ROBUST SCAREWARE IMAGE DETECTION

Christian Seifert, Jack W. Stokes
Christina Colcernian, John C. Platt

Microsoft Corp.
Redmond, WA 98052

Long Lu

Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

In this paper, we propose an image-based detection method to identify web-based scareware attacks that is robust to evasion techniques. We evaluate the method on a large-scale data set that resulted in an equal error rate of 0.018%. Conceptually, false positives may occur when a visual element, such as a red shield, is embedded in a benign page. We suggest including additional orthogonal features or employing graders to mitigate this risk. A novel visualization technique is presented demonstrating the acquired classifier knowledge on a classified screenshot.

Index Terms— security, scareware, social engineering

1. INTRODUCTION

Botnets with thousands of bots are controlled by cyber criminals today. These bots primarily reside on the average desktop computer at home, at one's work place, or in Internet Cafes. Client-side attacks are used by these criminals to infect one's machine and join it to the botnet. In the first half of 2011, several primary client-side attack vectors exist: drive-by-download attacks, AutoRun attacks, and social engineering attacks [1]. Web-based social engineering attacks present false information to the user and trick her into downloading and executing malware herself. As Internet threats are heavily touted in the community and media, attackers have adopted a social engineering angle in this space to accomplish their goal: scareware. Scareware attacks typically attempt to trick computer users into believing their computer has been infected and offering them a solution by purchasing a fake anti-virus product that allegedly removes the malware from their computer. There are numerous challenges in detecting scareware attacks: obfuscation and a wide array of technology that can be used to implement such an attack.

In this paper, we present a new, *robust* detection technology that aims to identify the scareware social engineering attack at a layer that the attacker cannot arbitrarily obfuscate: the visual images presented to the end user. The system is designed to be robust to evasion tactics and also independent of the language and underlying technology used to implement the attack.

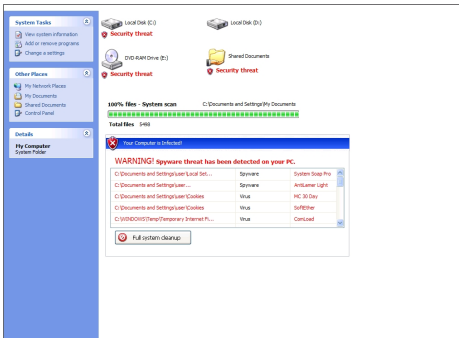

Fig. 1. Scareware Screenshot

Important contributions in this work include: I. a robust, language-independent image recognition algorithm that detects the scareware social engineering attack. II. a classification system that learns potential image obfuscation techniques. III. a large-scale implementation that illustrates the proposed methods are indeed effective. IV. a visualization technique for displaying the acquired classifier knowledge is presented.

2. BACKGROUND

Scareware: Many computer users have been affected by a scareware attack where the operating system or an antivirus (AV) program apparently runs a scan on their computer and appears to detect multiple instances of malware. In reality, the user's browser renders a web site displays an animation providing the illusion of the AV scan as shown in Figure 1. These animations visually try to match the look and feel of the underlying operating system with the goal of selling the fake product to the user and infect the user's machine [2]. Note that the visualization is entirely implemented using browser rendering techniques (e.g. HTML, Flash, JavaScript).

Detection Countermeasures: Operators of scareware attacks want to distribute malware and make money. As a result, they have a vested interest to remain undetected and be resilient against interruption of "service". First, the attacker can influence where the content responsible for the scareware

animation as well as the actual malware is hosted. These – so called scareware servers – could be running on compromised servers or on servers with hosting providers that don't respond to abuse reports. Often, the infrastructure is only utilized for a short duration of time [3] making it hard to detect and maintain a comprehensive list of scareware servers. Second, attackers can change the content itself through polymorphism and content obfuscation. This technique thwarts widely deployed signature-based approaches to detect scareware.

As the user is presented with a scareware animation, image-based detection techniques may be used to identify the scareware attack. However, it appears that scareware authors are aware of such techniques and are taking measures to thwart such detection to a certain extent as well. They may slightly modify individual images that are used to construct the animation. For instance, they may change one pixel on the disk drive icon or split the image into multiples blocks. Visual elements may be rearranged or an attacker could use different icons altogether. However, the attacker needs to walk a fine line between presenting a user with an animation that resembles the operating system and taking measures to evade image-based detection. If the attacker uses different icons altogether, the deception that the operating system or legitimate AV software scans the system for threats may break down. The result could be a decrease of conversion rates and subsequently a smaller return of investment for the attacker.

Image-Based Detection: As mentioned in the previous section, image-based detection may be one way to identify scareware. Several approaches can be adopted including exact match, near match, or similar match. In the exact match image matching method, one may apply a cryptographic hash (e.g. Sha256) of visual components. This is a rather fragile. Near image matching addresses the short coming of exact matches. In this case, visual descriptors, e.g. Daisy descriptors [4], are used to match images. Even slight modifications to the image may result in a match. However, a reordering or changing of visual elements may result in a mismatch and, as scareware tends to change the layout of their animation, near image matching would not be suitable to detect scareware attacks. We next describe a similar match algorithm to address this limitation.

3. SCAREWARE IMAGE CLASSIFICATION

In this section, we describe the proposed scareware image classification system which operates on screenshots (e.g. .jpg) of web pages. The goals of the classifier are to capture the common visual elements of scareware attacks and to identify screenshots of web pages containing similar elements.

Classifier Training: The classifier is trained from a set of web page screenshots, $\mathcal{S}$. Scareware attack images are included in $\mathcal{S}_{\mathcal{A}} \subset \mathcal{S}$. To describe an individual screenshot s_i, a set of interest point descriptors $\mathcal{D}$ are extracted by the Daisy detector algorithm [4]. One can assess the similarity of two descriptors through their Euclidian distance. Each descriptor $d_{ij} \in \mathbb{R}^M$ is a histogram of gradients surrounding the region of the interest point j in screenshot s_i while M is the dimension of the descriptor. In Table 1, we denote $d_{i\bullet}$ to be the set of all descriptors for s_i, and we use $M = 32$ which is the default setting for the Daisy algorithm. While the Daisy algorithm reliably identifies the interest points, small variations from screenshot to screenshot may result in *similar* descriptors. In order to identify descriptors which are common among the various scareware attack images, we learn clusters of descriptors which are important in scareware attacks. Using screenshots from the scareware attacks, we use a variation of the simple IsoData clustering algorithm [5, 6] to learn the clusters $\mathcal{C}$. The cluster medoid, $\hat{c}_j$, is next determined to be the descriptor closest to the mean of all descriptors contained in the cluster. Once the cluster center descriptor has been determined, the maximum cluster distance, T_j^C, from this cluster medoid to all descriptors contained within the cluster is computed. After all clusters have been determined, a labeled dataset containing a row for each labeled image (scareware, benign) is created. Formally, consider the hypothesis space $X \times Y$, where $X \in \mathbb{R}^d$, $Y \in \{-1, 1\}$. Here d is the dimension of the feature vector consisting of the number of selected cluster features. For each observation pair $\langle x_i, y_i \rangle$, $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$ for $i = 1..|\mathcal{D}|$. The features for the classifier consist of a histogram where each bin represents the number of descriptors that were assigned to the constructed clusters. A descriptor is assigned to cluster j when its Euclidian distance from the center cluster descriptor is less than or equal to the clusters maximum distance, T_j^C. The main three steps are explained in detail in this section and the algorithm is summarized in Table 1.

With the feature vector at hand, several different classifiers are trained including logistic regression with stochastic gradient descent (SGD) optimization and early stopping [7], logistic regression with L-BFGS optimization and L1 and L2 regularization [8], boosted decision trees using MART [9], and the averaged perceptron [10].

Prediction: The prediction of unknown screenshots has similar steps taken during the training phase. First the descriptors need to be extracted from the unknown screenshots. Its descriptors are assigned to the constructed clusters as described previously. From that assignment, a feature vector is constructed that can be fed into the classifier to generate a score. A score larger than a predefined threshold indicates a scareware page has been identified by the classifier whereas a lower score would indicate the opposite.

The weights from linear classifiers (e.g. logistic regression) have the capability to identify important clusters in determining whether a screenshot shows scareware. A visual representation of the acquired classifier knowledge can therefore be created. A scareware example is shown in Figure 2. Descriptors assigned to a cluster that bares a scareware weight are shown in dark red. With the weights one can identify

Input: Labeled Set of Screenshots $\mathcal{S}$
Initialize: $\mathcal{C} = \phi$
Extract interest points and interest point descriptors d_{ik} **from screenshot** $s_i \; \forall i, k$
Create Clusters:
foreach s_i **in** $\mathcal{S}_\mathcal{A}$ **do**
  if $||d_{ik} - c_{jl}||_2 < T_D \; \forall j, k, l$
    Assign d_{ik} to c_j
  else
    Create cluster $c_{\mathcal{C}+1}$ from d_{ik}
end foreach
Determine medoid $\hat{c}_j$ **and distance threshold** $T_j^C \; \forall j$
Create Feature Vectors:
for i = 1 to $\mathcal{S}$ **do**
  for k = 1 to $d_{i\bullet}$ **do**
    for j = 1 to $\mathcal{C}$ **do**
      if $||d_{ik} - \hat{c}_j||_2 < T_j^C$
        Increment x_{ij}
    end for
  end for
end for
Train Classifier using $(\mathbf{x}, \mathbf{y})$

Table 1. Algorithm for Classifying Scareware Webpages.

which areas of the image the classifier latches onto to determine whether the screenshot shows scareware.

4. METHODOLOGY AND RESULTS

A crawler was used to collect screenshots of benign and scareware pages. Benign pages were collected by crawling the most popular URLs from a search engine index. URLs that were likely to harbor scareware pages were also taken from a search engine index. These were identified using a variety of heuristics and AV technology. As only a few unique scareware sites could be identified this way, we proceeded to boost the scareware dataset with manually constructing scareware pages randomizing language, layout, font, color schemes, etc. The collected screenshots were processed as described in section 3. A 5-fold cross validation was performed. Further, the classifier was used to evaluate four additional sets of screenshots to get a sense of false positive/negative rate of the classifier: 12,551 screenshots obtained from a feed of phishing URLs, a large set of screenshots from a search engine, a set of mocked up screenshots that contained visual elements of a scareware attack (e.g. disk icon), but did not represent a scareware attack, and lastly a set of scareware screenshots that had undergone image transformations, such as stretching.

The classifier was trained on a set of 7475 screenshots. In the initial round of training, 518 clusters were created. Screenshot collection and training on a standard desktop PC

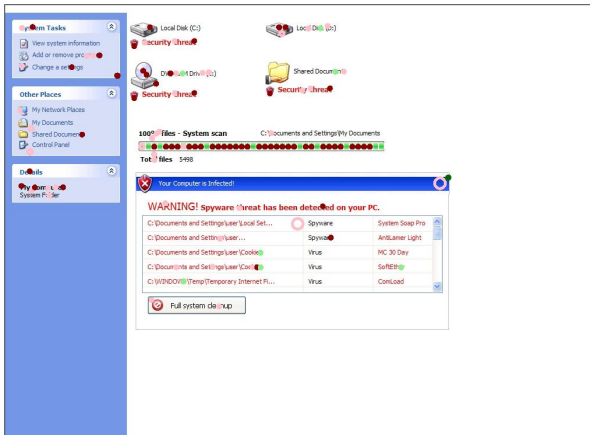

Fig. 2. Visualization of interest points assigned to a cluster

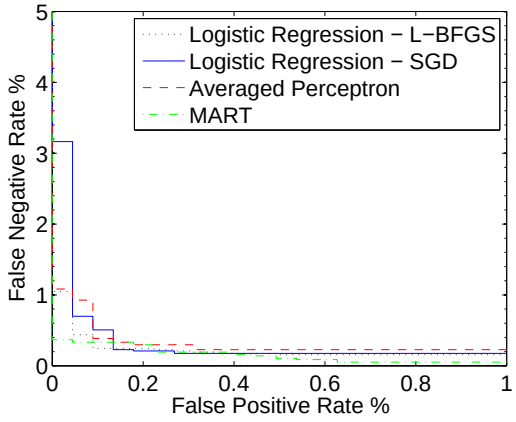

Fig. 3. DET curve for the four spyware classifiers

(Intel Xeon 3.07GHz with 12GB of RAM) took 372 minutes. The majority of time is spent on collecting screenshots and cluster creation. Once the feature vector is constructed, training takes little time. Classification of each screenshot takes approximately 664 milliseconds.

The DET curves for the four algorithms are shown in Figure 3. The equal error rate for logistic regression with SGD is the best at 0.018%. Only a few scareware screenshots are missed by our classifier at this setting. Visual inspection reveals that those screenshots were not rendered completely. Turning to the set of suspected phishing pages shows how well the classifier performs on a completely unknown set. It turns out that the 12,551 suspected phishing pages only had one scareware page which was identified. In addition, 14 false positives from 3 sites were raised. This illustrates the external validity of the classifier. It is capable of identifying new scareware pages from different sources with a false positive rate of 0.1%.

Turning to a larger dataset from a search engine index, we evaluated two days: 157,142 Urls. If a similar scareware

base rate as with the fully labeled phishing set is assumed (1/12,551 or 0.008%), we would expect about 13 scareware pages in the set of 157,142 URLs. We identified 37 URLs in the .in (India) top-level domain that showed an unknown scareware attack (score 0.68-0.88). No false positives were raised. This experiment validates the ability of the proposed algorithm to detect new, similar scareware attacks.

Experimenting with mocked up images reveals a scenario in which the classifier could lead to a false positive. While inclusion of a smaller scareware screenshot did not result in a false positive, inclusion of visual components with a strong weight, such as the shield, did (score 0.81). As such, there is the risk of incorrectly detecting benign pages that share common visual elements as scareware. Image transformation allows us to assess how a determined attacker may evade detection by our classifier. It turned out that the current training set does not yield a classifier that is robust against brightness, saturation, resizing and stretching. However, when retraining the classifier with images that had undergone these image transformation, it resulted in a robust classifier.

5. DISCUSSIONS

We have developed an image-based classifier that is capable of identifying scareware attacks. The classifier is robust in that it is capable of identifying common visual elements of scareware attacks and correctly labeling unknown screenshots. Rearranging the layout, and visual elements – an evasion technique that is generally effective against other image matching techniques – are ineffective against our classifier. Further, scareware pages in different languages were detected.

The detection accuracy of the classifier is favorable. However, false positives are possible and mitigation strategies are recommended, such as inclusion of additional URL-based features or validating positive classifications through human graders. A cost-based analysis reveals that the cost of identifying scareware URLs is decreased by orders of magnitude even when human graders are involved.

The proposed classifier is robust against rearranging layout, visual elements, and language. Knowing how the classifier works, an attacker could construct a scareware animation to evade our system. For instance, one visual element or even just one pixel is being displayed at a given time. Our efforts to construct such a page resulted in a flickering animation that would alert most users similar to some of the other image transformation presented. An attacker may be able to use different images altogether, but they must do so while at the same time maintaining the deception; E.g. inclusion of a folder icon that is unfamiliar to the user may destroy the illusion of the scan animation originating from the operating system or legitimate AV software.

Further, it is possible that scareware providers identified our crawler or crawling infrastructure and cloak the scareware attack. In those instances, the crawler is never presented with the scareware animation. This has been illustrated by numerous researchers [11, 12, 13]. Note that our method, however, is not tied to being deployed on a crawler. Conceivably, the method could be implemented on the client itself.

6. RELATED WORK

Scareware has only recently received attention in the research community. Rajab et al. performed a large scale study on scareware sites in 2010 [3]. While the researchers do not disclose how scareware sites are detected, they provide information about prevalence, network structure and lifetime of scareware attacks over a period of 13 months. Symantec analyzed a wide range of rogue security software and shown light on distribution channels and economics [14]. Several researchers focused on the economic aspects of rogue AV campaigns to provide a possible answer to their rising popularity [15, 16, 2]. Note that these studies appear to incorporate FakeAV malware binaries that may be distributed through different means other than a web page. Our detection method focuses on web pages that show an animation of a scareware scan.

Utilizing computer vision to identify social engineering attacks is not new. Wang et al. applied near duplicate detection to image spam [17]. Mostly, it has been applied in the phishing context though [18, 19, 20, 21]. Medvet et al. developed a classifier that identifies phishing pages based on visual text, images contained on the web page as well as a rendering of the web page [20]. An alternative approach to identify phishing pages focuses on layout similarity [19].

7. CONCLUSIONS

In this work, we show that detecting a web-based social engineering attack, such as scareware, on the final presentation layer is an effective detection method. As our detection technology is based on the final layer presented to the end user, the attacker has little opportunity to evade detection as the deceptive nature of the social engineering attack needs to be upheld. We illustrated some evasion techniques, such as reordering of the icons and changing hue, as possible avenues for evasion, but our classifier remains mostly robust. More importantly, however, our approach is independent of the underlying technology used to render the content. Whether the social engineering attack is rendered in Silverlight within a browser window or HTML5 is of no importance to our classifier. Tackling the detection at the presentation layer abstracts the technology away making our detection strategy robust to new technologies. Of course, detection accuracy is of importance, and we have shown that false alerts are possible with our approach and we touched on a variety of mitigation techniques. Overall, we believe the detection approach that operates on the presentation layer may be a beneficial addition to protecting the user from a wide range of social engineering attacks, such as phishing.

8. REFERENCES

[1] "Security intelligence threat report v11," Microsoft Corporation, Available from http://www.microsoft.com/sir/; last accessed 10/25/2011.

[2] Brett Stone-Gross, Ryan Abman, Richard A. Kemmerer, Christopher Kruegel, Douglas G. Steigerwald, and Giovanni Vigna, "The underground economy of fake antivirus software," in *Proceedings of 10th Workshop on Economics of Information Security (WEIS)*, 2011.

[3] Moheeb Abu Rajab, Lucas Ballard, Panayiotis Mavrommatis, Niels Provos, and Xin Zhao, "The nocebo effect on theweb: An analysis of fake anti-virus distribution," in *Proceedings of the 3rd Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2010.

[4] Simon Winder, Gang Hua, and Matthew Brown, "Picking the best daisy," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2009.

[5] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[6] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley, 1974.

[7] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[8] Galen Andrew and Jianfeng Gao, "Scalable training of l1-regularized log-linear models," in *Proceeding of the International Conference on Machine Learning*, 2007.

[9] J. Friedman, "Greedy function approximation: a gradient boosting machine," in *Annals of Statistics*, 2001, pp. 1189–1232.

[10] Y. Freund and R. Schapire, "Large margin classification using the perceptron algorithm," in *Machine Learning*, 1999, pp. 277–296.

[11] Billy Hoffman, "Circumventing automated javascript analysis," in *Proceedings of BlackHat*, 2008.

[12] Moheeb Abu Rajab, Lucas Ballard, Nav Jagpal, Panayiotis Mavrommatis, Daisuke Nojiri, Niels Provos, and Ludwig Schmidt, "Trends in circumventing web-malware detection," in *Google Technical Report rajab-2011a*, 2011.

[13] Kyle Zeeuwen, Matei Ripeanu, and Konstantin Beznosov, "Improving malicious url re-evaluation scheduling through an empirical study of malware download centers," in *Proceedings of the World Wide Web Conference*, 2011.

[14] Symantec, Inc., "Symantec report on rogue security software," 2009, Available from http://www4.symantec.com/Vrt/wl?tu_id=XuOB125692283892572210; last accessed 10/17/2011.

[15] Sean-Paul Correll and Luis Corrons, "The business of rogueware," 2009, Available from http://www.pandasecurity.com/img/enc/The%20Business%20of%20Rogueware.pdf; last accessed 10/17/2011.

[16] Marco Cova, Corrado Leita, Olivier Thonnard, Angelos D. Keromytis, and Dacier Marc, "An analysis of rogue av campaigns," in *Proceedings of Recent Advances in Intrusion Detection*, 2010, pp. 442–463.

[17] Zhe Wang, William K. Josephson, Lv Qin, Moses Charikar, and Kai Li, "c," in *Conference on Email and Anti-Spam*, 2007.

[18] Teh-Chung Chen, Scott Dick, and James Miller, "Detecting visually similar web pages: Application to phishing detection," *ACM Trans. Internet Technol.*, vol. 10, pp. 5:1–5:38, June 2010.

[19] Ieng-Fat Lam, Wei-Cheng Xiao, Szu-Chi Wang, and Kuan-Ta Chen, "Counteracting phishing page polymorphism: An image layout analysis approach," in *Proceedings of the 3rd International Conference and Workshops on Advances in Information Security and Assurance*, 2009.

[20] Eric Medvet, Engin Kirda, and Christopher Kruegel, "Visual-similarity-based phishing detection," in *Proceedings of SecureComm*, 2008.

[21] Liu Wenyin, Guanglin Huang, Liu Xiaoyue, Zhang Min, and Xiaotie Deng, "Detection of phishing webpages based on visual similarity," in *Special interest tracks and posters of the 14th international conference on World Wide Web*, New York, NY, USA, 2005, WWW '05, pp. 1060–1061, ACM.