Exploiting Visual Appearance to Cluster and Detect Rogue Software

Christian J. Dietrich[1,3], Christian Rossow[1,2], Norbert Pohlmann[1]

{dietrich|rossow|pohlmann}@internet-sicherheit.de

[1] Institute for Internet Security, University of Applied Sciences Gelsenkirchen, Germany
[2] Department of Computer Science, VU University Amsterdam, The Netherlands
[3] Department of Computer Science, University of Erlangen, Germany

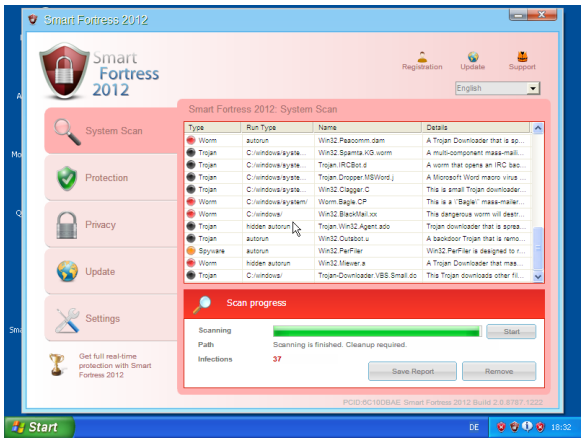
ABSTRACT

Rogue software, such as Fake A/V and ransomware, trick users into paying without giving return. We show that using a perceptual hash function and hierarchical clustering, more than 213,671 screenshots of executed malware samples can be grouped into subsets of structurally similar images, reflecting image clusters of one malware family or campaign. Based on the clustering results, we show that ransomware campaigns favor prepay payment methods such as ukash, paysafecard and moneypak, while Fake A/V campaigns use credit cards for payment. Furthermore, especially given the low A/V detection rates of current rogue software – sometimes even as low as 11% – our screenshot analysis approach could serve as a complementary last line of defense.

1. INTRODUCTION

While malware comes in many different flavors, e.g., spam bots [3, 13], banking trojans [2, 10] or DDoS bots [7], one important monetization technique of recent years is rogue software, such as fake antivirus software (Fake A/V) [9]. In this case, the user is tricked into spending money for a rogue software which, in fact, does not aim at fulfilling the promised task. Instead, the rogue software is malicious, and entices the user to pay. However, all rogue software has in common to provide a user interface, e.g., be it to scare the user, or in order to ask for banking credentials, or to carry out the payment process.

Figures 1(a) and 1(b) show example screenshots of two typical rogue software flavors. The first displays a Fake A/V user interface, while the second exhibits a ransom screen (in German), asking the user to pay before the computer is unlocked. As rogue software is required to provide a user interface, we aim at exploiting its visual appearance in order to cluster and classify rogue software. We motivate our efforts by the relatively low A/V detection rates of such rogue software, and we aim to complement existing techniques to strive for better detection rates. In particular, we observed



(a) Smart Fortress 2012 (Winwebsec family)

(b) Ransomware asking the user to pay (in German)

Figure 1: Example screenshots of rogue software

SAC'13 March 18-22, 2013, Coimbra, Portugal.

that the structure of the user interfaces of rogue software remains constant and can be used to recognize a rogue software family or campaign. Using a perceptual hash function and a hierarchical clustering approach, we propose a scalable and effective approach to cluster associated screenshot images of malware samples.

In short, the main contributions of this work are threefold:

- We provide a scalable method to cluster and classify rogue software based on its user interface, an inherent property of rogue visual malware.
- We applied our method to a corpus of more than 187,560 malware samples of more than 2,000 distinct families (based on Microsoft A/V labels) and revealed 25 distinct types of rogue software user interfaces. Our method successfully reduces the amount of more than 187,560 malware samples and their associated screenshot images down to a set of human-manageable size, which assists a human analyst in detecting, understanding and combating Fake A/V and ransomware.
- We provide insights into Fake A/V and ransomware campaigns as well as their payment means. More specifically, we show a distinction of payment methods between Fake A/V and ransomware campaigns.

2. METHODOLOGY

A coarse-grained overview of our methodology is shown in Figure 2. Our approach consists of three steps. First, we execute malware samples and capture the screen. Furthermore, we compute a perceptual hash of the resulting image and finally, we cluster screenshots into subsets of similar appearance. Our goal is to find subsets of images that – although slightly different in detail – exhibit a similar structure and a similar user perception.

We found that the user interfaces of Fake A/V campaigns vary concerning details such as the number of supposedly dangerous files as well as the rogue software name and logo. However, the *position* of the logo and the *sizes* of user interface elements remain constant among several campaigns of one family. As an example, Figures 3(a) and 3(b) display screenshots of two malware samples of the Winwebsec family, and Figures 3(c) and 3(d) display screenshots of two FakeRean malware samples. While the two Winwebsec samples (Smart Fortress 2012 and System Tool) show different logos at the top left corner of the application window and different icons for the application menu, their overall user interface structure is very similar. The same applies to FakeRean. Again, the name differs – Internet Security vs. Spyware Protection – but the application windows and their user interface elements are positioned in the same fashion.

For our clustering step, we aim at separating the images of the Winwebsec family from those of the FakeRean family. Furthermore, if possible, different campaigns should be separated. In other words, we aim at recognizing the structure of the visual appearance in a screenshot, e.g., concerning the position of a program or campaign logo and the sizes of user interface elements, but at the same time ignore detailed information such as colors, the exact program name, window title or the text inside user interface elements.

In fact, we found that rogue software is required to change its name regularly – most likely because users searching for such a software name on the Internet will eventually find out that they fall prey to a rogue software campaign. Our basic assumption can be fostered in the observation that rogue software seems to build its user interface from templates which remain constant and only the contents of certain user interface elements differs. Thus, our clustering approach specifically targets these templates.

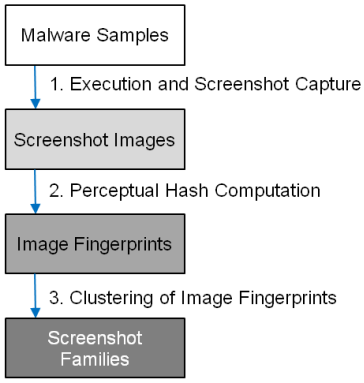


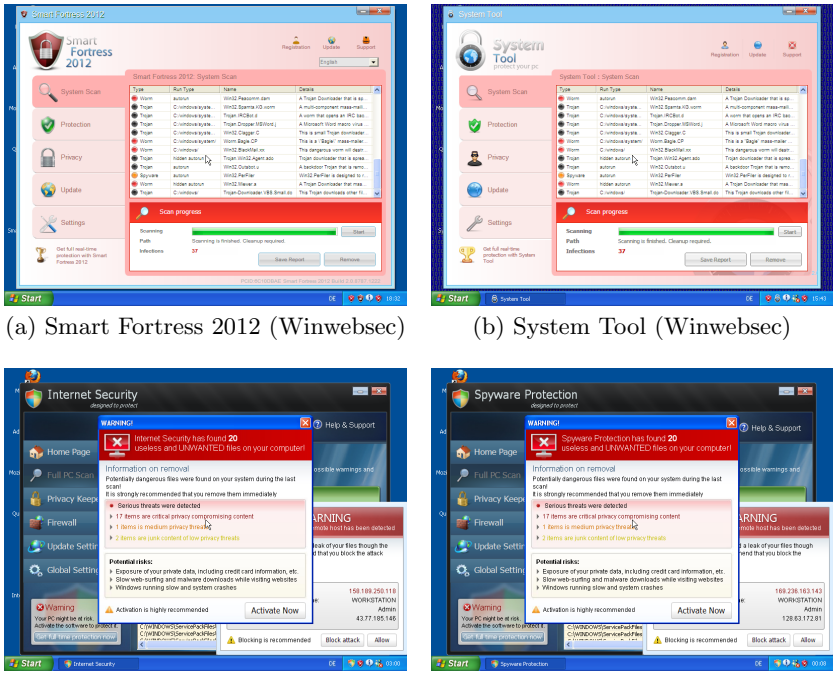
Figure 2: Overview of the screenshot clustering methodology

2.1 Dataset

For our image clustering technique, we compiled a corpus of 213,671 screenshots that originate from executing 213,671 MD5-distinct malware samples representing more than 2,000 malware families based on Microsoft A/V labels, spanning two years. Although by far, most of the samples in our malware feeds are indeed malicious, occasionally, a sample represents legitimate software, e.g., Adobe Flash Installer. For example, this stems from the fact that some samples are gathered from public submission systems where users are allowed to upload all kinds of software. However, for our approach, we see no need to make sure *all* legitimate software was excluded. In our clustering results, we expect legitimate software to be well-separated from rogue visual software because they exhibit different user interfaces.

2.2 Malware User Interfaces

In order to capture the visual appearance of a malware sample, we execute the malware sample in a virtual machine and store a screenshot of the whole virtual screen. Typically, each sample is executed for one hour. The virtual machines use Windows XP 32bit SP3 as operating system and have limited Internet access. SMTP as well as typical infection vector ports were transparently redirected to local spam traps and honeypots. During the execution of a sample, no user interaction was performed. In order to prevent harming others and to mitigate denial-of-service attacks, we restricted outgoing traffic to the protocols IRC, DNS and HTTP and throttled outgoing traffic. As the set of executed samples covers all kinds of malware, some of the screenshot images do not show any graphical user interface at all, e.g., malware operating completely in the background, or display an error message. We will deal with the fact that not all screenshots reflect rogue visual software in subsequent sections.



(a) Smart Fortress 2012 (Winwebsec) (b) System Tool (Winwebsec)

(c) Internet Security (FakeRean) (d) Spyware Protection (FakeRean)

Figure 3: Screenshot images of the Winwebsec and FakeRean malware families

2.3 Perceptual Hash Functions

In our case, one screenshot image consists of 800x600 pixels of 24 bit color depth, resulting in ca. 11 MB of uncompressed data per image. To reduce the amount of data, we extract a fingerprint of the visual appearance, for each image. This fingerprint is then used in the clustering phase. As we are mainly interested in the perception, we turn to so-called perceptual hash functions. While cryptographic hash functions aim at resulting in two different hash values upon the slightest difference between two input values, perceptual hash functions aim at grasping semantic differences of a certain context.

We use the perceptual hash function p based on the discrete cosine transform (DCT) as proposed by Zauner [14]. First, an image is converted to grey scale and a smoothing for each 7x7 pixel subarea is performed. Furthermore, the image is resized to 32x32 pixels and the two-dimensional type-II DCT coefficients are calculated. As high frequencies might not sustain compression, 64 low-frequency coordinates (8x8) are considered – similar to the JPEG compression standard – to build up an 8-byte bitstring fingerprint. Effectively, this fingerprint provides a structural context-sensitive feature vector of the screenshot image.

An advantage of this perceptual hash function is its robustness against minor modifications to the visual appearance such as a change in color, certain levels of distortion and non-uniform scaling. Thus, even if certain elements of a rogue software user interface are changed, distorted or blurred – for example in order to result in different cryptographic hash values – the perceptual hash function will resist (to a certain degree). During the manual classification, which will be described in more detail in Section 2.4, we found samples that changed the desktop wallpaper and their perceptual hash value is close to the ones with the regular wallpaper. Also we found user interfaces where the colors of some user interface elements were changed, which also produced nearby perceptual hash values.

2.4 Intra-Fingerprint Coherence

For all of the 213,671 screenshot images, we applied the perceptual hash function, which results in a set of 17,767 distinct perceptual hash fingerprint values. We denote the full set of perceptual hash fingerprints as F. Note that not all of these fingerprints refer to user interfaces of rogue software, but also include error messages – e.g., cases where the sample to be executed was corrupt – or a blank screen, i.e., the sample did not exhibit a visual appearance at all.

Using a perceptual hash function, two images with a very high degree of structural similarity might result in the same hash value. While in general, this fact is desired, in the worst case, two images with the same hash value might not reflect the structural and perceptual similarity we aim at. In this case, we would need to extend the fingerprint either by increasing the fingerprint length or by taking additional features into account. We define the term *Intra-Fingerprint coherence* to reflect that images with the same fingerprint indeed provide the same structural and perceptual properties and thus likely reflect images of the same malware family or campaign.

In order to test our fingerprints for Intra-Fingerprint coherence, we randomly selected 345 fingerprint values (ca. 2% of 17,767) with at least 35 associated screenshot images per fingerprint, and for each fingerprint we inspected at least three random images manually. We checked whether the positions of user interface elements remain constant, especially the positions of logos, progress bars and text area, list or table elements. In all cases, the images provided the required similarity in structure.

Perceptual Hash Label	MS A/V Family
Fake A/V and Rogue	
Clean Catch	undetected
Smart Fortress 2012	Winwebsec
System Tool	Winwebsec
Personal Shield Pro	undetected
FakeScanti	FakeScanti
S.M.A.R.T Failure	FakeSysdef
Security Tool	Winwebsec
Internet Security	FakeRean
XP Home Security 2012	FakeRean
Security Monitor 2012	undetected
Antivirus Protection 2012	FakeRean
Spyware Protection	FakeRean
Antivirus Action	undetected
PC Performance and Stability	FakeSysdef
Security Shield	undetected
Security Central	undetected
Windows Trojans Sleuth	FakePAV
Microsoft Security Essentials	FakePAV
Ransomware	
Windows Security Center	Ransom
Bundespolizei	Sinmis

Table 1: Perceptual Hash Fingerprint Labels for 18 fake A/V and 2 ransomware campaigns and, if available, Microsoft A/V Family Labels

At the same time, we classified the fingerprints and assigned a campaign label or – in case the screenshot does not show rogue software – a state label such as whether an error message, a blank screen or a browser window is displayed or describe the application window. For example, a blank screen occurs if the malware operates completely in the background, e.g., in case of Zeus, a popular banking trojan [2]. A prevalent group of fingerprints is caused by the Hotbar/Clickpotato downloader which masks as installers for various software such as VLC or xvid. Other examples for malware which show neither Fake A/V nor ransomware include Adware causing the web browser to open a specific website, cracks or serial key generator programs and Windows Explorer displaying the contents of a certain folder.

The labeled set of 345 fingerprints covers 18 different fake A/V campaigns and two ransomware campaigns, shown in Table 1. The second column of Table 1 denotes the Microsoft A/V family label, if at least one of the associated samples was detected by Microsoft.

2.5 Distance Computation and Clustering

While the perceptual hash computation relaxes near-duplicates, i.e., it aggregates very similar images into one fingerprint value, a malware family or campaign typically spans multiple different fingerprint values. In other words, one fingerprint is too specific to reflect a campaign or even a whole malware family. We found several reasons for this. First, the application window might not always be at the same position on the screen or other foreground application windows might obscure parts of the rogue software's window(s). This could result in different perceptual hash values. Second, rogue software interfaces can provide several different views. For example, a fake A/V program might provide one view of a scan in progress (Figure 3(a)) and another one when the scan is finished and the user is prompted for action, e.g., as shown in Figure 3(c).

Thus, in order to aggregate several closely related fingerprints into subsets of campaigns or malware families, we add a clustering step. In this clustering phase, we use the normalized bit-wise hamming distance as distance function between two perceptual hash fingerprints. As the perceptual hashes have a fixed length of 8 bytes, the hamming distance returns the number of differing bits, with a maximum of 64. We normalize the hamming distance to the range 0.0 to 1.0 by dividing it by 64.

Since the perceptual hash is a locality-sensitive hash [6], we can rely on the hamming distance as a simple and fast distance function. Similar images will result in a low hamming distance (towards 0.0), while different images will exhibit higher distance values (towards 1.0). Another advantage of the hamming distance is its high performance (XOR arithmetic). The combination of the perceptual hash function and the normalized hamming distance allows us to use agglomerative hierarchical clustering despite its worst-case complexity of $\mathcal{O}(n^3)$.

Thus, we apply agglomerative hierarchical clustering to the set of fingerprint values. Furthermore, we decided to use average linkage clustering to avoid the chaining phenomenon. Finally, to aggregate similar fingerprints into one cluster, a cut-off threshold determines the maximum distance between any two different fingerprint clusters. If the distance of any two distinct fingerprint clusters is less than the cut-off threshold, the associated screenshot fingerprints will be filed in the same cluster, otherwise they would be filed into different clusters.

3. EVALUATION

Our clustering evaluation can be divided into two parts, Intra-Fingerprint Coherence as well as Cluster Generalization. In addition, we evaluate the true A/V detection rate of Fake A/V campaigns identified by means of our clustering. As described in Section 2.4, we performed the Intra-Fingerprint evaluation by manually inspecting at least 3 random screenshot images for 345 random fingerprints and assigned a campaign description to each fingerprint. This way, we made sure that there is no collision of fingerprint hashes which would break our similarity assumptions and we achieve Intra-Fingerprint Coherence.

3.1 Clustering evaluation

In the next step, we aim at evaluating how well our approach generalizes. We build a subset F_m consisting of the 345 labeled fingerprints plus 700 most prevalent non-labeled fingerprints (measured by number of screenshot images per fingerprint) and clustered this subset. The subset F_m represents 187,560 screenshots and samples, respectively. Our goal is to verify the clustering by checking if the non-labeled fingerprints correspond to the labeled fingerprints of the same cluster. Therefore, for each resulting cluster, we inspect at least three of the corresponding screenshot images manually, to see if they relate to the assigned cluster's campaign.

However, before evaluation, the optimum cut-off threshold has to be determined. We use the metrics *precision* and *recall*, widely used for unsupervised machine learning techniques, to measure the clustering accuracy. More precisely, *precision* represents how well our clustering separates fin-

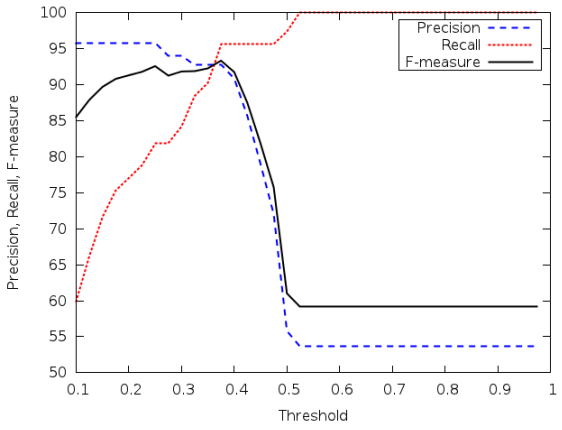


Figure 4: Precision, recall and F-measure ($\beta = 1/2$) evaluation of the clustering threshold

gerprints of different campaigns. *Recall* denotes whether all fingerprints of a certain campaign or family are grouped into the same cluster. In our case, precision is more important than recall, i.e., it is more important to separate fingerprints of two campaigns than to have all fingerprints of one campaign in only one single cluster. In other words, for example, we prefer to have two clusters for one family (or campaign) over aggregating multiple families (or campaigns) into one cluster. For the reasons mentioned in Section 2.5, such as different views of the rogue software's user interface, we tolerate multiple clusters per campaign.

Precision and recall are combined into F-measure. We place twice as much emphasis on precision over recall for the reasons outlined above and we formally define the *F-measure* to evaluate our clustering with threshold th and $\beta = 1/2$:

$$\text{F-measure}_{th} = (1+\beta^2) \cdot \frac{P_{th} \cdot R_{th}}{\beta^2 \cdot P_{th} + R_{th}} \quad (1)$$

Using the labeled fingerprints, we estimate the optimum cut-off threshold by clustering with cut-off thresholds in the range 0.1 to 1.0 and a step size of 0.025. As a result, the best cut-off threshold is determined as 0.375 at a weighted F-measure of 93,32%. Figure 4 shows the precision, recall and F-measure ($\beta = 1/2$) over the range of evaluated thresholds.

The clustering of the set F_m with cut-off threshold 0.375 results in 51 clusters. 10 of these 51 clusters did not have any labeled fingerprints in the same cluster and were manually inspected. Of the 10 unlabeled clusters, 5 clusters represent previously unseen Fake A/V and ransomware campaigns, shown in Table 2. The remaining 5 unlabeled clusters displayed distinct user interfaces of a crack program, Firefox, Windows Photo Viewer, Media Finder Installer and the Run As dialog, waiting for the user to enter Administrator credentials.

For those clusters that have at least one labeled instance, we assign the label of the first labeled instance to the whole cluster. Note that none of the clusters with labeled instances had more than one distinct label, i.e., no cluster contained conflicts among labeled instances. In addition, for each cluster, we inspect three of the corresponding screenshot images manually, to verify that they relate to the assigned cluster's

Perceptual Hash Label	MS A/V Family
Fake A/V and Rogue	
Cloud Protection	undetected
Antivirus Live	FakeSpyprot
Undecipherable	undetected
Ransomware	
GEMA Blocked	undetected
Gendarmerie nationale (FR)	undetected

Table 2: Previously unseen campaigns and, if available, Microsoft A/V Family Labels

Campaign Label	#MD5	no AV	Rate
Cloud Protection	737	15	97.96%
FakeRean:InternetSecurity	323	80	75.23%
FakeRean:SpywareProtect.	608	352	42.11%
Winwebsec:SmartFortress	2656	2367	10.88%
Winwebsec:SmartProtect.	139	83	40.29%
Winwebsec:SystemTool	401	175	56.36%

Table 3: A/V Detection of six Fake A/V campaigns

campaign label. In all cases, the cluster assigned the correct campaign label to the previously unlabeled images.

To sum up, the clustering phase successfully grouped the set F_m consisting of 700 unlabeled fingerprints and 345 labeled fingerprints, and revealed five previously unseen campaigns. Of these five campaigns, only one (Antivirus Live) was detected by Antivirus.

Figure 5 shows the dendrogram of the clustering of F_m, with the campaign label for the cluster (black font color). Red font color denotes prevalent non-rogue software clusters such as those displaying an error message caused due to a malformed malware sample, missing libraries (e.g., DotNET) or language support (e.g., Chinese, Japanese and Russian), runtime errors or bluescreen crashes. Clusters labeled as "Blank screen" contain images that did not contain any significant foreground application window, but exhibit a variety of fingerprints because new desktop links have been added or the desktop link icons have been rearranged, resulting in different fingerprints. Blue font color denotes clusters that displayed malware which is not primarily considered Fake A/V or ransomware such as the Hotbar/Clickpotato downloader or installers for various other software.

3.2 Antivirus Detection

Manual inspection revealed that many of the samples exhibit low A/V detection rates. Based on six well-known campaigns (Table 3), we computed the A/V detection rate of the samples per campaign. For each sample, we queried VirusTotal for the associated A/V labels when we received the sample. Typically, samples were already known by VirusTotal when we queried, since we receive samples with a delay of up to one day from our sample providers. Note that even if a sample is known by VirusTotal, this does not imply that at least one A/V vendor detects it. Table 3 shows the number of MD5-distinct samples per campaign, as well as the number of MD5-distinct samples without any A/V label. The detection rate is measured as number of distinct sample MD5s with an A/V label over the total number of distinct sample MD5s per campaign.

As shown in Table 3, A/V detection rates vary widely

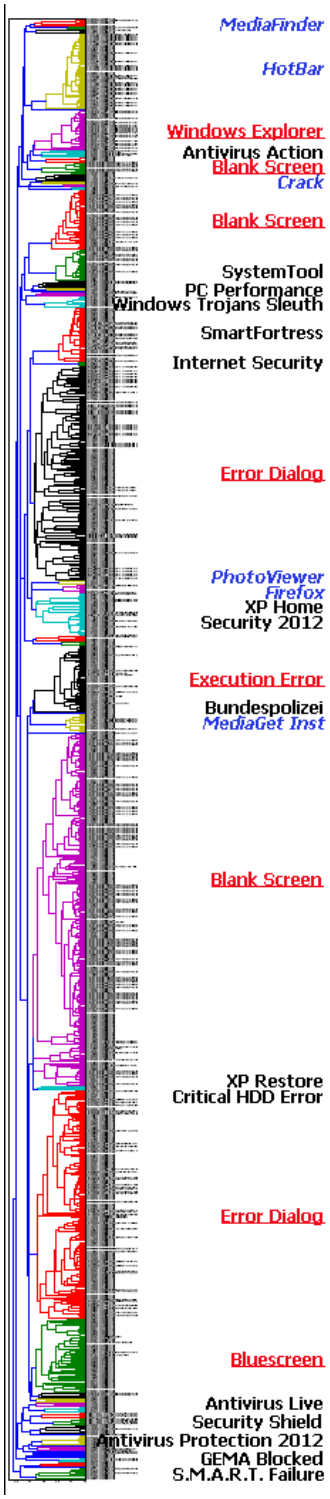


Figure 5: Dendrogram of the clustering of 1045 screenshot fingerprints into 51 clusters with campaign labels for some large clusters. Black font=Fake A/V or ransomware, blue&italic=other software, red&underline=no characteristic rogue appearance.

among Fake A/V campaigns. Of the Smart Fortress campaign, only about 11% of the samples have been detected by at least one A/V vendor. These results confirm the observation of low A/V detection rates by Rajab et al. [9] from August 2009 to January 2010, where A/V detection has ranged between 20% and 60% over all Fake A/V samples.

Our experiments have shown that perceptual clustering of rogue software effectively groups associated samples of visual malware. Furthermore, we see our approach as a starting point towards a complementary classification method which, given suspected malware sample screenshots, recognizes campaigns of rogue visual malware.

3.3 Performance

We implemented the computation of the perceptual hash in C++ and the clustering in Python. Albeit not specifically tailored for high performance, this evaluation will give a rough impression of the processing speed of our screenshot clustering approach. The perceptual hash fingerprints were computed for all screenshot images and stored in a PostgreSQL database. Per 10,000 images, the perceptual hash computation takes 20.13 minutes on a single core, including opening and reading the uncompressed image file from disk. This equals to ca. 120 ms per image.

For clustering performance evaluation, we measured the time required to cluster the full set of 17,767 perceptual fingerprints which relate to 213,671 executed malware samples with the parameters determined in Section 3.1. In total, without any performance improvements, the clustering of the 17,767 fingerprints takes just over 10 minutes on a commodity computer hardware. All in all, if used in practice, the clustering is fast enough to allow for periodic re-clustering of the whole set of screenshots, for example once a day.

4. MONETIZATION AND LOCALIZATION

While the previous sections outlined our methodology, we will make use of the results and shed light on the monetization and localization methods.

4.1 Ransomware Campaigns

In the following, we restrict ourselves to the four ransomware campaigns. The results are summarized in Table 4. The four ransomware campaigns prevent the user from accessing its computer and motivate the user to pay in order to unlock the computer. Three of the four campaigns provided the user an input field for either ukash or paysafecard codes, while the German GEMA ransomware campaign provided only paysafecard as payment method. To sum up, in all cases, the campaigns' monetization is backed up by pre-paid voucher-like services. We subsume these as prepay payment method because in all cases the user has to buy a voucher before the equivalent amount can be spent by providing the voucher code. The use of prepay methods exposes a significant difference to the payment as can be observed with Fake A/V campaigns which typically offer credit card payment. Stone-Gross et al. [12] show that Fake A/V firms even provide refunds in order to avoid credit card chargebacks, because these chargebacks and associated consumer complaints might lead to a cancellation by the credit card payment processor. From the perspective of ransomware miscreants, we speculate several advantages of these prepay payment methods over credit card payment. First, prepay payment avoids to be dependent on credit card payment

processors which are required to act against fraud. Second, while Fake A/V may be doubtful, ransomware is clearly considered fraud in legislation, which might even prevent ransomware campaigns from finding any cooperating credit card payment processors.

While all samples were executed on a virtual machine with the OS localized to Germany and an IP address of the German DFN network (AS 680), the Gendarmerie campaign only provided French texts in the user interface. Even when executing this sample without Internet connectivity, the user interface remains in French.

In case of the Bundespolizei and the Windows Security Center campaigns, a public GeoIP service is used to map the IP address of the infected computer to ISP, country and city. The external IP address, city, region and ISP's AS name are then displayed in the user interface. However, the sample did not adapt the user interface language when confronted with a manipulated geolocation response. In addition, successful installs of the Bundespolizei samples are reported to a C&C server using a plaintext HTTP GET request.

For the GEMA campaign, we observed that the user interface contents consists of HTML with all text and images being server-side generated and transferred via plaintext HTTP. This way, the localization of the campaign might be adapted based on the origin of the HTTP requests.

4.2 Fake A/V Campaigns

We analyzed 14 Fake A/V campaigns in order to shed light on their payment process as well as localization. The results are summarized in Table 5. While all of the ransomware campaigns rely on prepay payment methods, 6 of the 14 Fake A/V campaigns provided credit card payment. For 8 of the 14 Fake A/V campaigns, the payment process failed because their payment servers were no longer reachable, and thus we could not determine the payment method. However, the fact that we could not find one Fake A/V campaign using prepay payment supports our observation that there is a distinction in payment methods between ransomware and Fake A/V software. For ethical reasons, we did not perform any payments and judged on the accepted credit cards only based on the user interfaces. When turning to the amounts of the Fake A/V programs, values range from $50 to $90 USD, depending on supposedly different versions of the program.

Interestingly, while the ransomware campaigns had user interface texts translated to the locale's language, only one of the 14 Fake A/V campaigns of Table 5 exhibits user interface texts matching the locale of the OS (German), all others of the Fake A/V campaigns were only in English. In addition, all amounts were given in US dollars for the Fake A/V campaigns. In contrast, all ransomware campaigns had their amounts adapted to the locale's currency (Euro).

5. LIMITATIONS

Although our approach is based on the inherent property of rogue software to display a user interface, as always, there are some limitations and room for evasion. Targeting the image processing part, i.e., the computation of the perceptual hash function, malware authors could add random noise to the user interface, so that the resulting screenshots differ widely among samples of one campaign. However, since our perceptual hash function depends on low-frequency coordinates, random noise which results in a change in high frequencies will not significantly change the perceptual hash value. Another line of evasion lies in the resemblance of rogue software user interface with that of legitimate software. So far, as long as user interfaces of rogue software differ from those interfaces of legitimate software, our approach can possibly detect and exploit exactly this difference. If Fake A/V appearance exhibits the same user interface as one of the legitimate Antivirus programs, our approach might fail. However, at some point, Fake A/V will always have to provide some kind of payment user interface which could still be used to separate from legitimate software.

6. RELATED WORK

While Fake A/V software has been studied before, e.g., in [4, 5, 9, 12], to the best of our knowledge, we are the first to propose a clustering and classification approach that targets the visual appearance of rogue software user interfaces. Thus, we not only cover Fake A/V, but provide a much broader approach of visual malware, including ransomware. Furthermore, our approach serves as a means of classifying malware executions of a dynamic malware analysis environment and allows to identify and classify failed executions such as bluescreen crashes, error dialogs or lack of visual appearance (silent background malware).

Related work can be grouped into three trails of research. The technical aspects, such as the distribution and infrastructure of Fake A/V malware has been analyzed by Cova et al. [4], Rajab et al. [9] as well as Komili et al. [5], e.g., by measuring the lifetime and distribution of domains involved in Fake A/V software or analyzing the corresponding server geolocation, DNS and AS information. Especially two results of [9] inspired the development of our visual clustering approach. First, the infrastructure of Fake A/V software is volatile. Throughout the measurement period of one year, the median lifetime of Fake A/V domains dropped to below one hour [9]. Thus, relying on network properties such as domains and IP addresses in order to detect Fake A/V software becomes a tremendous effort. Second, Fake A/V malware suffers from decreasing A/V detection rates, even as low as 20% [9]. Our measurement confirms this, showing that in case of the SmartFortress campaign even only ca. 11% of the samples were detected by A/V (Table 3). This demands for a different detection approach such as ours by exploiting the visual appearance of rogue software.

The second trail of research deals with economical aspects of Fake A/V software. Stone-Gross et al. [12] show that more than 130 million dollars have been earned in three monitored Fake A/V businesses. We complement existing research by our analysis of ransomware payment properties and by discovering a shift towards prepay payment methods, such as ukash and paysafecard, in all four ransomware campaigns.

The third area of research covers clustering approaches using behavioral features of executed malware samples. Perdisci et al. [8] develop signatures for characteristic pattern elements in HTTP requests by means of clustering. Similarly, Rieck et al. [11] propose a system to detect malware by models of recurring network traffic payload invariants. Bayer et al. [1] provide a broad overview over malware behavior using clustering to avoid skew by aggressively polymorphic malware. Our approach complements existing approaches by mapping visual appearance of rogue software to campaign and malware family clusters. Especially if none of the ex-

Campaign	Language	Amount	Payment	Limit
GEMA Blocked	German	50 EUR	p	none
Gendarmerie nationale	French	200 EUR	u+p	3 days
Bundespolizei	German	100 EUR	u+p	1 day
Windows Security Center	German	100 EUR	u+p	1 day

Table 4: Localization and monetization methods of four ransomware campaigns; u=ukash, p=paysafecard

Campaign	Language	Amounts	Payment
Antivirus Action	English	$60	n/a
Antivirus Live	English	n/a	n/a
Antivirus Protection	English	3 M: $49.45, 6 M: $59.95, LL: $69.95	n/a
Internet Security	English	n/a	n/a
Cloud Protection	English	$52	VISA / MC
MS Security Essentials	English	$50	n/a
PC Performance	English	$74.95 (light), $84.95 (Prof)	VISA / MC
Personal Shield	English	$1.50 activation + 1 Y: $59.90, 2 Y: $69.95, LL: $83	VISA / MC
Smart Fortress	English	1 Y: $59.95, 2 Y: $69.95, LL: $89.95	VISA / MC
Smart Protection	English	1 Y: $59.95, 2 Y: $69.95, LL: $89.95	VISA / MC
Smart Repair	English	$84.50	VISA / MC
Spyware Protection	English	$60	n/a
XP Antispyware	German	1 Y: $59.95, 2 Y: $69.95, LL: $79.95 + $19.95 Phone Support 24/7	n/a
XP Home Security	English	1 Y: $59.95, 2 Y: $69.95, LL: $79.95	n/a

Table 5: Localization and monetization methods of 14 Fake A/V campaigns; M=months, Y=years, LL=lifelong; MC=Mastercard; All amounts in USD

isting approaches apply, e.g., if a sample does not exhibit network traffic at all, our screenshot clustering approach can still be used.

7. CONCLUSION

In recent years, rogue software has become one of the most prevalent means of monetization for malware. We propose a new approach to cluster and detect rogue software based on its inherent need to display a user interface. Using a perceptual hash function and a hierarchical clustering on a set of 187,560 screenshot images, we successfully identified 25 campaigns, spanning Fake A/V and ransomware. We observed that especially rogue software suffers from very low Antivirus detection rates. Four of the five previously unseen campaigns have not been detected as malware by the time we received the samples. While malware authors seem to succeed in evading classic antivirus signatures, our approach helps to avoid undetected rogue software. Furthermore, we have shown that our approach scales to a large set of samples, effectively analyzing 213,671 screenshot images.

Using the results of our clustering approach, we show that the two prevalent classes of visual malware, Fake A/V and ransomware, exhibit distinct payment methods. While Fake A/V campaigns seem to favor credit card payment, ransomware programs use prepay methods.

Acknowledgements

We thank Jona Greiwe, the people behind Anubis and the anonymous sample providers for providing malware samples. This work was supported by the German Federal Ministry of Education and Research (Grant 01BY1110, MoBE).

8. REFERENCES

[1] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, E. Kirda, and S. Barbara. Scalable, Behavior-Based Malware Clustering. In *NDSS 2009*.

[2] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. M. Youssef, M. Debbabi, and L. Wang. On the analysis of the zeus botnet crimeware toolkit. In *PST*, 2010.

[3] K. Chiang and L. Lloyd. A case study of the rustock rootkit and spam bot. In *HotBots*, HotBots'07, Berkeley, CA, USA, 2007. USENIX Association.

[4] M. Cova, C. Leita, O. Thonnard, A. D. Keromytis, and M. Dacier. An Analysis of Rogue AV Campaigns. In *RAID*, 2010.

[5] O. Komili, K. Zeeuwen, M. Ripeanu, and K. Beznosov. Stragegies for Monitoring Fake AV Distribution Networks. In *Virus Bulletin*, 2011.

[6] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*. IEEE, 2009.

[7] J. Nazario. BlackEnergy DDoS Bot Analysis. `http://sites.google.com/site/evelynnjou/BlackEnergyDDoSBotAnalysis.pdf`.

[8] R. Perdisci, W. Lee, and N. Feamster. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In *NSDI 2010*.

[9] M. A. Rajab, L. Ballard, P. Mavrommatis, N. Provos, and X. Zhao. The Nocebo Effect on theWeb: An Analysis of Fake Anti-Virus Distribution. In *USENIX LEET*, 2010.

[10] M. Riccardi, D. Oro, J. Luna, M. Cremonini, and M. Vilanova. A Framework For Financial Botnet Analysis. In *eCrime Researchers Summit*, 2010.

[11] K. Rieck, G. Schwenk, T. Limmer, T. Holz, and P. Laskov. Botzilla: Detecting the "Phoning Home" of Malicious Software. In *SAC 2010*.

[12] B. Stone-Gross, R. Abman, R. A. Kemmerer, C. Kruegel, D. G. Steigerwald, and G. Vigna. The Underground Economy of Fake Antivirus Software. In *WEIS*, 2011.

[13] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna. The Underground Economy of Spam: A Botmaster's Perspective of Coordinating Large-Scale Spam Campaigns. In *USENIX LEET*, 2011.

[14] C. Zauner. Implementation and Benchmarking of Perceptual Hash Functions. Master's thesis, 2010.