D^2PI: Identifying Malware through Deep Packet Inspection with Deep Learning

Ronald Cheng, rscheng@cs.umd.edu
Gavin Watson, gkwatson@cs.umd.edu
University of Maryland, College Park

Abstract - Malicious contents' main means of distribution are through the Internet. Common and effective security measures that detect and can signal the prevention of malicious content propagation are Network Intrusion Detection Systems (NIDS) such as Snort. In this paper, we propose D^2PI, a novel way of identifying network traffic with malware by performing deep packet inspection with a Convolutional Neural Network. D^2PI is a neural network architecture that uses character embeddings followed by deep convolutional networks trained upon the payloads of packets from the dataset and functions as an NIDS. In an evaluation that uses a dataset of 127 distinct malwares and a sampling of over 16GB of benign traffic, our D^2PI outperforms the popular open source intrusion detection system Snort by more than 17% in F1 score. Furthermore, D^2PI should lend itself well to integration with other NIDS techniques or systems to further improve accuracy and might be more effective at identifying zero-day attacks than current state of the art commercial NIDS.

Introduction

Network traffic classification is an important task in modern communication networks due to the rapid growth of high throughput, traffic demands, and the security concerns that arises with network traffic. Attackers take advantage of this growing Internet connectivity to access computers over the network to do things like encrypt important data (ransomware), install backdoors (trojan horses), and send self-propagating programs that infect more machines (worms). Static analysis of malware in machines often occurs too late; the malware has had time to execute arbitrary code already [1]. An area of security research that arose to detect such attacks in real time by analyzing the traffic itself is called network intrusion detection (NID).

A network intrusion detection system (NIDS) is composed of software and/or hardware designed to detect unwanted attempts to access, manipulate, and/or disable computer systems. An NIDS is used to detect several types of malicious behaviors that can compromise the security and trust of a computer's system. These threats are various, and include network attacks against vulnerable services, data driven attacks on applications, and host based attacks such as privilege escalation, unauthorized accesses, and malware (viruses, worms) [2][3].

These detection systems can be categorized into two methods: (1) Auditing packet information and signatures available to classify traffic, and (2) Observing traffic directly using packet filters or other detection schemes. These detection systems are usually equipped with static analyzers and feature extractors that leverage things like deep packet inspection, which examines packet payloads [4][5]. They use malware scanners that dynamically use run-time information in memory to identify behavior that appears malicious or statically extract features from disk of files that may be malware [6].

However, these approaches defend best when malware signatures are explicitly known, so they are generally ineffective against zero-day attacks and amorphous malwares. They also require great investments of security expert time to identify features that effectively classify the types of attacks that are known. It is not unusual for these systems to leverage hundreds of hand-selected features, and new ones often have to be defined to deal with emerging malware vectors. We propose a new approach of identifying malware by using a Convolutional Neural Network (CNN) that does

not require prior knowledge of malware operations or extracting features of network traffic because it automatically learns features. This is not only a more economical approach to intrusion detection, but it is also likely to be more robust in detecting existing malware that is altered to avoid detection and easier to adapt to emerging threats because it could be retrained with the inclusion of newly discovered malwares. This does require keeping the entire training set, identifying new malware, and letting the system automatically retrain on the the new extended set over at least several hours if not days depending on the set size.

Related Work

Traditional defense against malware that propagates through the internet uses techniques such as signature detection, feature extraction, and deep packet inspection. Others inspect the behavior of possible malware binaries with static analysis [7]. The NIDS of traditional approaches emphasize heavily upon low rate of falsely identifying benign traffic as malicious, but the tradeoff is that the system only alerts on malicious traffic that the system is very convinced is malicious. This is ideal in a system where falsely identifying benign content and being aggressive in identifying malicious content might hinder productivity and annoy users to ignore the alerts. Some work has been done in terms of machine learning on features extracted for intrusion detection systems [8] [9], but the this work still fundamentally relies on feature extraction and only leverages machine learning as an automatic weighting vector for features. We are, to our best knowledge, the first to apply machine learning concepts into separating malicious and benign traffic without feature extraction.

The most relevant work to ours was done by Lotfollahi et al. [10] [11] who classified types of encrypted traffic such as VPN vs non-VPN traffic by using a similar CNN structure and Saxe et al. [12] who classified malicious and benign URLs with another similar CNN. The former concentrate more on blocking types of traffic unwanted by company policy rather than identify security risks, and the latter are specifically concerned with URLs. Our focus is on intrusion detection, but we take great confidence from the previous success of CNNs on these other tasks. Our network's embedding layer is not as sophisticated as these networks, however, which may be a future source of improvement.

Threat Model and Goals

The attacker has total control over the network packets transmitted to the user's computer, which is protected by our CNN firewall. The attacker does not have control over this firewall or the user's computer in any way. However, the user will receive any packet that the attacker wants the user to receive. In particular, we have chosen to focus on malicious binaries sent over TCP by the attacker in our analysis, but we believe that our approach could be extended to other types of attacks like SQL injection if trained with appropriate datasets.

Our goal is to identify malware and flag a a session as suspicious or malicious in real time after being trained on a sufficiently adequate dataset. We should then be able to detect malware with good accuracy without being trained on or aware of the specific attack (which may be a zero-day attack). Although our classifier has not been used on a live system at the time of this paper, we believe that our analysis shows that it could perform well in such an environment.

Goals that are out of scope of what we are accomplishing are identifying behavior of what the attacker is doing, detecting botnet traffic, DOS attacks, scanning attacks, and anything that doesn't rely on transferring packet payloads.

Solution Overview

Our proposed solution to this malware classifying problem is a neural network architecture that is trained on the payloads of downloads of known malware executables and the payloads of general, known benign packets. This network leverages techniques that have been used in natural language processing like character

embeddings followed by deep convolutional networks because of our underlying assumption that executable code and other attacks have similar feature constructs to natural languages. It is trained and predicts on sessions of internet traffic between the host and clients and functions as an Intrusion Detection System (IDS). That is to say that it does not take any security action upon predicting that a client has sent a malicious executable; it defers any such action to an underlying policy that is determined by the host's administrator.

As control group to evaluate our classifier against a metric, we used Snort, for which we also go into detail in the section below. We evaluated the accuracy rate of our classifier using anonymized datasets that we both got from public sources and created by ourselves. We open these dataset to the public for interested parties. Our evaluation plan is also gone through in detail below.

D^{2}PI Network Design

Our neural model's main workhorse called the a payload classifier, which consists of a character embedding layer that's followed by four convolutional and pooling layers that are followed by a two classification softmax layer as shown in Figure 1. They were all programmed in Python through the Keras library on top of Tensorflow [13][14]. Most of the parameters like number of layers, sliding window size, number of kernels, activation functions, and pooling layer sizes are based on the designs of previous effective works or initial testing with a small dataset, and there may well be room for improvement by sampling different combination of these.

The embedding layer is a pre-trained character-character matrix of 128x128 values that encode a vector of the context for each of the possible 128 ASCII characters. All of the training payloads are run through with a sliding context window of three characters, and the vector representing the middle character is given plus one in weight at the index of the other characters seen in the window. Each vector in the embedding matrix is then normalized to a magnitude of one. This is a relatively simple embedding model that could well be improved by switching to a word2vec model that operated on characters instead of words [15]. When the payload classifier is actually used after this pre-training, a payload is converted into a 1500x128 matrix before being input to the convolutional layers by converting

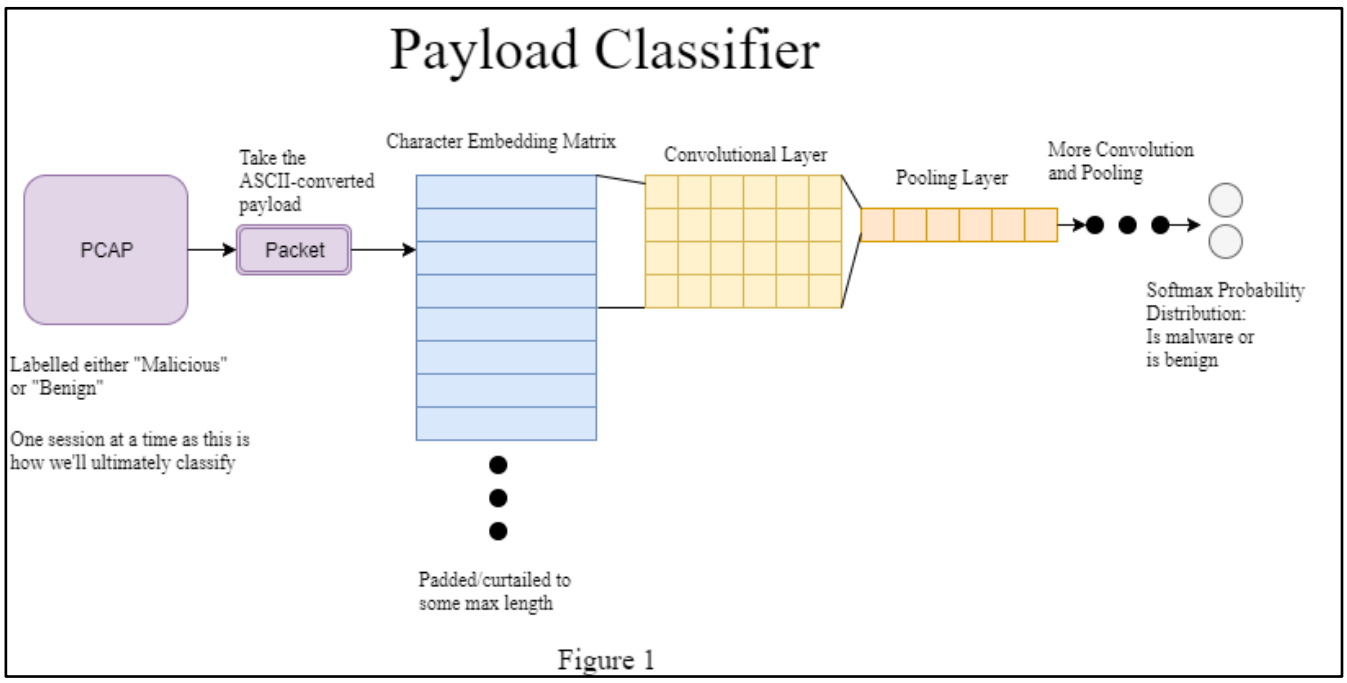

Figure 1

each character into its learned embedding and appending all of the embeddings onto each other in the same order as the payload. If payloads are larger or smaller than 1500 characters, this matrix is curtailed or padded to zeros. They should not be longer than this by convention according to Lotfollahi et al., so this should not highly affect results [10].

The kernels in the convolutional layers each span four entire character vectors and use rectified linear unit activation. The four layers have 32, 64, 128, and 128 kernels respectively, and each is followed by a max pooling layer over four entire character vectors, except for the last one that pools over sixteen vectors. The last pooling layer is then connected to a sixteen node rectified linear unit layer, which is then connected to a two node softmax layer. The output of this layer corresponds to a percentage of confidence that input payload was malicious or benign.

Building upon this payload classifier, we have a slightly larger network to actually train and predict things for sessions of multiple packets although the payload classifier has to be trained before this network can be trained. This session classifier inputs up to 10,000 packets at a time from the same session, passes all the non-empty TCP packets through the payload classifier to get predictions, and then aggregates these predictions into a malware score by adding all of the malicious prediction percentages and subtracting all of the benign prediction percentages. What we end up with is a single number score in the range (-10,000, 10,000) that corresponds to how strong a prediction the the network has on the session as to whether it is malicious or not with positive scores meaning malicious, negative scores meaning benign, and scores near zero either corresponding to unseen data, low confidence predictions, or sessions without many packets. This score, however, is not directly used to predict; we run the all the scores of the same sessions on which we trained the payload classifier through another single perceptron that learns the decision boundary for malware and benign. In our results, this perceptron learns a boundary around zero and does not help more than a manual cutoff, but it is a convenient placeholder for possibly combining our payload classifier with other features extracted from packets in the future to create an even more robust predictor.

For our test runs, we predicted sessions that were larger than 10,000 packets in non-overlapping increments of 10,000 packets, and we took the increment that had the highest malware score and therefore looked the most malicious as the de facto score for the session because we assume that if a session has 10,000 packets anywhere in it that look like malware, it's probably malicious. It is possible we could have gotten better malicious predictions with a sliding window, over the sessions' packets, but we expect this would be prohibitively expensive in practice. As it was, our system already took around 2.2 minutes to predict on 10,000 packets, which is slightly slower than Snort's 1.6 minutes in our tests.

It is worth noting that this same architecture could define an anomalous or unsure prediction zone around the malware score of zero, but we have not qualitatively analyzed such a zone at this time.

Baseline: Snort

Our classifier was compared to Snort, the most popular open source Intrusion Detection System that classifies traffic based on rules and signatures and is updated by the SourceFire team monthly [16]. We set up Snort in Network Intrusion Detection System Mode with live capture mode off, and fed it pcaps from our datasets that were stored in a local directory instead of analyzing live traffic. We also equipped Snort with a wealth of additional detection schemes from Emerging Threats that has over 2,100,000 signature IDs, and Bro's Team Cymru's malware hash registry. This allows us to further emulate commercial firewalls in practice.

Datasets

As with any machine learning applications, the datasets used to train and predict are very important in evaluating the performance of the application. For malicious traffic pcaps, we pulled 127 distinct samples of malware executables from an online repository known as Contagio, which accounted for nearly half a gigabyte in traffic [17]. We limited the problem space to executable malware, and sampled the malware families that we believe are a good representation of the malicious content rampant on the Internet as of now. These include: trojan programs, worms, and exploit kits among others.

For benign traffic pcaps we sampled the ISCX IDS 2012 benign traffic pcaps created by Canadian Institute for Cybersecurity of UNB [18]. This is a simulated dataset that is intended to mimic real-world traffic without the need for anonymization and is derived from real traffic for the HTTP, SMTP, SSH, IMAP, POP3, and FTP protocols. In particular, we took their first day of general benign traffic, separated it into session pcaps, took the 2000 largest sessions, and then randomly selected 150 of these sessions to reduce our data to just over 800MB from an overwhelming ~16GB.

In addition, we created our own explicitly benign executable pcaps by downloading 40 popular pieces of software (~2GB) and monitoring with wireshark. We made this set in case it better represented data that a payload classifier would have a harder time separating than the general case since the malicious set is also executables. The softwares that we downloaded were about 36% .dmg format for Mac and about 64% .exe for Windows. The types of software ranged from small tools such as winzip to IDEs like eclipse. We tried to find benign versions of the trojan softwares that were present in Contagio such as bitcoin miners, game engines, and wordpress plugins. We then ran them through VirusTotal, which checks each for malware signatures with multiple anti-virus softwares to make sure that they were benign.

All of our dataset pcaps were transmitted over TCP regardless of whether they were benign or malicious. Any other protocols were ignored after passing it through either snort or our classifier. This is ideal since 93% of traffic that goes through common IDS are TCP [2]. This is also essential since, if there were a protocol bias that was not TCP, it would reflect poorly on our classifier.

Evaluation

After initial testing of the percentage of the number of pcaps from which our classifier appeared to learn the most, we settled on randomly selecting 20 malicious pcaps, 10 benign ISCX pcaps, and 6 benign executable pcaps from our datasets for the training set for our classifier. On every run, these pcaps were randomly chosen and our classifier metrics were only taken from the test set, which was all of the pcaps from our datasets that were not used in training. We ran several tests this way and report the findings of our best run although we note that many runs did not work well, which means our network may need stability improvements.

Furthermore, when training our network, we first extracted all of the TCP payloads, randomly dropped as many payloads as we needed to in order to make the malicious and benign sets even, and randomly ordered the remaining payloads such that benign and malicious ones were mixed together and kernels would not converge prematurely to weights that only predicted a few of the packets well.

We evaluated the overall effectiveness of our classifier and Snort with the F1 score metric as given in Equation 1 in which we considered classifying malware correctly as true positives. Our best classifier's and Snort's rates of correct prediction can be seen in Table 1. Our classifier's F1 score was 0.7724 while Snort's was only 0.6003. This is a very promising result for our classifier although we note, as shown in Table 1, that Snort was perfect when predicting benign traffic and our classifier was not. We expect that this is intentional on the part of Snort for usability

$$F1 = \frac{2 * TruePos}{2 * TruePos + FalseNeg + FalsePos}$$

Equation 1

	ISCX Benign	Created Benign	Malicious
Predicted - Benign	131 (93.57%)	34 (100%)	36 (33.64%)
Predicted - Malicious	9 (6.43%)	0 (0%)	71 (66.36%)

Table 1: CNN results

	ISCX Benign	Created Benign	Malicious
Predicted - Benign	140 (100%)	34 (100%)	55 (51.40%)
Predicted - Malicious	0 (0%)	0 (0%)	52 (48.60%)

Table 2: Snort Results

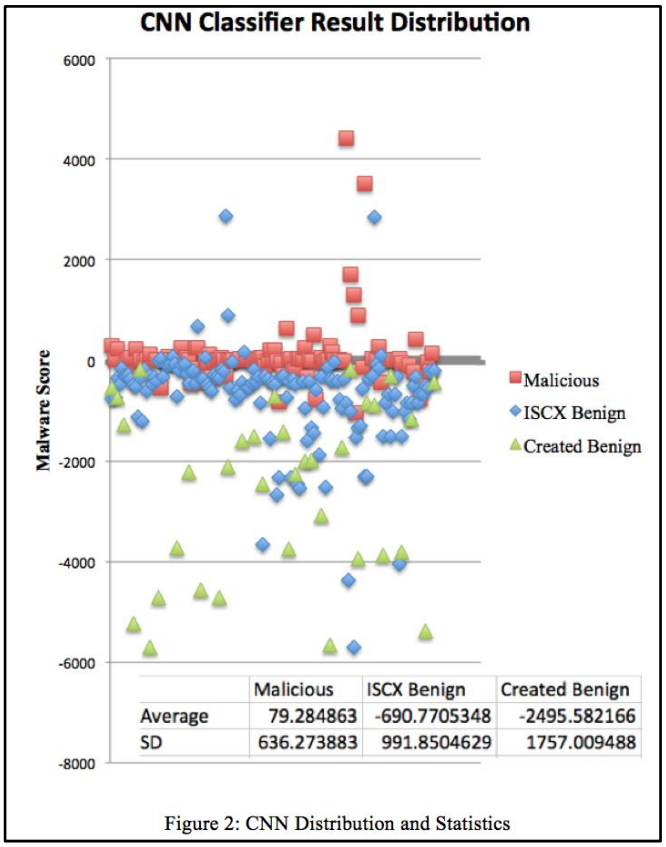

Figure 2: CNN Distribution and Statistics

on real-world traffic, so our classifier may require more work to be usable in the same way.

There did not appear to be any pattern between the malwares that each system predicted correctly. However, as can be seen in the appendix, there was a high discrepancy between the malwares that were predicted correctly between the two systems. So, a simple two step classifier where we call a file malware if either of the systems called it malware would have correctly predicted over 90% of the malware in our data set without predicting the benign files any worse that our classifier did by itself. This epitomizes the possible usefulness of our classifier in a system with multiple classification schemes.

We also clustered the malware scores created by our classifier for every test pcap in Figure 2, and while they do not show quite as much distinction as we might have hoped, there is a pretty clear line of separation for most of the pcaps near 0. More sophisticated variations of our techniques may be able to improve on this separation.

Conclusions and Future Work

While we know from past work that the protocols of internet traffic can be differentiated by similar networks to ours, we believe that our results show that convolutional neural networks can make even more nuanced predictions and identify the differences between things such as malicious and benign files at a respectable rate. This is especially convincing in the reasonable accuracy in Table 1 for the Created Benign and Malicious datasets. Furthermore, our classifier outperforms Snort on our realistic dataset as measured by an F1 score, which implies that it may be more practical as an NIDS, although it is not unilaterally better since it predicts more false negatives.

The most exciting part about this classifier is that it only really addresses one part of the packets that are run through it: the payloads. This is analogous to Deep Packet Inspection, which is only one part of similar classifiers, so other parts of the packets like the IP addresses and metadata could be leveraged yet to make our classifier even more robust.

Other future work that could be done would be to devise an integrated IDS with a voting scheme between Snort and our CNN. This would allow us integrate the pros and cons of both systems, which might greatly decrease the false positives and false negatives of our results. Furthermore, our classifier currently only supports TCP with IPv4 and further work might include UDP and IPv6,

and our classifier might be extendable to work on encrypted protocols since it can automatically build its own features and the similar classifier constructed by Lotfollahi et al. was able to classify on encrypted payloads [10].

Acknowledgments

We thank Professor Levin from University of Maryland CS department for his invaluable mentorship and guidance on this research. Also, the dataset kindly provided to us from Mila Parkour in Vizsec and the researchers at UNB helped us greatly in evaluating our accuracy rate of our classifier. Lastly, thanks to all GradSec classmates who gave us feedback and advice on the way!

References

[1] Y. Ye et al., "Intelligent file scoring system for malware detection from the gray list," in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. 2009, ACM: Paris, France. p. 1385-1394.

[2] A. Shabtai, D. Potashnik, Y. Fledel, R. Moskovitch, and Y. Elovici, "Monitoring, analysis, and filtering system for purifying network traffic of known and unknown malicious content." Security and Communication Networks, 4(8), pp.947-965, 2010.

[3] Z. Wang, "The Applications of Deep Learning on Traffic Identification." https://www.blackhat.com/docs/us-15/materials/us-15-Wang-The-Applications-Of-Deep-Learning-On-Traffic-Identification-wp.pdf [Accessed 12 Dec. 2017], 2017.

[4] S. Sen, O. Spatscheck, D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures" in Proceedings of the 13th international conference on World Wide Web. ACM, 2004: 512-521.

[5] D. Zuev and A.W. Moore, "Traffic classification using a statistical approach" in Passive and Active Network Measurement. Springer Berlin Heidelberg, 2005: 321-324.

[6] N. Idika and A.P. Mathur, "A Survey of Malware Detection Techniques." 2007.

[7] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket", Proceedings 2014 Network and Distributed System Security Symposium, 2014.

[8] V. Paxson, "Bro: a system for detecting network i

[9] F. Střasák and S. Garcia, "Detecting malware even when it is encrypted." Brucon 2017.ntruders in real-time", Computer Networks, vol. 31, no. 23-24, pp. 2435-2463, 1999.

[10] M. Lotfollahi, R. Hossein Zade, M. Jafari Siavoshani and M. Saberian, "Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning", ARXIV, vol. 1709, no. 02656, 2017.

[11] S. Bagui, X. Fang, E. Kalaimannan, S. C. Bagui, J. Sheehan, Comparison of machine-learning algorithms for classification of vpn network traffic flow using time-related features, Journal of Cyber Security Technology (2017) 1–19.

[12] J. Saxe, K. Berlin, "eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys", ARXIV, vol. 1702, no. 08568, 2017.

[13] https://keras.io/

[14] https://www.tensorflow.org/

[15] T. Mikolov, K. Chen, G. Corrado, J. Dean, "Efficient Estimation of Word Representations in Vector Space," ARXIV, vol. 1301, no. 3781, 2013.

[16] https://www.snort.org

[17] http://contagiodump.blogspot.com/2013/04/collection-of-pcap-files-from-malware.html

[18] http://www.unb.ca/cic/datasets/ids.html

Appendix: Malware Predictions by File Name

Malicious Pcap Name	D^2PI prediction	Snort prediction
8202_tbd_ 6D2C12085F0018DAEB9C1A53E53FD4D1	Malicious	Benign
AlienSpyRAT_79E9DD35AEF6558461C4B93CD0C55B76	Malicious	Malicious
AlienspyRAT_DB46ADCFAE462E7C475C171FBE66DF82-WinXP	Malicious	Malicious
AlinaSpark_BE6371B8C90D8EECB749311373CEC0ED	Malicious	Benign
BIN_8202_6d2c12085f0018daeb9c1a53e53fd4d1	Malicious	Benign
BIN_9002_D4ED654BCDA42576FDDFE03361608CAA_2013-01-30	Benign	Benign
BIN_Alurewo_2502edca284bd8bf782a65123a22f9a6	Benign	Malicious
BIN_Andromeda_85F908A5BD0ADA2D72D138E038AECC7D_2013-04	Malicious	Benign
BIN_Bitcoinminer_12E717293715939C5196E604591A97DF-2013-05-12	Malicious	Benign
BIN_ChePro_2A5E5D3C536DA346849750A4B8C8613A-1	Benign	Benign
BIN_Cidox_Nuclear-EK_malware-traff-analysis-blog_2014-08-06	Benign	Malicious
BIN_CitadelPacked_2012-05	Malicious	Benign
BIN_CitadelUnpacked_2012-05	Malicious	Benign
BIN_Cutwail-Pushdo(1)_582DE032477E099EB1024D84C73E98C1	Malicious	Malicious
BIN_Cutwail-Pushdo(2)_582DE032477E099EB1024D84C73E98C1	Malicious	Malicious
BIN_DNSChanger_2011-12	Malicious	Malicious
BIN_DNSWatch_protux_4F8A44EF66384CCFAB737C8D7ADB4BB8_2012-11	Malicious	Benign
BIN_DarknessDDoS_v8g_F03Bc8Dcc090607F38Ffb3A36Ccacf48_2011-01	Malicious	Benign
BIN_Enfal_Lurid_0fb1b0833f723682346041d72ed112f9_2013-01	Malicious	Benign
BIN_GameThief_ECBA0FEB36F9EF975EE96D1694C8164C_2013-03	Malicious	Benign
BIN_Gh0st-gif_f4d4076dff760eb92e4ae559c2dc4525	Malicious	Benign
BIN_Gh0st_variant-v2010_B1D09374006E20FA795B2E70BF566C6D_2012-08	Malicious	Benign
BIN_Googledocs_macadocs_2012-12	Benign	Benign
BIN_Gypthoy_3EE49121300384FF3C82EB9A1F06F288	Malicious	Benign
BIN_Hupigon_8F90057AB244BD8B612CD09F566EAC0C	Malicious	Benign
BIN_IRCbot_c6716a417f82ccedf0f860b735ac0187_2013-04	Malicious	Malicious
BIN_IXESHE_0F88D9B0D237B5FCDC0F985A548254F2-2013-05	Malicious	Malicious
BIN_Imaut_823e9bab188ad8cb30c14adc7e67066d	Malicious	Malicious
BIN_Kelihos_aka_Nap_0feaaa4adc31728e54b006ab9a7e6afa	Malicious	Benign
BIN_Kuluoz-Asprox_9F842AD20C50AD1AAB41F20B321BF84B	Malicious	Malicious
BIN_LURK_AF4E8D4BE4481D0420CCF1C00792F484_20120-10	Malicious	Benign
BIN_Lader-dlGameoverZeus_12cfe1caa12991102d79a366d3aa79e9	Benign	Malicious
BIN_LetsGo_yahoosb_b21ba443726385c11802a8ad731771c0_2011-07-19	Malicious	Benign
BIN_Likseput_E019E37F19040059AB5662563F06B609_2012-10	Malicious	Benign
BIN_LoadMoney_MailRu_dl_4e801b46068b31b82dac65885a58ed9e_2013-04	Benign	Malicious
BIN_MatsnuMBRwiping_1B2D2A4B97C7C2727D571BBF9376F54F	Benign	Benign
BIN_Mediana_0AE47E3261EA0A2DBCE471B28DFFE007_2012-10	Malicious	Benign
BIN_NJRat-BackdoorLV_6fd868e68037040c94215566852230ab_CNtiananmensquare	Malicious	Benign
BIN_Nettravler_1f26e5f9b44c28b37b6cd13283838366	Malicious	Malicious
BIN_Nitedrem_508af8c499102ad2ebc1a83fdbcefecb	Benign	Malicious
BIN_Nocpos_3def6f8d1b709e61b83e9a697d64e129	Malicious	Malicious
BIN_PUP_Selfinstall_9f26de41c7520929ea4f4f7e61abe1a6	Benign	Benign

BIN_PlugX_2ff2d518313475a612f095dd863c8aea	Malicious	Benign
BIN_Ponyloader-Zeus_B10393BE747143F3B4622E9E5277FFCE	Benign	Malicious
BIN_PowerLoader_4497A231DA9BD0EEA327DDEC4B31DA12_2013-05	Benign	Malicious
BIN_Prosti-Screenblaze_00001ffe4e2c3218db5eecfd16b97a9f	Malicious	Benign
BIN_Reedum_0ca4f93a848cf01348336a8c6ff22daf_2013-03	Malicious	Benign
BIN_RssFeeder_68EE5FDA371E4AC48DAD7FCB2C94BAC7-2012-06	Malicious	Malicious
BIN_Sality_03fa78bd6c71d76a50e63ab0b9a4505f	Malicious	Malicious
BIN_Sanny-Daws_338D0B855421867732E05399A2D56670_2012-10	Benign	Benign
BIN_Scudy_5c085d004270abbc6a21151e60a984d1	Malicious	Malicious
BIN_SpyEye_2010-02	Benign	Malicious
BIN_Stabuniq_F31B797831B36A4877AA0FD173A7A4A2_2012-12	Malicious	Malicious
BIN_Taidoor_40D79D1120638688AC7D9497CC819462_2012-10	Malicious	Malicious
BIN_Taleret.E_5328cfcb46ef18ecf7ba0d21a7adc02c	Malicious	Benign
BIN_Tapaoux_60AF79FB0BD2C9F33375035609C931CB_winver_2011-08-23	Malicious	Benign
BIN_Tbot_23AAB9C1C462F3FDFDDD98181E963230_2012-12	Benign	Benign
BIN_Tbot_2E1814CCCF0C3BB2CC32E0A0671C0891_2012-12	Malicious	Benign
BIN_Tbot_5375FB5E867680FFB8E72D29DB9ABBD5_2012-12	Benign	Benign
BIN_Tbot_A0552D1BC1A4897141CFA56F75C04857_2012-12	Benign	Benign
BIN_Tinba_2012-06	Malicious	Benign
BIN_TrojanPage_86893886C7CBC7310F7675F4EFDE0A29	Benign	Benign
BIN_Twerket_a27721f3b9566601030daab58c092c14	Malicious	Benign
BIN_UStealD_2b796f11f15e8c73f8f69180cf74b39d	Malicious	Benign
BIN_Vobfus_634AA845F5B0B519B6D8A8670B994906_2012-12	Malicious	Malicious
BIN_Wauchos_Zbot_0d8d7a8074ee36a626d086f02490aaab	Malicious	Benign
BIN_Win.Trojan.Waski_document_234787_pdf.exe_ee299b606ea2165a88a06c3347c0319b	Benign	Malicious
BIN_Wordpress_Mutopy_Symmi_20A6EBF61243B760DD65F897236B6AD3-DeepEndR	Malicious	Malicious
BIN_Wordpress_Mutopy_Symmi_20A6EBF61243B760DD65F897236B6AD3-ShortRun	Malicious	Benign
BIN_Xpaj_2012-05	Malicious	Malicious
BIN_ZeroAccess_3169969E91F5FE5446909BBAB6E14D5D_2012-10	Malicious	Malicious
BIN_ZeroAccess_Sirefef_29A35124ABEAD63CD8DB2BBB469CBC7A_2013-05	Benign	Malicious
BIN_ZeusGameover_2012-02	Benign	Malicious
BIN_Zeus_b1551c676a54e9127cd0e7ea283b92cc-2012-04	Benign	Benign
BIN_Zeus_outbound_1-25050_2014-10-08-phishing-malware-analysis-from-malwr.com	Benign	Benign
BIN_dirtjumper_2011-10	Malicious	Malicious
BIN_fd0ff4992247bbcc2bde6379e10c1499_Hyteod	Malicious	Benign
BIN_njRAT-Backdoor	Malicious	Benign
BIN_torpigminiloader_011C1CA6030EE091CE7C20CD3AAECFA0	Benign	Malicious
BIN_torpigminiloader_C3366B6006ACC1F8DF875EAA114796F0	Benign	Malicious
BitcoinMiner_F865C199024105A2FFDF5FA98F391D74	Malicious	Benign
Citadel_3D6046E1218FB525805E5D8FDC605361-2013-04	Malicious	Benign
Darkcomet_DC98ABBA995771480AECF4769A88756E	Malicious	Benign
EK_Blackholev1_2012-03	Malicious	Malicious
EK_Blackholev1_2012-08	Benign	Benign

EK_Blackholev2_2012-09	Benign	Malicious
EK_CDN-gate_malw-traf-analysis-blog_2014-09-04-Sweet-Orange-EK-traffic	Benign	Benign
EK_Malware_traffic_analysis_blog_2014-05-19-FlashPack-EK-traffic (1)	Malicious	Malicious
EK_Smokekt150(Malwaredontneedcoffee)_2012-09	Benign	Benign
EK_Styx+BIN_Simda_Proxyer_Malware_traffic_analysis_blog_2014-03-15-Styx-EK-traffic	Benign	Malicious
EK_popads_109.236.80.170_2013-08-13	Malicious	Malicious
InvestigationExtraction-RSA_Sality	Benign	Malicious
Kelihos_C94DC5C9BB7B99658C275B7337C64B33	Benign	Malicious
Mswab_Yayih_FD1BE09E499E8E380424B3835FC973A8_2012-03	Malicious	Benign
Netwire_79e6ea386833b2443fe093cf0c1e8c66-network	Malicious	Malicious
OSX_DocksterTrojan	Malicious	Malicious
PassAlert_B4A1368515C6C39ACEF63A4BC368EDB2-2013-05-13	Malicious	Benign
Pony_B5E7CD42B45F8670ADAF96BBCA5AE2D0	Malicious	Malicious
RTF_Mongall_Dropper_Cve-2012-0158_C6F01A6AD70DA7A554D48BDBF7C7E065_2013-01	Malicious	Malicious
Tijcont_845B0945D5FE0E0AAA16234DC21484E0	Benign	Malicious
TinyZBot_96e372dea573714d34e394550059b1d7	Malicious	Malicious
Toopu_26475c32a3f60ab902cdc8ed9102b383	Benign	Malicious
Toopu_57a20c291ac47a75e0274d52a2aab36b	Benign	Malicious
Toopu_f3f087cf7788a9de9557f782ef2882d3	Benign	Malicious
Xinmic_8761F29AF1AE2D6FACD0AE5F487484A5	Malicious	Benign
cryptolocker_9CBB128E8211A7CD00729C159815CB1C	Malicious	Malicious
purplehaze	Malicious	Malicious