browser extension	181 (12.9%)
Multimedia scam	94 (6.7%)
Phishing	70 (5.0%)
Survey scam	25 (1.8%)
Tech support scam	20 (1.4%)
Fake browser history injection	16 (1.1%)
Malvertisement redirection	13 (0.9%)
Cryptojacking	3 (0.2%)
Other SE attacks	109 (7.8%)
Total	1,404 (100%)

The 2,141 out of the 3,336 binaries that were not uploaded had 1,347 unique filenames, which were automatically set according to the previous page (e.g., "[*the title of the previous page*]`.exe.rename`"). Figure 8 shows examples of these web pages. The web pages that downloaded these binaries contained instructions to entice users to remove "`.rename`" and execute them. We found 504 unique domain names downloading these binaries. The 175 out of these 504 domain names matched the blacklists and the other 329 domain names were newly detected by StraySheep. The reason for making users change the file extension is to circumvent the download-protection function of web browsers. Since the hash values of these binaries also changed at every downloading, none were ever uploaded to VirusTotal. To check whether these binaries were malicious, we chose ten samples from the binaries and uploaded them to VirusTotal. Then, all ten samples were detected as "StartSurf" or "Prepscram" family names.

Unwanted Browser Extension We categorized 181 domain names as distributing unwanted browser extensions. We confirmed that these domain names were detected as "Fake Browser Extension Download" or "Unwanted Extension", which led to install pages (`https://chrome.google.com/webstore`) of 128 unique Google Chrome browser extensions. However, we found that 119 (93.0%) extensions were still available on the browser extension install pages a month after the crawling. By investigating these browser extensions, we found that 18 (14.1%) extensions were search tool bars, and 14 (10.9%) extensions were file converters. Security vendor blog postings and on-


H1Z1-keygen
download is ready!

Note! Rename the downloaded file to setup.exe and open it

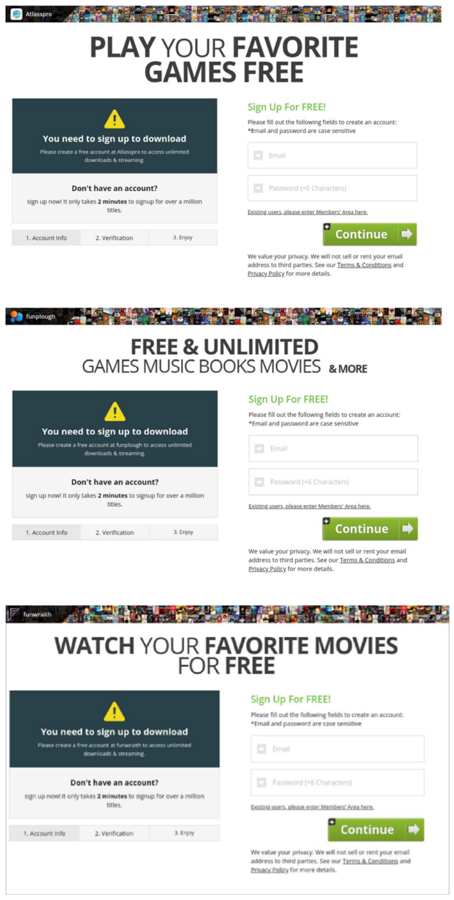
Warcraft 2 Beyond the Dark Portal Download PC
download is ready!

Note! Rename the downloaded file to setup.exe and open it

PLAYSTATION 2 ALL BIOS FILES COLLECTION POST
download is ready!

Note! Rename the downloaded file to setup.exe and open it

Fig. 8 Examples malware-distribution pages that require user to rename files and execute them.


PLAY YOUR FAVORITE GAMES FREE

You need to sign up to download
Please create a free account at Atlasspro to access unlimited downloads & streaming.

Don't have an account?
sign up now! It only takes 2 minutes to signup for over a million titles.

Sign Up For FREE!
Please fill out the following fields to create an account:
*Email and password are case sensitive

Existing users, please enter Members' Area here.

We value your privacy. We will not sell or rent your email address to third parties. See our Terms & Conditions and Privacy Policy for more details.

FREE & UNLIMITED
GAMES MUSIC BOOKS MOVIES & MORE

You need to sign up to download
Please create a free account at funplough to access unlimited downloads & streaming.

WATCH YOUR FAVORITE MOVIES FOR FREE

You need to sign up to download
Please create a free account at furwraith to access unlimited downloads & streaming.

Fig. 9 Examples of multimedia scams.

line forums stated that some extensions were malicious extensions or browser hijackers that modify web browser settings, track user's browsing, and inject unwanted advertisements [34]. As a result of our dynamic analysis of some browser extensions using a real browser, we observed suspicious behavior such as displaying popup advertisements and changing the default browser's homepage and search engine to web pages hard coded in many malware samples. To determine the popularity of these browser extensions, we searched each extension name on a search engine. We then found that the search results of 100 extensions (78.1%) consisted of one or more web pages explaining "How to remove [*browser extension name*]" or "Virus removal guide". Surprisingly, most of these web pages introduced not only removal methods but also suggested yet more fake removal tools, which were detected as PUPs or malware. Attackers prepared the web pages for tricking technically unsophisticated users who disrupt these browser extensions. Thus, even if the users successfully remove the unwanted browser extensions, they also become victims of other SE attacks. StraySheep's SE-detection module newly found 21 domain names out of the 181 domain names we categorized. StraySheep successfully finds unwanted browser extensions by analyzing distribution web pages and sequences that led to them instead of analyzing their source codes and behaviors.

Multimedia Scam We found web pages (94 domain names) that ask for credit card registration in exchange for offering free access to movies or music. Their content, such as input forms, logos, and background images, were shared among each other. We call them *multimedia scams* in this paper. Only 27.7% (26/94) of domain names were listed in blacklists; however, StraySheep's SE-detection module newly found 72.3% (68/94) domain names. Some security vendor blog postings and online forums reported that these web pages fraudulently charge credit cards [35]. We found that some words (e.g., `media`, `play`, and `book`) were frequently used in the domain names, such as `etnamedia.net`, `kelpmedia.com`, `dewymedia.com`, `parryplay.com`, `cnidaplay.com`, and `mossyplay.com`.

Figure 9 shows examples of multimedia scams. These web pages suggest users to register for free membership to obtain movies, music, or games. When users are tricked to input their credit card numbers, the web pages fraudulently charge them.

Phishing We observed 94 domain names detected as *phishing*, which were attempting to steal user's sensitive information such as email addresses or passwords.

Survey Scam We found 25 *survey scam* domain names, which spoofed famous companies and promised rewards such as iPhones and gift cards. Although Surveylance [6] only identified landing pages that have survey content and interacted with them to reach survey scams, StraySheep recursively followed lure elements to detect survey scams reached from landing pages that did not have survey content.

Tech Support Scam We observed 20 *tech support scam* domain names that displayed fake virus-infection messages and telephone numbers of support centers to urge users to call. StraySheep reached the scams from sequences of web pages starting from the search engine's results and social media postings, which are not observed with other systems [4], [5].

Fake Browser History Injection We found 16 *Fake browser history injection attacks* domain names, which injected URLs into the browser's history to force users to redirect to another SE page when the browser's back button is clicked. To interact with such attacks, StraySheep attempted clicking the back button for each web page and determined that the action led to other SE pages.

Malvertisement Redirection We found 13 *Malvertisement website redirect* domains [36], [37] that also led users to other SE pages.

Cryptojacking We found three *Cryptojacking* domain names that secretly used user's CPU resources to mine cryptocurrencies by injecting JavaScript codes.

Other SE Attacks We observed various SE attacks other than those mentioned above, such as one just indicates the "Social engineering" label.

5.2 Common Infrastructures of Multi-Step SE Attacks

To clarify the common infrastructures of multi-step SE attacks and attacker's techniques leading to the attacks, we analyzed the 56,922 sequences (see Sect. 5) that led to SE pages.

SE Attacks Caused by Unintended Clicks We observed opening popup/popunder windows caused by unintended clicks such as clicking anywhere on a web page and on the browser's back button. Such popups are often set by JavaScript codes provided by advertising providers to the web page's owner. The following three files are the most frequently loaded on web pages leading users to SE pages: "`c1.popads[.]net/pop.js`", "`cdn.popcash[.]net/pop.js`", and "`cdn.cpmstar[.]com/cached/jspopunder_v101.pack.js`". Since such advertisements are common infrastructures for SE attack distribution, they are used in various web pages. The sequences in which popups caused by unintended clicks occurred were 20.0% (11,373/56,922) of all sequences. The sequences in which popups caused by unintended clicks occurred in the landing pages were 8.7% (4,952/56,922) of all sequences. We also observed exit-driven redirections that were triggered by clicking on the browser's back button, which was 4.5% (2,578/56,922) of all sequences.

Alert Dialog Of all sequences, 2.9% (1,651/56,922) included a web page that displayed more than one alert dialog. We found 66 distinct alert messages, such as those of fake virus infection and fake rewards, which might strongly influence user psychology. To investigate the relationship between the content of alert messages and SE attacks, we categorized the 66 alert messages into the three attack classes of *comply*, *alarm*, and *entice*. These classes were defined in a previous study [3]. We found 30 *Comply* alerts that were often used on fake Java/Flash update web pages for luring users to install PUPs and malware, such as "Please install Java to continue." and "Your Flash Player might be out of date. Please install update to continue." We found 19 *Entice* alerts that made users input sensitive information, such as "CONGRATULATIONS! Your IP address has been selected to receive a Year of FREE Netflix!" We found 17 *Alarm* alerts that showed warning messages such as "IMMEDIATE ACTION REQUIRED We have detected a trojan virus" with alert sounds in some cases (e.g., `<audio src=''alert.mp3'' autoplay>`). Users were directed to install fake anti-virus software or call fake technical support centers.

Advertising Domain Names Online advertising often results in SE attacks [3], [6]. To analyze SE attacks delivered by advertising providers, we extracted advertising providers' domain names (ad domain) from server-side redirection on the sequences. We leveraged public advertising provider lists [38] to identify ad domains. Table 11 shows a list of ad domains and the number of unique domain names of SE pages redirected from each ad domain. We found 25 ad domains that led to SE attacks. Categories of SE attacks frequently distributed by ad domains were multimedia scam, unwanted browser extension, fake anti-virus software (PUP/malware category), and fake Java update (PUP/malware category). The ad domain that redirected to the most SE page's domain names was `newstarads.com`, which led to 155 unique domain names. Two domain names (`doubleclick.net` and `googleadservices.com`) redirected to 89 and 79 unique domain names of unwanted browser extension and they also redirected to the same phishing domain names. We found that 30.4% (427/1,404) of the total SE domain names were reached from these advertising domain names.

Prevalence of SE attacks We analyzed the statistics of user accesses to measure how many users encountered multi-step SE attacks. We used SimilarWeb[†], Alexa Web Information Service (AWIS)[††], and DNSDB[†††] to investigate website traffic volumes of 1,404 domain names that StraySheep collected, as mentioned in Sect. 4. SimilarWeb and AWIS provide website traffic statistics of domain names. DNSDB is a passive DNS database that provides the total number of DNS queries of domain names. Table 10 lists the numbers of unique domain names newly observed at each depth in ascending order and statistics (minimum, maximum, sum,

[†]https://www.similarweb.com/
[††]https://awis.alexa.com/
[†††]https://www.dnsdb.info/

Table 10 Newly observed SE pages' domain names at each depth in ascending order and their user access investigated by SimilarWeb, Alexa Web Information Service, and DNSDB.

Depth	# of domains	SimilarWeb (total visits per month)					Alexa Web Information Service (pageviews per million users)					DNSDB (DNS queries)				
		# of domain names with valid data	Min	Max	Sum	Mean	# of domain names with valid data	Min	Max	Sum	Mean	# of domain names with valid data	Min	Max	Sum	Mean
1	94	39	16,616	110,300,000	184,891,612	4,740,811	68	0.01	41.70	143.49	2.11	83	1	30,896,251	58,835,762	708,865
2	680	261	9,926	205,800,000	1,483,161,357	5,682,611	344	0.01	631.50	1684.62	4.90	425	2	19,727,061	52,593,168	123,749
3	288	108	16,326	92,460,000	301,856,918	2,794,971	144	0.01	264.00	476.95	3.31	181	1	61,561,400	302,969,603	1,673,865
4	342	28	15,191	115,400,000	158,933,466	5,676,195	51	0.01	8.62	34.32	0.67	93	2	11,357,814	63,461,471	682,381
Total	1,404	436	9,926	205,800,000	2,128,843,353	4,882,668	607	0.01	631.50	2339.38	3.85	782	1	61,561,400	477,860,004	611,074

mean) of website traffic and DNS queries. Note that *# domain names with valid data* means the number of domain names excluding the data that are zero or not available in the data sources. Since most SE pages' domain names were observed at depth two, there were still 44.9% (630) of domain names observed at deeper depths. In other words, there are many domain names at deeper depths that can only be reached by following multiple web pages with StraySheep. The statistics of user accesses and DNS queries show that these websites have the same level of population with domain names observed at shallow depths, some of which are covered by previous systems. For example, the mean of SimilarWeb's total visits at depth four (5,676,195) is almost the same as that at depth two (5,682,611) and is larger than that at depth one (4,740,811). Also, the sum of AWIS's pageviews per million at depths three and four is 511.27, which is 21.9% of the total. In the data of DNSDB, the number of valid domain names at depth three (181) is less than depth two (425); however, the sum of DNS queries (302,969,603) is larger than that at depth three (52,593,168). Therefore, we showed that there are many malicious domain names that StraySheep reaches by following multiple web pages from landing pages. Also, these domain names, which previous systems cannot reach, have a large number of user accesses. One reason for this is that these domain names are distributed by large-scale advertising providers, as shown in Table 11.

IP Addresses Used for SE Attacks To analyze the relationship between each SE page's domain name, we leveraged DNSDB. The DNSDB enables us to find IP addresses historically associated with domain names. If the same IP address is set in the A record of different domain names, we assume that these domain names are related. As a result of investigating the 1,404 domain names, we detected a total of 96,544 IP addresses associated with 1,349 domain names (55 domain names were not found in the DNSDB). Note that multiple IP addresses were associated with one domain name; thus, there are more IP addresses than domain names. We found that 29.6% (28,617/96,544) of IP addresses were shared among more than two domain names we detected, and these IP addresses (28,617) were associated with 39.5% (554/1,404) of domain names. The 554 domain names were mainly used for multimedia scams, PUP/malware distributions, survey scams, and unwanted browser extension installs. We now focus on 94 multimedia scam domains and their corresponding 20,589 IP addresses. We found that 87.8% (18,086/20,589) IP addresses were shared among more than two multimedia scam domains. One of these IP addresses was shared with 84 multimedia scam domains we detected.

Table 11 Advertising provider domain names redirected to SE domain names.

Advertising provider	# of unique SE domains redirected from ad provider
`newstarads.com`	155
`traktrafficflow.com`	134
`mybestmv.com`	123
`revimedia.com`	122
`naganoadigei.com`	99
`doubleclick.net`	90
`googleadservices.com`	80
`adk2x.com`	75
`clksite.com`	52
`cobalten.com`	41
`bodelen.com`	41
`googlesyndication.com`	36
`cpmstar.com`	29
`go2affise.com`	28
`inclk.com`	23
`digitaldsp.com`	21
`dtiserv2.com`	19
`tradeadexchange.com`	19
`adf.ly`	19
`adreactor.com`	19
`friendlyduck.com`	16
`revcontent.com`	16
`servedbytrackingdesk.com`	16
`reimageplus.com`	14
`adnetworkperformance.com`	14
All advertising provider	427

Geographical Attribution We analyzed the geographical attribution of IP addresses used for SE attacks. We used the same 96,544 IP addresses as the above analysis. We queried GeoIP2 Databases[†] for the country and Autonomous system (AS) information associated with the IP addresses. Table 12 shows the top 10 countries whose IP addresses were used for distributing SE attacks. United States accounted for 72.9% of all IP addresses, Thailand for 2.3%, Mexico for 1.9%, and Vietnam for 1.9%. Table 13 shows the top 10 ASes. We confirmed CDNs and cloud hosting providers are frequently abused for SE attacks. Amazon, Google, and Cloudflare accounted for 50.5, 7.6, and 2.9%.

[†]https://www.maxmind.com/en/geoip2-databases

Table 12 Top 10 countries mapped.

Country	Percentage
UnitedStates	72.9%
Thailand	2.3%
Mexico	1.9%
Vietnam	1.9%
Canada	1.8%
Brazil	1.6%
Korea	1.4%
Indonesia	1.0%
Philippines	0.9%
Taiwan	0.8%

Table 13 Top 10 ASes hosting SE attacks.

AS	Percentage
Amazon.com, Inc.	50.5%
Google LLC	7.6%
Cloudflare Inc	2.9%
Uninet S.A. de C.V.	1.0%
VNPT Corp	1.0%
Level 3 Parent, LLC	0.6%
JasTel Network International Gateway	0.6%
Telefonica Brasil	0.6%
TOT Public Company Limited	0.6%
FPT	0.4%

6. Discussion

In this section, we discuss the limitations of StraySheep and ethical considerations during our study.

6.1 Limitations

There are limitations with StraySheep in terms of system environment, system implementation, and evasion of our system.

System Environment In the evaluation discussed in this paper, StraySheep was run in a single environment. Some SE pages may not serve the same web page every time due to an ad network or cloaking technology. Specifically, a website changes the web page to be delivered according to the source IP address, web-browser environment, and browsing history. In this case, there are SE attacks that cannot be reached in the current StraySheep environment. However, as described in Sect. 3.2, StraySheep does not depend on the selected browser environment and connection network. Thus, preparing multiple browser environments and connection networks enables us to collect environment-dependent SE attacks.

System Implementation StraySheep implements web-search-based URL collection methods; thus, attacks originating from other types of sources (e.g., email) are out of its scope. Since SE attacks attempt to lure more users to their web pages, attackers should prepare landing pages that can be easily visited from popular web platforms, i.e., search engines and social media. StraySheep covered these platforms and retrieve landing pages using easily customizable search queries. Since interacting HTML forms are not implemented in StraySheep's current web-crawling module, it cannot crawl web pages that require login, account creation, and survey. However, StraySheep's SE-detection module can identify these web pages because it uses not only features of the reached web page but also features extracted from the entire sequence.

Evasion There may be an evasion technique against StraySheep's web-crawling module to create a web page that redirects users to SE attacks without preparing any lure elements. This technique leads to a lowering of the collection efficiency of SE attacks of StraySheep. There may be another evasion technique that introduces CAPTCHA authentication in the middle of an SE attack. An SE attack with CAPTCHA authentication cannot be collected with the current implementation of StraySheep. However, these evasion techniques greatly reduce the number of potential victims, which leads to a reduction in the success rate of attacks. Therefore, we believe that it is unlikely that an attacker actually carries them out, as it goes against the current trend of SE attacks.

There may also be an evasion technique against StraySheep's SE-detection module designed as a classification approach. Attackers modify an SE page's appearance to evade the future design and structure of web pages. However, this module also extracts features from the entire sequence of web pages, such as the occurrence of popup windows displaying fake infection alerts and redirections caused by a user's unintended clicks. Thus, we believe it is still difficult for attackers to evade because these features represent attackers' effective techniques to lure users to their web pages.

6.2 Ethical Consideration

Our study followed research ethics principles and best practices [5], [36], [37], [39]. While we conducted parallel crawling for various websites, each of our crawling sessions sequentially traversed web content on the same website, so only a restricted amount of traffic to the website was generated, which did not increase website workload. Our crawling carefully created web requests according to the manner of a real web browser and did not create any harmful web requests breaking or exploiting websites. Due to using a real web browser, our crawling faithfully performs according to the natural behavior of web browsers. Furthermore, the intention of our automated crawling is not to thwart the monetization model of benign web ads. There is no alternative and realistic way to directly observe SE attacks except for active crawling; however, there is a risk of unexpectedly contributing to malicious pay-per-click (PPC) or pay-per-install (PPI) monetization. Our crawling did not intentionally concentrate on specific PPC or PPI services.

7. Related work

Web-based SE attacks and their defenses have been gaining the attention of researchers. We review related work in terms of collecting these attacks and analyzing the attack mechanisms. Duman et al. focused on the visual properties of trick banners, which lure users into clicking on fake links [18]. They built a Firefox browser extension called TrueClick to detect such trick banners based on image processing and machine learning. StraySheep finds lure elements including trick banners, and interacts with them to confirm whether they actually lead users to SE attacks. Rafique et al. analyzed free live streaming services and their ecosystems [8]. They found that users of these services are exposed to ads, malware, and unwanted browser extensions. Our analysis found that not only live streaming services but also web pages showing illegal content, such as music and games, use lure elements to lead users to malware and unwanted browser extensions. Nelms et al. studied the sequences of visited web pages preceding malware downloads in drive-by download and SE attacks [2]. They proposed a system called WebWitness to passively trace back the visited web pages to analyze how users reach the attacks. They also presented a systematic study on successful SE attacks leading to malicious and unwanted software [3]. They categorized and identified the tactics used in such SE attacks to gain users' attention. While these studies [2], [3] passively traced back real victim's traffic, StraySheep actively collects SE attacks and does not rely on real victims. Vadrevu et al. developed a specific web-browser system called ChromePic to enable the reconstruction of SE attacks [12]. ChromePic introduces a detailed snapshot of logging into Chromium to enable the investigation of SE attacks. Whereas ChromePic focuses on forensics after users reached SE attacks, proactive and large-scale crawling of the latest SE attacks. Miramirkhani et al. conducted the first systematic analysis of technical-support-scam web pages [4]. Specifically, they developed a system that can identify such web pages and collect them to show their prevalence, the abused infrastructure, and illicit profits. Srinivasan et al. analyzed technical support scams by focusing on search-engine results and corresponding sponsored advertisements [5]. They generated technical-support-related special search-engine queries to discover previously unknown technical support scams. StraySheep also finds identified technical support scams based on search engine results, as well as scams that require multiple interactions to reach. Kharraz et al. proposed a system called Surveylance [6] to identify survey scams using search engines and a web-crawling approach. While this system identifies only landing pages having survey content (e.g., advertisement in iframe), StraySheep also identifies survey scams by clicking lure elements, which do not display survey content.

8. Conclusion

We proposed a system called StraySheep to crawl web pages and detect multi-step SE attacks. Our key idea is based on (1) simulating multi-step browsing behavior of users to efficiently crawl web pages leading to SE attacks and (2) extracting features from reached web pages as well as the entire sequence of web pages to accurately detect such attacks. Our experimental results indicate that StraySheep can lead to 20% more SE attacks than Alexa top sites and search results of trend words, crawl five times more efficiently than a simple crawling module, and detect SE attacks with 95.5% accuracy. StraySheep will be useful for security vendors, search engine providers, and social-media companies in terms of analyzing trends in SE attacks.

References

[1] T. Koide, D. Chiba, and M. Akiyama, "To get lost is to learn the way: Automatically collecting multi-step social engineering attacks on the web," 15th ACM Asia Conference on Computer and Communications Security, ASIA CCS'20, Taipei, Taiwan, Oct. 2020, ACM, 2020.

[2] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, "Webwitness: Investigating, categorizing, and mitigating malware download paths," 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, Aug. 2015, pp.1025–1040, USENIX Association, 2015.

[3] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, "Towards measuring and mitigating social engineering software download attacks," 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, Aug. 2016., pp.773–789, USENIX Association, 2016.

[4] N. Miramirkhani, O. Starov, and N. Nikiforakis, "Dial one for scam: A large-scale analysis of technical support scams," 24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, Feb.–March, 2017.

[5] B. Srinivasan, A. Kountouras, N. Miramirkhani, M. Alam, N. Nikiforakis, M. Antonakakis, and M. Ahamad, "Exposing search and advertisement abuse tactics and infrastructure of technical support scammers," Web Conference, WWW 2018, Lyon, France, April 2018.

[6] A. Kharraz, W.K. Robertson, and E. Kirda, "Surveylance: Automatically detecting online survey scams," 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, San Francisco, California, USA, May 2018, pp.70–86, 2018.

[7] C.J. Dietrich, C. Rossow, and N. Pohlmann, "Exploiting visual appearance to cluster and detect rogue software," 28th Annual ACM Symposium on Applied Computing, SAC'13, Coimbra, Portugal, March 2013, pp.1776–1783, ACM, 2013.

[8] M.Z. Rafique, T. van Goethem, W. Joosen, C. Huygens, and N. Nikiforakis, "It's free for a reason: Exploring the ecosystem of free live streaming services," 23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, Feb. 2016, pp.21–24, The Internet Society, 2016.

[9] G. Stringhini, C. Kruegel, and G. Vigna, "Shady paths: Leveraging surfing crowds to detect malicious web pages," 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, Nov. 2013, pp.133–144, 2013.

[10] H. Mekky, R. Torres, Z. Zhang, S. Saha, and A. Nucci, "Detecting malicious HTTP redirections using trees of user browsing activity," 2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April– May 2014, pp.1159–1167, 2014.

[11] T. Taylor, X. Hu, T. Wang, J. Jang, M.P. Stoecklin, F. Monrose, and R. Sailer, "Detecting malicious exploit kits using tree-based similarity searches," Proc. 6th ACM on Conference on Data and Application Security and Privacy, CODASPY 2016, New Orleans, LA, USA, March 2016, pp.255–266, 2016.

[12] P. Vadrevu, J. Liu, B. Li, B. Rahbarinia, K.H. Lee, and R. Perdisci, "Enabling reconstruction of attacks on users via efficient browsing snapshots," 24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, Feb.–March

2017.

[13] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, "Hulk: Eliciting malicious behavior in browser extensions," 23rd USENIX Security Symposium, San Diego, CA, USA, Aug. 2014, pp.641–654, USENIX Association, 2014.

[14] X. Xing, W. Meng, B. Lee, U. Weinsberg, A. Sheth, R. Perdisci, and W. Lee, "Understanding malvertising through ad-injecting browser extensions," 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 2015, pp.1286–1295, ACM, 2015.

[15] K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, N. Provos, and M.A. Rajab, "Ad injection at scale: Assessing deceptive advertisement modifications," 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 2015, pp.151–167, IEEE Computer Society, 2015.

[16] Internet Archive, "Heritrix," https://github.com/internetarchive/heritrix3, 2019.

[17] Selenium Developers Group, "Selenium," https://www.seleniumhq.org/, 2019.

[18] S. Duman, K. Onarlioglu, A.O. Ulusoy, W.K. Robertson, and E. Kirda, "TrueClick: Automatically distinguishing trick banners from genuine download links," 30th Annual Computer Security Applications Conference, ACSAC 2014, New Orleans, LA, USA, Dec. 2014, pp.456–465, ACM, 2014.

[19] L. Lu, R. Perdisci, and W. Lee, "SURF: Detecting and measuring search poisoning," 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, Oct. 2011, pp.467–476, ACM, 2011.

[20] L. Invernizzi and P.M. Comparetti, "Evilseed: A guided approach to finding malicious web pages," IEEE Symposium on Security and Privacy, SP 2012, May 2012, San Francisco, California, USA, pp.428–442, IEEE Computer Society, 2012.

[21] H. Yang, X. Ma, K. Du, Z. Li, H. Duan, X. Su, G. Liu, Z. Geng, and J. Wu, "How to learn klingon without a dictionary: Detection and measurement of black keywords used by the underground economy," 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 2017, pp.751–769, IEEE Computer Society, 2017.

[22] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B.Y. Zhao, "Detecting and characterizing social spam campaigns," 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia, Nov. 2010, pp.35–47, ACM, 2010.

[23] S. Lee and J. Kim, "WarningBird: Detecting suspicious urls in Twitter stream," 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, Feb. 2012, The Internet Society, 2012.

[24] N. Nikiforakis, F. Maggi, G. Stringhini, M.Z. Rafique, W. Joosen, C. Kruegel, F. Piessens, G. Vigna, and S. Zanero, "Stranger danger: Exploring the ecosystem of ad-based URL shortening services," 23rd International World Wide Web Conference, WWW'14, Seoul, Republic of Korea, April 2014, pp.51–62, ACM, 2014.

[25] "Tesseract open source OCR engine," https://github.com/tesseract-ocr/tesseract, 2019.

[26] P.F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," British Machine Vision Conference, BMVC 2013, Bristol, UK, Sept. 2013, pp.13.1–13.11, 2013.

[27] "Doc2vec paragraph embeddings," https://radimrehurek.com/gensim/models/doc2vec.html, 2019.

[28] Symantec, "DeepSight intelligence," https://www.symantec.com/services/cyber-security-services/deepsight-intelligence, 2019.

[29] Malwarebytes, "hpHosts," http://www.hosts-file.net/, 2019.

[30] K. Thomas, J.A.E. Crespo, R. Rasti, J.M. Picod, C. Phillips, M. Decoste, C. Sharp, F. Tirelo, A. Tofigh, M. Courteau, L. Ballard, R. Shield, N. Jagpal, M.A. Rajab, P. Mavrommatis, N. Provos, E. Bursztein, and D. McCoy, "Investigating commercial pay-per-install and the distribution of unwanted software," 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, Aug. 2016, pp.721–739, USENIX Association, 2016.

[31] P. Kotzias, L. Bilge, and J. Caballero, "Measuring PUP prevalence and PUP distribution through pay-per-install services," 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, Aug. 2016, pp.739–756, USENIX Association, 2016.

[32] Microsoft, "Microsoft cognitive services Bing search engine APIs," https://azure.microsoft.com/en-us/services/cognitive-services/search/, 2019.

[33] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "AVclass: A tool for massive malware labeling," Research in Attacks, Intrusions, and Defenses - 19th International Symposium, RAID 2016, Paris, France, Sept. 2016, Proceedings, pp.230–253, 2016.

[34] P. Arntz, "Stolen security logos used to falsely endorse pups," https://blog.malwarebytes.com/threat-analysis/social-engineering-threat-analysis/2018/01/stolen-security-logos-used-to-falsely-endorse-pups/, 2018.

[35] "Web of trust," https://www.mywot.com/en/scorecard/etnamedia.net, 2019.

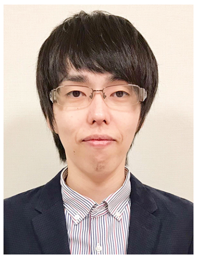
[36] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang, "Knowing your enemy: Understanding and detecting malicious web advertising," ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, Oct. 2012, pp.674–686, ACM, 2012.

[37] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna, "The dark alleys of madison avenue: Understanding malicious advertisements," 2014 Internet Measurement Conference, IMC 2014, Vancouver, BC, Canada, Nov. 2014, pp.373–380, ACM, 2014.

[38] "hosts-blocklists," https://github.com/notracking/hosts-blocklists, 2019.

[39] M. Bailey, D. Dittrich, E. Kenneally, and D. Maughan, "The menlo report: Ethical principles guiding information and communication technology research," Technical Report, U.S. Department of Homeland Security, Aug. 2012.

Takashi Koide received the B.S. and M.S. degrees in Engineering from Yokohama National University in 2014 and 2016, respectively. He is currently a researcher at NTT Secure Platform Laboratories, Tokyo, Japan. His research interests include network and Web security. He won the Research Award from the IEICE Technical Committee on Information and Communication System Security in 2018.



Daiki Chiba is currently a researcher at NTT Secure Platform Laboratories, Tokyo, Japan. He received his B.E., M.E., and Ph.D. degrees in computer science from Waseda University in 2011, 2013, and 2017. Since joining Nippon Telegraph and Telephone Corporation (NTT) in 2013, he has been engaged in research on cyber security through data analysis. He won the Research Award from the IEICE Technical Committee on Information and Communication System Security in 2016, 2018, and 2019 and the Best Paper Award from the IEICE Communications Society in 2017. He is a member of IEEE and IEICE.

Mitsuaki Akiyama received his M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Japan in 2007 and 2013. Since joining Nippon Telegraph and Telephone Corporation (NTT) in 2007, he has been engaged in research and development on cybersecurity. He is currently a Senior Distinguished Researcher with the Cyber Security Project of NTT Secure Platform Laboratories. His research interests include cybersecurity measurement, offensive security, and usable security and privacy.

Katsunari Yoshioka is an Associate Professor at Yokohama National University since 2011. His research interests cover wide area of system security and network security including malware analysis and IoT security. He received the commendation for science and technology by the minister of MEXT, Japan in 2009, the award for contribution to Industry-Academia-Government Collaboration by the minister of MIC, Japan in 2016, and the Culture of Information Security Award in 2017.

Tsutomu Matsumoto is a professor of Faculty of Environment and Information Sciences, Yokohama National University and directing the Research Unit for Information and Physical Security at the Institute of Advanced Sciences. He is also the Director of Cyber Physical Security Research Center (CPSEC) at National Institute of Advanced Industrial Science and Technology (AIST). He received Doctor of Engineering from the University of Tokyo in 1986. Starting from Cryptography in the early 80's, he has opened up the field of security measuring for logical and physical security mechanisms. Currently he is interested in research and education of Embedded Security Systems such as IoT Devices, Cryptographic Hardware, In-vehicle Networks, Instrumentation and Control Security, Tamper Resistance, Biometrics, Artifact-metrics, and Countermeasure against Cyber-Physical Attacks. He is serving as the chair of the Japanese National Body for ISO/TC68 (Financial Services) and the Cryptography Research and Evaluation Committees (CRYPTREC) and as an associate member of the Science Council of Japan (SCJ). He was a director of the International Association for Cryptologic Research (IACR) and the chair of the IEICE Technical Committees on Information Security, Biometrics, and Hardware Security. He received the IEICE Achievement Award, the DoCoMo Mobile Science Award, the Culture of Information Security Award, the MEXT Prize for Science and Technology, and the Fuji Sankei Business Eye Award.