

Machine Learning in Cyber-Security - Problems, Challenges and Data Sets

Idan Amit¹, John Matherly², William Hewlett¹, Zhi Xu¹, Yinnon Meshi¹, Yigal Weinberger¹

¹Palo Alto Networks

²Shodan

iamit@paloaltonetworks.com, jmath@shodan.io, whewlett@paloaltonetworks.com,
zxu@paloaltonetworks.com, ymeshi@paloaltonetworks.com, yweinberge@paloaltonetworks.com

Abstract

We present cyber-security problems of high importance. We show that in order to solve these cyber-security problems, one must cope with certain machine learning challenges. We provide novel data sets representing the problems in order to enable the academic community to investigate the problems and suggest methods to cope with the challenges. We also present a method to generate labels via pivoting, providing a solution to common problems of lack of labels in cyber-security.

Introduction

Cyber-security is an important area in which machine learning is becoming increasingly significant. Many machine learning algorithms such as convolutional neural networks (David and Netanyahu 2015), LSTM (Woodbridge et al. 2016) and others (Khorshidpour, Hashemi, and Hamzeh 2017; McLaughlin et al. 2017; Hu and Tan 2017a; Villegas 2017; Patri, Wojnowicz, and Wolff 2017) were applied to cyber-security problems. It is important to note that machine learning in cyber-security is far more than merely applying established machine learning methods to data sets of cyber entities.

Cyber-security involves machine learning challenges that require elegant methodological and theoretical handling. We believe that such challenges are of interest to people with passion for machine learning, without necessarily requiring cyber domain expertise or prior knowledge. The machine learning community is usually unaware of these challenges.

We are not the first to discuss security and AI challenges (Stoica et al. 2017) or alert on the lack of data sets (Kumar, Wicker, and Swann 2017). The novelty of our work is the presentation of new cyber-security problems, the machine learning challenges involved in them and the publication of data sets that enable investigating them. We hope it will lead to new methods in both machine learning and cyber-security.

Cyber-Security Problems

Malware Classification And Detection : Identifying Malicious Programs

Malware is a program or a file that is harmful to a computer system. As part of the arms race, the attacker tries to avoid detection. Some evasion techniques are polymorphism, impersonation, compression and obfuscation (You and Yim 2010). For example, in malware coloring the attacker slightly change the malware, leading to polymorphism (many variants). Since many threat intelligence repositories are based on signatures of the malware (e.g., SHA1, MD5), a slight modification enables to a signature-based detection.

Current detection systems use various algorithms: Naive Bayes Classifier (Luo 2016), SVM (Kuriakose and Vinod 2015), Random Forest (Hu and Tan 2017b), DNN (Xie, Girshick, and Farhadi 2015), CNN (David and Netanyahu 2015) and LSTM (Woodbridge et al. 2016; Saxe and Berlin 2017).

Labeling Malware Via Opertor Domain Pivoting

Historically, malware classification was based on signatures and domain experts. However, domain experts are limited in the number of cases they analyze, and signatures lead to labeling errors. The large number of malware and their population rapid growth make the problem even more severe. For these reasons, we have constructed a malware data set and grouped malware contacting the same malicious site.

Let m_1, m_2 be malware.

Let $OperatorDomains(m)$ be the unique domains with which the malware communicates.

The specific definition of $OperatorDomains(m)$ is use case specific and requires domain knowledge. For example, many malware communicate with benign domains (e.g., google.com) in order to check Internet connectivity. So, communication with Google is not an indication of the operator or maliciousness. A good definition of $OperatorDomains(m)$ should focus in malicious (or at least not known benign domains) that are not communicating with too many files.

If $OperatorDomains(m_1) \cap OperatorDomains(m_2) \neq \emptyset$, then m_1, m_2 belong

to the same operator.

Hence, given a data set D of malware and a $OperatorDomains(m)$ function we label $\forall m_1, m_2 \in D, (m_1, m_2, OperatorDomains(m_1) \cap OperatorDomains(m_2) \neq \emptyset)$.

In some uses it is more appropriate to work with a multi-class data set, assigning to each pair an indication of the class the pair belongs to (e.g., the mutual domain).

Thanks to that, we have a large number of labels, agnostic to the content, that can be used to build and evaluate content-based malware classification models. To our knowledge, this is the first time that a domain-based grouping data set is suggested and provided.

Host Similarity : Identifying Malicious Sources

The same malicious operator will tend to use hosts with the same services on them, since they are used for the same need. Having a similarity function among hosts in the Internet, it will enable us to deduce on one host using similar hosts. Assume that we see two hosts that use the same exact version of MySQL, PHP and many other products. It is unlikely to be coincidence and if there is threat intelligence on malicious activity of one host, then it is likely the other host belongs to the same operator and is used for malicious activity too. Like in the malware analysis, we can use malware in order to label related hosts. If a malware communicates with two malicious domains, the malware are likely to belong to the same operator and so are the domains. Given the services profile of the hosts, we can learn the similarity function and use it to identify new domains for which we didn't see malware communication. Attributing different domains to the same operator was done before (Starov et al. 2018) but as far as we know we are the first to present host similarity problem based on agnostic label, and provide a data set for it.

Labeling Hosts Via Operator Domain Pivoting The labeling method for hosts is based on the malware domain labeling. For each domain d , one can resolve the ip address on which it is being hosted (e.g., via nslookup). We will mark the resolving function as $resolve(ip)$. Shodan provide scans of hosts in the Internet, giving the $signature(ip)$ of services available on the host. Hence, given a data set D of malware, and the functions $OperatorDomains(m)$, $resolve(ip)$, $signature(ip)$ function we label

$$\forall m_1, m_2 \in D, \forall d \in OperatorDomains(m_1) \cap OperatorDomains(m_2), \\ \forall ip_1, ip_m \in resolve(d) \\ (signature(ip_1), signature(ip_m), True).$$

The negative samples are due to a Cartesian product of signatures of ip address that don't obey the positive samples rule.

Lateral Movement : The Attacker Moves to its Target

An attacker usually starts an attack in a point in the network that is far from the final destination. Therefore, the attacker should learn the network and travel to the destination, a phase called lateral movement (Johnson and Hogan 2013). This path might be long and use edges that on their own might be reasonable but as a part of a path they might be very suspicious. The goal of the defender is to learn the access pattern graphs (which might differ by usage) and use it in order to assign probabilities to paths.

Stealth Port Scan : The Attacker Explores its Target

An attacker that has a hold on one host (computer) in a network is likely to want to gain a hold on other hosts too. Knowing which services are deployed on a host might help the attacker to use an exploit for one of them. Accessing common ports related to these services can help with mapping the available services. A port scan (Gadge and Patil 2008) is access to a sequence of ports in order to map them. This technique is problematic from the point of view of the attacker since a scan of many ports is very noisy and the attacker might be detected. More sophisticated attackers, called Advanced Persistent Threat (APT) (Virvilis and Gritzalis 2013), use stealth port scans. They select very few informative ports in order to profile the target host. A possible method to identify a stealth port based on the low probability of access to a given combination of ports. To our knowledge, this is the first time that the problem is defined, and a data set is provided for it.

Bind and Reverse Shell : The Attacker Gets a Hold on its Target

As part of a manually operated targeted attack process, the attacker may be advancing from host A to host B, ultimately gaining a remote interactive command shell on host B. Two very common approaches to achieve a remote shell are reverse shell and bind shell, and they can be used interchangeably, commonly depending on the attacked network structure and firewall configurations. Both methods require two consecutive connections between the source and the destination and are available in common penetration tools such as Metasploit (<https://www.metasploit.com/>). In the bind shell scenario, an initial connection from A to B is used to exploit the vulnerability on a specific port (compromised service) and a follow-up connection from A to B is used as an interactive shell on a different (and commonly unused by other services) port. In the reverse shell scenario, the follow-up connection is from B to A and not from A to B. To our knowledge, this is the first time that a labelled data set is provided for this problem.

Machine Learning Challenges in Cyber-Security

We detail the below machine learning challenges in cyber-security. These challenges were chosen since they are common in cyber-security problems and that coping with them is essential for solution. For example, imbalanced data sets appear whether one tries to do malware detection, domain reputation or identify network intrusion. If one fails to identify an imbalanced data set, one might end up with a model claiming: "Everything is benign". While the model will have astonishing accuracy, it will provide no value. Off course, the imbalanced challenge is not unique to the cyber security domain, yet it is an essential challenge in many cyber security problems.

Attacker-Defender Game

Cyber-security contexts may have different, purposeful adversaries with different aims, techniques of attack, and capabilities, the latter including the knowledge of their adversary (the defender). Advancement of one of the sides will not end the game but will lead to a new round with different settings. Citing Slick Willie Sutton: "I rob banks because that's where the money is"; We know that this game will be played for long time.

Game theory has been applied to cyber-security (Manshaei et al. 2013), (Roy et al. 2010). What makes the game more interesting is the asymmetries in it. The defense systems are usually deployed before the attack and the attacker can treat them as given. We can assume that the attacker knows which defense system they should cope with. The attacker might even have access to the defense system and know its logic.

There are also asymmetries in the knowledge. The attacker knows the goals of the attack, the timing and the attack steps. The defender is familiar with the protected assets and the attacked network. A typical example of the use of such asymmetry in the knowledge is the use of honeypots (Spitzner 2003). Honeypots are traps, hosts that no benign user should access. The attacker, not knowing that, might access the honeypot and found by the defender.

The defender can win not only by preventing an attack, but also by making it infeasible. Cyber-espionage is less attractive if the secrets will be found out years after they were used. Cyber-crime economics will make long attack not profitable in most cases.

The last interesting point is considering not a single attack as a game but the entire cyber-security arena as a game. A malware caught in one network might be uploaded to common threat intelligence repositories (e.g. <https://www.virustotal.com/>) and be identified in other locations. The defending side can proactively hunt resources and tools used by attackers (e.g., malicious domains, files signatures), making the compromising of future attacks easier. Coping with the

host similarity problem is an important step in this direction.

Lack of Labeled Samples and Certainty in Ground Truth

Most cyber-security tasks are essentially supervised learning tasks. Given an entity or activity, we should decide whether it is malicious or benign. Unfortunately, many times we lack the labels which are required for supervised learning.

Manual labeling is limited in scope and subjective. Therefore, most labels are coming from heuristics. Other than having errors, one might end up trying to model the heuristics he started with.

Using the community knowledge is a common approach. When the entity to classify is universal (appearing also outside the examined case, like a malware or a malicious domain but not network activity), one can use external black lists and white lists for labels. This method may introduce a domain adaption problem, since the lists are not coming from the same source as the entities to classify. Off course, such lists might be of low quality.

Consensus agreement, using the verdict of many vendors as the concept, is another common solution. Other than the potential intellectual property problems, there are algorithmic problems with this approach. The consensus doesn't necessarily agree with the ground truth. Important demonstration of this problem is with Advanced Persistent Threats (APT) (Virvilis and Gritzalis 2013). Attacks of APTs are usually sophisticated and targeted. An APT will tend to use its malware on a single target and therefore its malware won't have threat intelligence recognizing it and so the malware will evade detection. Using consensus agreement as the concept is like asking your model not to detect APTs.

Algorithmic solutions might involve the use of unsupervised learning (Hastie, Tibshirani, and Friedman 2009) methods and specifically anomaly detection (Chandola, Banerjee, and Kumar 2009). While anomaly detection is popular in many domains, it is usually not used in cyber-security due to inherent difficulties (Sommer and Paxson 2010), that we see as machine learning challenges. These solutions may be problematic since some domains, like networks, have a plethora of benign anomalies. A sophisticated attacker is likely to be aware of the defender and try to make sure that its actions do not look anomalous. In this scenario we might end up with plenty of anomalies, none of which is related to the attacker.

Some promising research directions are active learning (Settles 2010) and semi-supervised learning (Zhu 2006). Active learning can help in identifying the most informative entities to label, and help in building a small labeled data set. One should note that some of the labeling is subjective and in many cases it is not possible to label with high certainty. This leads us to learning

in the presence of labeling errors (Kearns and Li 1993), (Kearns 1998). Given a small labeled data set, one can apply semi supervised learning and by doing so be able to predict as in supervised learning.

Imbalanced Data Sets

We consider a data set to be imbalanced when the ratios between the majority set and the minority sets are large (Chawla and Japkowicz 2004). While that in the machine learning community a ratio of 1 to 10 is considered imbalanced, malicious training examples in cyber-security data sets may be extremely rare, and imbalance ratios of 1 to 10,000 are common.

When the ratio between the sets is large but there are enough samples in each set, we use the term relative imbalance. In the more severe case in which there are not enough samples in the minority set, this is absolute imbalance. This is a common situation in cyber-security.

The Tragedy of Metrics

As we said, in the typical cyber-security use case we have only a handful positive labeled samples and we usually don't know how many other samples in our data set are actually positive.

This leads to a tragedy that goes much further than algorithmic considerations. Since we don't know what the actual positives are, we usually cannot know the positive rates. This in turn means that we cannot estimate our recall. Note that the cost of a false positive is a waste of some hours, loss of confidence in the system and in the long term the loss of a customer by the vendor. On the other hand, the cost of false negative might be millions of times more expensive. The inability of estimating recall leads customers and vendors into improving the precision. Since it is easy to trade off recall for precision by requiring higher confidence, we might expose the protected assets to more risk by racing to reduce false alarms.

A common method to handle this scenario is to use a cost matrix and assign a higher cost to false negatives. This method is not directly applicable here since we are not aware of our false negatives.

Achieving high lift is usually passed unnoticed. If the positive ratio is 1 to 10,000 and our classifiers precision is 10% then our precision lift is 1,000. However, the user of the system observes a single success in 10 alerts, which is not observed as good performance.

The common assumption of Independent Identical Distribution doesn't hold in cyber-security. For example, malware coloring leads to many polymorphic variations of the same malware. These instances are very dependent, making the usual statistical guarantees misleading. Even without deliberate attempts the assumption might be violated. Hosts in the same network segment (e.g., DMZ) are related. Being able to identify hardware of the same vendor will probably won't generalize well to other vendors. There is a need for mathematical methods to cope with these scenarios.

Domain Adaptation

Domain adaptation is a scenario in which the test distribution on which a model is evaluated is different from the train distribution that was used to build the model (III 2009). The cyber world looks different in each of its areas. Threats, entities and networking behavior are different among parties acting in various scales, domains, cultures and geo-locations. A basic challenge therefore is adapting an effective defense from one domain to another.

Concept Drift

A security expert copes successfully with the challenges listed above and deploys high performing system to production environment. In a short time, the system suffers a severe degradation in performance. The model did not change. It is the world that changed. Cyber-security is a rapidly evolving field. Models and insights are likely to become obsolete quickly. Volumes change due to technological advancement, new protocols and domains appear and malicious activity also has trends and fashions. The change in the source of analyzed entity due to a change in time is called concept drift (Gu, Tan, and He). To avoid continually starting from scratch, ways to track concept drift and extreme changes in the domain need to be developed.

Data Sets

We think that the use of machine learning in cyber-security should change. We also think that the cyber community should help the machine learning community to become more involved in this field.

One of the key obstacles to investigating cyber-security problems is the lack of appropriate data sets. There are many important cyber-security data sets like Microsoft's malware data set (Ronen et al. 2018), Los Alamos's traffic data set (Turcotte, Kent, and Hash 2017) and EndGame's Ember malware properties data set (Anderson and Roth 2018). However, we feel that there are no suitable data sets that will enable academic researchers to cope with the problems and challenges we listed.

In general, companies do not tend to share their data, and cyber-security data is even more sensitive than usual. However, by using anonymization, a bit older data, and removing sensitive information, one can address most concerns. Once this is done, the data contributing company will be able to enjoy from the combined work of a world of experts working on cyber problems.

The following data sets were contributed by Palo Alto Networks and Shodan (for academic use only). We do hope that these data sets will boost the research of machine learning in cyber-security. We also hope that other organizations will follow and publish more data sets and the research will be further boosted.

Access to Data Sets

We would like to provide researchers access to the data sets for academic use. In order to gain access to the data sets please contact data-sets@paloaltonetworks.com with "Access to data request" in the title.

Malware Polymorphism

We relate a malware to the domains it contacted. The data set is made of a sample of malware identified by Palo Alto Networks in a given period. For each malware we provide identifiers and the domains accessed by the file. One can find malware variants by creating pairs of files communicating with the same domains. Examples of pairs that are not variants can be generated by matching files communicating with no common domain. Note that one can generate $O(n^2)$ negative pairs and it is up to the researcher to choose the threshold that fits best its needs.

The malware data set construction demonstrates away to cope with the lack of labeled samples. The labels are imbalanced. The variety of malware lead to domain adaptation. We would like to provide in the future data collected on later periods, and present concept drift.

Malware data set

- *SHA256* The signature of the file given the SHA-256 hash function.
- *md5* The signature of the file given the MD5 hash function.
- *ssdeep* The signature of the file given the SSDEEP fuzzy hash function
- *size* File size

Communication data set

- *SHA256* The signature of the file given the SHA-256 hash function.
- *domain* The name of the domain with which the malware communicated. A malware that communicated with few domains will have a record per domain.
- *IP* The IP address to which the domain was resolved.

Host Similarity

The host similarity data set is based on the malware polymorphism data set. We consider two hosts (represented as IP addresses) as similar if the same malware communicated with both.

The host similarity data set construction demonstrates a way to cope with the lack of labeled samples. The labels are imbalanced. The variety of host lead to domain adaptation. We would like to provide in the future data collected on later periods, and present concept drift.

Shodan (<https://www.shodan.io/>) contributed the host services signature: The list of services opened on the host and the profile of each service. For example, see <https://www.shodan.io/host/180.183.160.140>. Given the pairs of related and unrelated host, generated

by malware polymorphism, one can learn a similarity function in the services space. This similarity function enables identifying hosts similar to a host in question. Given that the similar hosts are malicious, it is likely that the considered host belongs to the same operator and is also malicious.

The communication data set from the malware polymorphism section is applicable here too.

For each IP address, we provide the the scan result in a JSON format. For details about the format, see Shodan <https://www.shodan.io/>.

Ngrams Data Set

The data set is a derived from a series of benign and malicious files, with identifying names (SHA256s) removed, gathered over a period of about a month in 2017. For each file, we capture a histogram of 4-grams of the byte code. For example, the string 01234 would produce the 4-grams 0123 and 1234. Note that you can derive approximate (exact except for the end of file) 1, 2, and 3 grams from this data. We also have labels and family tags. The labels are benign, malicious, questionable. The tags describe families of malware, not every sample has a family tag, some have more than one. The median number of 4grams per file is 105,888.

The variety of malware lead to domain adaptation. We would like to provide in the future data collected on later periods, and present concept drift.

There are two pieces of data:

1. A mapping of files indices to ngrams where each line is a tab separated list:
 - File Index
 - Ngram 1 : Ngram 1 Count
 - Ngram 2 : Ngram 2 Count
 - etc.
2. A mapping of files to labels where each line is a tab separated list:
 - File Index
 - Verdict (0=benign, 1=malware, 2=greyware)
 - Family Label 1
 - Family Label 2
 - etc.

Android Malware Family Data Set

The data set is a derived from a series of mixed Android APK files, with identifying names (SHA256s) removed, gathered over a period of about a month in 2017. For each file, we provide an XML report of metadata generated during the static and dynamic analysis (generated by *WildFireATM*). For example, the certificate that used to sign the Android APK file; the domains that being contacted after the APK file got installed; the API call sequences after installation. We also have labels and family tags. The labels are benign, malicious, greyware. The family tags describe family names of malware. One malware may have multiple family tags.

The labels in this data set are imbalanced. The variety of malware lead to domain adaptation. We would like to provide in the future data collected on later periods, and present concept drift.

There are two pieces of data:

1. A mapping of files to labels and tags:
 - File Index
 - static_report : a JSON object
 - dynamic_report : a JSON object
 - file_info : a JSON object
2. A mapping of files to labels where each line is a tab separated list:
 - File Index
 - Verdict (0=benign, 1=malware, 2=greyware)
 - Family Label 1
 - Family Label 2
 - etc.

Bind Shell Data Set

The bind shell data set was collected from some networks over a period of 4 months. It consists of a set of consecutive connection pairs, that were created by pairing connections from source to destination that follow the following rules:

The data set presents the challenge of lack of labels and is also highly imbalanced.

1. Both connections are observed within a specific short time frame.
2. The destination port of the first connection is different from the second connection.
3. The connection pairs make good candidates for actual bind shell attack, based on a noise filtering algorithm applied at collection time.

The aggregative features mentioned below that relate to counts of hosts and ports are calculated on the connection pairs from the entire network without noise filtering over a given time period.

Each connection pair has the following features:

- index , Index of the sample
- label , Is forward shell, 1 for true, 0 for false, -1 for missing
- source_host_id , Index of the source
- is_new , True if the source and destination hosts were not seen on the network for a period of time prior to the creation of the connection pair
- s_phase1_initiators_hosts , Amount of Destinations accessed from this source using the same phase_1 port
- s_phase2_initiators_hosts , Amount of destinations accessed from this source using the same phase_2 port
- s_phase1_initiators_ports , Amount of distinct phase_2 ports accessed from this source to all destinations using the same phase_1 ports

- s_phase2_initiators_ports , Amount of distinct phase_1 ports accessed from this source to all destinations using the same phase_2 port
- s_port_count , Popularity of this phase_1 and phase_2 ports couple across the network
- s_src_port_phase1 , Port used in the first session
- s_src_port_phase2 , Port used in the second session
- s_pair_phase1_cnt , Amount of distinct phase_1 ports between this source and destination
- s_pair_phase2_cnt , Amount of distinct phase_2 ports between this source and destination
- s_start_time_phase1 , Start time of the first session
- s_start_time_phase2 , Start time of the second session
- s_duration_phase1 , Duration of the first session
- s_duration_phase2 , Duration of the second session
- s_dst_port_phase1 , Destination port of the first session
- s_dst_port_phase2 , Destination port of the second session
- s_volume_phase1 , Volume (source to destination) of the first session
- s_volume_phase2 , Volume of the second session
- s_rvolume_phase1 , Returning volume (destination to source) of the first session
- s_rvolume_phase2 , Returning volume (destination to source) of the second session
- s_path_phase1 , Protocol used in the first session
- s_path_phase2 , Protocol used in the second session
- s_spfss_unique_srcs , Amount of distinct sources creating connections to the candidate's destination with the same phase_1 and phase_2 ports
- s_arb_host_count , Amount of sources that accessed this destination with the same phase_1 port and followed up with another connection (i.e. a phase_2 session was detected)
- s_arb_port_count , Amount of distinct phase_2 ports accessed after accessing this destination and phase_1 port

Network Traffic Data Set

The network traffic data set is a collection of network sessions. The sessions are aggregated in 10 minutes buckets. Hence, if A communicated with B twice in that bucket there will be a single traffic record with cnt=2.

The data set lacks labels and even the concepts of stealth port scan and lateral movement are a bit vague (e.g., How long should be a path of hosts in order to be considered as lateral movement). The data sets were collected from different sites, presenting domain adaptation challenge. Data set from different periods present concept drift.

The columns are the following:

- *min_start_time* Time of the session, data set beginning is set to Alan Turing's birthday, June 23th, 1912, and relative time is given.
- *src_index* Source IP representation. IP addresses are collected in this in the index in the data set (in order to keep privacy).
- *dst_index* Destination IP representation. IP addresses are collected in this in the index in the data set (in order to keep privacy).
- *src_port* Source port
- *dst_port* Destination port
- *tvolume* Volume of traffic from source to destination
- *rtvolume* Volume of traffic from destination to source
- *pkt* Number of packets from source to destination
- *rpkt* Number of packets destination to source
- *cnt* Number of sessions
- *failed_num* number of failed sessions
- *path* Protocol used for communication

This data set can be used for the cyber problems of stealth port scan and lateral movement.

References

- [Anderson and Roth 2018] Anderson, H. S., and Roth, P. 2018. EMBER: an open dataset for training static PE malware machine learning models. *CoRR* abs/1804.04637.
- [Chandola, Banerjee, and Kumar 2009] Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41(3):15:1–15:58.
- [Chawla and Japkowicz 2004] Chawla, N. V., and Japkowicz, N. 2004. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl* 1–6.
- [David and Netanyahu 2015] David, O. E., and Netanyahu, N. S. 2015. DeepSign: Deep learning for automatic malware signature generation and classification. *Proceedings of the International Joint Conference on Neural Networks* 2015-Septe.
- [Gadge and Patil 2008] Gadge, J., and Patil, A. A. 2008. Port scan detection. In *2008 16th IEEE International Conference on Networks*, 1–6.
- [Gu, Tan, and He] Gu, S.; Tan, Y.; and He, X. Concept drifting.
- [Hastie, Tibshirani, and Friedman 2009] Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *Unsupervised Learning*. New York, NY: Springer New York. 485–585.
- [Hu and Tan 2017a] Hu, W., and Tan, Y. 2017a. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN.
- [Hu and Tan 2017b] Hu, W., and Tan, Y. 2017b. On the Robustness of Machine Learning Based Malware Detection Algorithms. 1435–1441.
- [III 2009] III, H. D. 2009. Frustratingly easy domain adaptation. *CoRR* abs/0907.1815.
- [Johnson and Hogan 2013] Johnson, J. R., and Hogan, E. A. 2013. A graph analytic metric for mitigating advanced persistent threat. In *2013 IEEE International Conference on Intelligence and Security Informatics*, 129–133.
- [Kearns and Li 1993] Kearns, M., and Li, M. 1993. Learning in the presence of malicious errors. *SIAM Journal on Computing* 22:807–837.
- [Kearns 1998] Kearns, M. 1998. Efficient noise-tolerant learning from statistical queries. *J. ACM* 45(6):983–1006.
- [Khorshidpour, Hashemi, and Hamzeh 2017] Khorshidpour, Z.; Hashemi, S.; and Hamzeh, A. 2017. Learning a Secure Classifier against Evasion Attack. *IEEE International Conference on Data Mining Workshops, ICDMW* 295–302.
- [Kumar, Wicker, and Swann 2017] Kumar, R. S. S.; Wicker, A.; and Swann, M. 2017. Practical machine learning for cloud intrusion detection: Challenges and the way forward. *CoRR* abs/1709.07095.
- [Kuriakose and Vinod 2015] Kuriakose, J., and Vinod, P. 2015. Unknown Metamorphic Malware Detection : Modelling with Fewer Relevant Features and Robust Feature Selection Techniques Unknown Metamorphic Malware Detection : Modelling with Fewer Relevant Features and Robust Feature Selection Techniques. (APRIL).
- [Luo 2016] Luo, G. 2016. PrediT ML : a tool for automating machine learning model building with big clinical data. *Health Information Science and Systems* 1–16.
- [Manshaei et al. 2013] Manshaei, M. H.; Zhu, Q.; Alpcan, T.; Bacşar, T.; and Hubaux, J.-P. 2013. Game theory meets network security and privacy. *ACM Comput. Surv.* 45(3):25:1–25:39.
- [McLaughlin et al. 2017] McLaughlin, N.; del Rincón, J. M.; Kang, B.; Yerima, S. Y.; Miller, P. C.; Sezer, S.; Safaei, Y.; Trickel, E.; Zhao, Z.; Doupé, A.; and Ahn, G.-J. 2017. Deep Android Malware Detection. *Codaspy* 301–308.
- [Patri, Wojnowicz, and Wolff 2017] Patri, O. P.; Wojnowicz, M. T.; and Wolff, M. 2017. Discovering Malware with Time Series Shapelets. 6079–6088.
- [Ronen et al. 2018] Ronen, R.; Radu, M.; Feuerstein, C.; Yom-Tov, E.; and Ahmadi, M. 2018. Microsoft malware classification challenge. *CoRR* abs/1802.10135.
- [Roy et al. 2010] Roy, S.; Ellis, C.; Shiva, S. G.; Dasgupta, D.; Shandilya, V.; and Wu, C. Q. 2010. A survey of game theory as applied to network security. *2010 43rd Hawaii International Conference on System Sciences* 1–10.
- [Saxe and Berlin 2017] Saxe, J., and Berlin, K. 2017. eXpose: A Character-Level Convolutional Neural Net-

- work with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. *arXiv*.
- [Settles 2010] Settles, B. 2010. Active learning literature survey. Technical report.
- [Sommer and Paxson 2010] Sommer, R., and Paxson, V. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, 305–316.
- [Spitzner 2003] Spitzner, L. 2003. Honeypots: catching the insider threat. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, 170–179.
- [Starov et al. 2018] Starov, O.; Zhou, Y.; Zhang, X.; Miramirkhani, N.; and Nikiforakis, N. 2018. Betrayed by your dashboard: Discovering malicious campaigns via web analytics. In *Proceedings of the 2018 World Wide Web Conference, WWW '18*, 227–236. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee.
- [Stoica et al. 2017] Stoica, I.; Song, D.; Popa, R. A.; Patterson, D. A.; Mahoney, M. W.; Katz, R. H.; Joseph, A. D.; Jordan, M. I.; Hellerstein, J. M.; Gonzalez, J. E.; Goldberg, K.; Ghodsi, A.; Culler, D.; and Abbeel, P. 2017. A berkeley view of systems challenges for AI. *CoRR* abs/1712.05855.
- [Turcotte, Kent, and Hash 2017] Turcotte, M.; Kent, A.; and Hash, C. 2017. Unified host and network data set. *ArXiv e-prints*.
- [Villegas 2017] Villegas, A. M. 2017. Function identification and recovery signature tool. *2016 11th International Conference on Malicious and Unwanted Software, MALWARE 2016* 157–164.
- [Virvilis and Gritzalis 2013] Virvilis, N., and Gritzalis, D. 2013. The big four - what we did wrong in advanced persistent threat detection? In *2013 International Conference on Availability, Reliability and Security*, 248–254.
- [Woodbridge et al. 2016] Woodbridge, J.; Anderson, H. S.; Ahuja, A.; and Grant, D. 2016. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. *Arxiv*.
- [Xie, Girshick, and Farhadi 2015] Xie, J.; Girshick, R.; and Farhadi, A. 2015. Unsupervised Deep Embedding for Clustering Analysis. 48.
- [You and Yim 2010] You, I., and Yim, K. 2010. Malware obfuscation techniques: A brief survey. In *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, 297–300.
- [Zhu 2006] Zhu, X. 2006. Semi-supervised learning literature survey.