future internet

MDPI

Article

Windows Based Data Sets for Evaluation of Robustness of Host Based Intrusion Detection Systems (IDS) to Zero-Day and Stealth Attacks

Waqas Haider [1,*], Gideon Creech [1], Yi Xie [2] and Jiankun Hu [1]

[1] School of Engineering and Information Technology, Australian Defence Force Academy, University of New South Wales, Canberra 2052, Australia; g.creech@adfa.edu.au (G.C.); J.Hu@adfa.edu.au (J.H.)

[2] School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, China; xieyi5@mail.sysu.edu.cn

* Correspondence: waqas.haider@student.adfa.edu.au; Tel.: +61-452-026-599

Academic Editor: Sherali Zeadally
Received: 13 April 2016; Accepted: 24 June 2016; Published: 5 July 2016

Abstract: The Windows Operating System (OS) is the most popular desktop OS in the world, as it has the majority market share of both servers and personal computing necessities. However, as its default signature-based security measures are ineffectual for detecting zero-day and stealth attacks, it needs an intelligent Host-based Intrusion Detection System (HIDS). Unfortunately, a comprehensive data set that reflects the modern Windows OS's normal and attack surfaces is not publicly available. To fill this gap, in this paper two open data sets generated by the cyber security department of the Australian Defence Force Academy (ADFA) are introduced, namely: Australian Defence Force Academy Windows Data Set (ADFA-WD); and Australian Defence Force Academy Windows Data Set with a Stealth Attacks Addendum (ADFA-WD: SAA). Statistical analysis results based on these data sets show that, due to the low foot prints of modern attacks and high similarity of normal and attacked data, both these data sets are complex, and highly intelligent Host based Anomaly Detection Systems (HADS) design will be required.

Keywords: operating system; kernel; auditing; anomaly; low foot print attacks

1. Introduction

Intrusion Detection Systems (IDS), specifically Host based Anomaly Detection Systems (HADS), have received a great deal of attention over the past decade due to their capability to detect zero-day and stealth attacks at the OS level [1,2]. There are three main components of a HADS: a data source (DS); feature selection and construction (FSC); and a decision Engine (DE) [3]. For an IDS design and evaluation, it is critical to have a comprehensive and public data set (e.g., one that can be related to a DS) commensurate with the targeted OS. In the IDS research community, for nearly two decades, relevant IDS algorithms have been generally tested using the publicly available benchmark data set KDD 98. Unfortunately, the KDD data set was compiled in 1998 which has lost the comprehensiveness and qualitative factors [4,5]. To tackle this, an ADFA-Linux Data Set (ADFA-LD) has recently been created [4]. However, both these data sets (e.g., KDD 98 and ADFA-LD) are LINUX oriented; and not applicable for a Windows based IDS.

In order to protect the largest OS market share (i.e., Windows OS) from the highly sophisticated threats such as zero-day attacks, stealth attacks, data exfiltration and Distributed denial of service (DDoS) attacks [6], following the *"defence in depth philosophy"*, the first step is to develop a comprehensive windows based IDS data set [7]. Although Windows OS has protection methods such as signature-based anti-viruses (AV), Address Space Layout Randomization (ASLR), and Data

Execution Prevention (DEP) [3], AVs are logically unable to detect zero-day attacks and stealth attacks can bypass these protection mechanisms [8,9]. Moreover, data exfiltration and DDoS attacks are acknowledged that the recent wave of cyber threat which can be encounter through anomaly detection mechanism [10,11].

In this paper two Windows OS-based IDS data sets targeting zero-day and stealth attacks are provided, which are being released publicly for the IDS research community. These data sets are available at the link [12]. In the following sections, firstly, we provide descriptions of these data sets, including their design purposes, rationales for their generation environments, their logic for audit data selection, and reasons for the selection and construction of attacks. Secondly, their structures and formats, and a way of using them to design HIDS for windows OS, are discussed. Finally, we provide a preliminary analysis of both data sets by the use of following two methodologies:

(a) The evaluation of the complexity of the data sets through frequency distribution method (see Section 5.1), which identify the frequency distributions of audit data (i.e., DLL calls) among normal and attacked data. Its purpose is to identify the natural similarity between attacked and normal data that is likely to assist in the design of an HIDS decision engine.
(b) Evaluating the complexity of the data sets through anomaly detection frame work by the use of several machine learning algorithms integrated with the novel feature selection scheme (see Section 5.2).

2. ADFA-WD

The main intention behind the generation of this data set was to fill the gap created by the unavailability of IDS data sets for Windows OS. It generally contains identified Windows-based vulnerability-oriented zero-day attacks. This section provides details of the ADFA-WD.

Purposes of Design: (i) to establish common evaluation standards for the Windows IDS research community; (ii) to incorporate a reflection of the modern sophisticated threads associated to Windows OS; (iii) to express the generic Windows OS endpoint; and (iv) to ensure the manageability, scalability and generality of the audit data set.

Rational for Generation Environment: the OS selected as the targeted host was Windows XP Service Pack 2 which does not have ASLR and DEP protection. The major reason for adopting this OS was to incorporate a defence-in-depth consideration which requires overlapping security controls to be optimized in isolation [13]. The targeted host also includes an FTP server, web server and management tool, a streaming audio digital radio package, wireless and Ethernet networking connectivity and a fake wireless access point. The purpose of setting up a variety of connectivities was to provide a representative network-based attack surface.

Logic of Audit Data Selection: nine core DLL calls (ntdll.dll, kernel32.dll, user32.dll, comctl32.dll, ws2 32.dll, mswsock.dll, msvcrt.dll, msvcpp.dll, and ntoskrnl.dll) were selected as audit data for both data sets based on three major considerations:- firstly, the audit data should be capable of representing system behaviour; secondly, it should have the capability to reflect modern threat vectors; and thirdly, we needed to consider its effectiveness in achieving efficient training and testing at the decision engine of HIDS.

Reasons for Selection and Construction of Attacks: in ADFA-WD, for attack construction, the relationships between vulnerabilities identified by CVE [14] and current attack vectors are considered. There are 12 known vulnerabilities exploited using automated hacking tools such as Metasploit [3] and the attack vectors includes TCP ports, browser attacks, Web based vectors and malware attachments. Basically, the major reason for selecting these attacks and crafting them was to realize the modern general threads which have been reported in the literature for the Windows OS.

3. ADFA-WD: SAA

The stealth attack-oriented Windows OS-based data set is actually an extension of ADFA-WD (Table 1). Its main goal was to establish a common standard evaluation data set for validating the resistivity of future HIDS against Windows based stealth attacks.

Table 1. Compositions of Data Sets.

Data Sets	Normal Training	Normal Validation	Attack
ADFA-WD	356 traces	1828 traces	5773 traces
ADFA-WD:SAA	same as above		863 traces

Purposes of Design: (i) to examine the Windows OS characteristics in order to provide a much more permissive environment than that of the LINUX OS for stealth attacks; and (ii) to realize the bypassing of stealth attacks from signature based protection such as AVs.

Rational for Generation Environment: as it is acknowledged, stealth attacks are often specific to a program. The remainder of the generation environment was the same as ADFA-WD apart from two additional specific targeted programs, Icecast V2.O and CesarFTP V0.99g, which could provide attack vectors for conducting stealth attacks.

Selection and Construction of Attacks: ADFA-WD: SAA was compiled by crafting three stealth attacks, namely; Doppelganger, Chimera, and Chameleon (note: Chameleon was conducted as both network and malware) considering the two targeted programs discussed above. The general process for conducting these stealth attacks comprises three hacking steps: firstly, an available specific targeted program resource is utilized: secondly, the targeted program's resource utilization sequence during normal operation is identified; and, finally, the resource utilization sequence is shuffled according to the particular malicious activity [3].

4. Structures, Formats and Utilization of Data Sets

The audit data (i.e., nine core DLL calls) was collected by running the Procmon [3] program to capture all DLL while various normal and attack activities were conducted. Each captured process trace contained DLL calls, with all traces provided by the extension .GHC. In Table 1, the structures of both data sets are described.

In order to use both data sets for the HADS design, normal training data traces could be used for training, normal validation data traces are used to measure the False Positive Rate (FPR) and attack data traces are used to measure the Detection Rate (DR) and False Negative Rate (FNR). On the other hand, while designing the signature-based Host IDS, a mixture of normal and attack data traces could be utilized for training, normal validation data for measuring the FPR, and attack data traces that are not used in training can be used for estimating the DR and FNR.

5. Data Analysis

In this section, analysis of both data sets (ADFA-WD and ADFA-WD: SAA) are provided. They were designed to answer the following two questions regarding both data sets: (i) how could the similarity or discrimination between their attacked and normal data be measured; (ii) what were their levels of complexity, i.e., did they represent sophisticated attacks? To answer the first question, a frequency distribution method was adopted and, for the second, a complete anomaly detection frame work is implemented using machine-learning algorithms and feature construction method, to measure the complexity in terms of accuracy and error.

5.1. Complexity Analysis through Frequency Distribution Method

We observed that in both data sets, each process consisted of nine selected DLL calls, of which seven different ones constituted a trace. This phenomenon depends on process execution, resource utilization and audit data selection. We further calculated the frequency percentage of the each distinct DLL for both data sets by the use of adding the DLLs of particular DLL and then divided by total number of DLLs. It is noticed that the frequencies of DLLs in attacked and normal data are very close to each other. For instance, the frequency percentages of kernel32.dll, in the total normal data of both data sets is 70.12, abnormal data of ADFA-WD has 74 and abnormal data of ADFA-WD:SAA has 67.69. Moreover, the complete percentages of audit data frequency distributions among normal and attacked data is provided for both data sets in Figure 1.

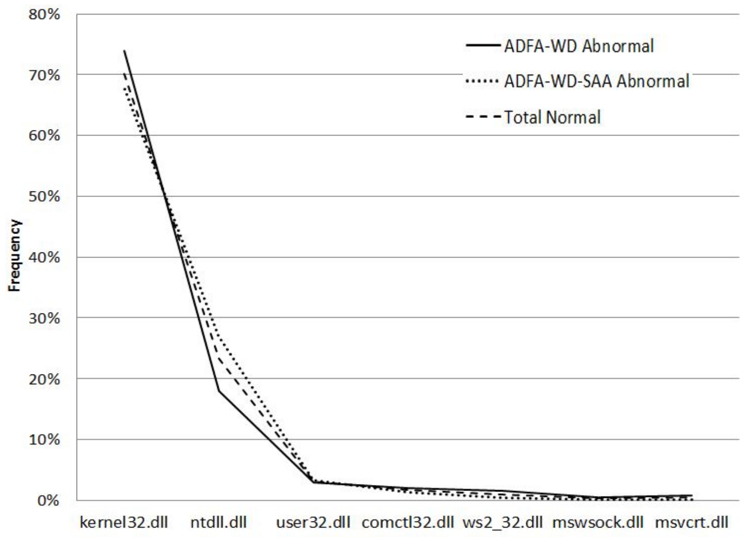


Figure 1. Audit data frequency distribution map.

It can be observed from Figure 1, that ADFA-WD had less similarity between attacked and normal data, which reflects the existence of a mid-level hacking, but in ADFA-WD: SAA a high similarity is observed which shows a highly skilled hacking presence. Further, it can be observed that the attacked and normal data in both data sets, in terms of frequency distribution map, are very close to each other and indicates the binary classification complexity.

5.2. Complexity Analysis through Anomaly Detection Frame Work

In the IDS domain it is observed that, one of the way to evaluate the complexity of the IDS data set is the utilization of anomaly detection or signature detection framework, where most of the frameworks are equipped with the machine learning algorithms. For instance, in [15–17] the authors utilized several machine learning algorithms as the part of anomaly detection framework, while the evaluation of ADFA-LD IDS data set [4]. Therefore, we implemented an anomaly detection frame-work for conducting a complexity analysis, by the use of machine-learning tool box in Matlab R2014a, Intel corei-7 @ 3.40 GHz processor with a 64-bit operating system and 16 GB of RAM. The framework is shown in the Figure 2 where it actually reflects the typical anomaly detection process or binary classification method assisted with machine -learning algorithms. Both data sets can be utilized at the data set entity point.

After the acquisition of a process trace, a novel Distinct Dynamic Link Library Count (DDLLC) was introduced and applied as a scheme for constructing a Feature Vector (FV) for the trace. It is elaborated in Figure 3, in which each trace is transformed to the feature vector consisting of the count of distinct DLL calls.

In order to generate a profile of the behaviors of traces and later classify test traces, we adopted the Support Vector Machine (SVM) with linear and RBF kernels [18], k-Nearest Neighbour (KNN) [19], Extreme Learning Machine (ELM) [20], Artificial Neural Network (ANN) [21] and an improved version of the Naive Bayes (NB) classifier [22] from Microsoft research. Algorithm 1 shows the binary classification task for N traces, where N represents the total number of training and test traces.

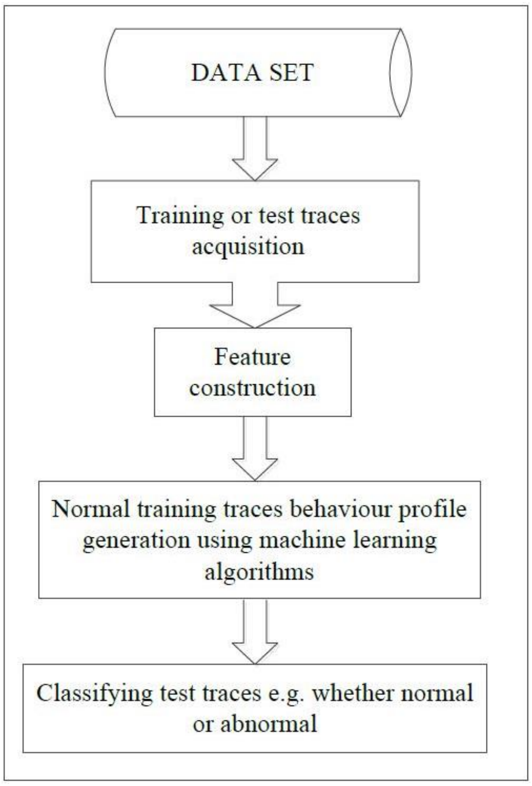
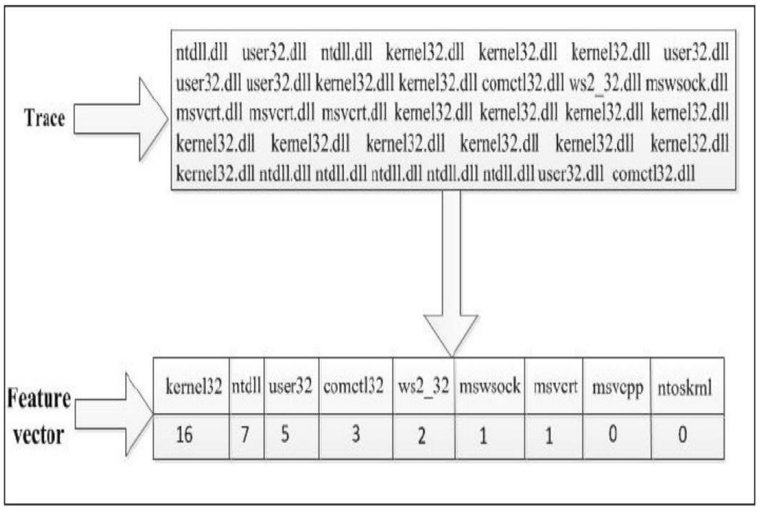

DATA SET
Training or test traces acquisition
Feature construction
Normal training traces behaviour profile generation using machine learning algorithms
Classifying test traces e.g. whether normal or abnormal

Figure 2. Anomaly detection frame work for the data sets complexity analysis.


Trace
ntdll.dll user32.dll ntdll.dll kernel32.dll kernel32.dll kernel32.dll user32.dll
user32.dll user32.dll kernel32.dll kernel32.dll comctl32.dll ws2_32.dll mswsock.dll
msvcrt.dll msvcrt.dll msvcrt.dll kernel32.dll kernel32.dll kernel32.dll kernel32.dll
kernel32.dll kernel32.dll kernel32.dll kernel32.dll kernel32.dll kernel32.dll
kernel32.dll ntdll.dll ntdll.dll ntdll.dll ntdll.dll ntdll.dll user32.dll comctl32.dll
Feature vector

kernel32	ntdll	user32	comctl32	ws2_32	mswsock	msvcrt	msvcpp	ntoskrnl
16	7	5	3	2	1	1	0	0

Figure 3. Distinct Dynamic Link Library Count (DDLLC) feature construction scheme.

In Algorithm 1 FM represents the feature matrix of size N rows and 9 columns, that is, 356 rows and 9 columns because the data set ADFA-WD has 356 normal training traces and each trace will have 9 features or attributes through DDLLC accordingly. The feature matrix for N traces can be formulated as:

$$FM^{i,j}(traces) = \Sigma_{i=1}^{N}\Sigma_{j=1}^{9}(FV^{i,j}) \quad (1)$$

where *j* is the counter to the dimensions of *FM* and for implementing the scheme of DDLLC. In line 1, Algorithm 1 inputs feature matrices of the training and test traces, with lines 2 to 5 designed to construct a profile of normal traces behaviors using any machine learning algorithm respectively. Note that each algorithm has its own key properties while learning feature matrices and classifying test traces. SVM classifies data points of feature matrix into two classes through maximizing the margin of the hyper plane by first transforming the data points in a linear or circular way. In KNN the distance between points in the *FV* is measured and, based on close distance, traces are placed into either one of two classes (e.g., 0 or 1, normal or abnormal). In ANN, each data point in an *FV* is considered an input and then the initial weight *w* of each *FV*, i.e., *w.FV*, is calculated. The activation function (i.e., the sigmoid function) is applied on *w.FV* to fit the ANN output with the actual value (i.e., 0 or 1 about traces). In ELM, a single hidden layer feedforward neural network selects randomly hidden layers and determines the output weight *w.FV* required to fit the target output relevant to a trace in *FM*. In contrast to the traditional ANN, the hidden layers do not need to be tuned iteratively and the activation functions are adoptable. Moreover, in NB, data points in the *FV* and class labels are processed according to their conditional probability to estimate the maximum likelihood of each trace in the *FM*.

Algorithm 1 Building Profile and Classification of traces

```
Require: $FM^{train}$ && $FM^{test}$
1: $I \leftarrow [(FM^{train}$, label]
2: $Build\_Profile^{train} \leftarrow \{I\}\&\&\{SVM \mid\mid KNN \mid\mid ANN \mid\mid ELM \mid\mid NB\}$
3: for i = 1 to N do
4: Learn $[Build\_Profile^{test}] \leftarrow Trace_i \epsilon (FM^{test})$ is -normal- or- abnormal
5: end for
```

Each adopted ML algorithm is tested with different possible parameter values and the results with optimum ones are displayed in the Table 2. The libSVM with the parameters $n = 5$ (cross validation value), $s = 0$ (default type of SVM), $t = 2$ (radial basis kernel function), $d = 5$ (degree in kernel function), $g = 0.14$ (as we have 7 features of each trace so 1 divided by the 7) and rest all on default values. Similarly, KNN with $k = 5$ (e.g., *k*-fold cross validation), ELM with number of hidden neurons = 50 , activation function = radbas, value to normalize DLL count = max DLL count plus 5 (e.g., this value is used to divide each feature instance of *FM* in the ELM case to normalize the data points between 1 and -1), and ANN with $i = 21$ (*i* shows the number of neurons) and activation function = sigmoid. The DR, FAR and the processing time of each algorithm along with DDLLC feature construction scheme are calculated with these parameters.

Table 2. Complexity of data sets using DDLLC as feature construction.

Algorithms	ADFA-WD			ADFA-WD-SAA		
	DR%	FAR%	Processing Time	DR%	FAR%	Processing Time
SVM	64	13	55	61	18	52
KNN	59	16	12	53	23	12
ANN	48	10	32	42	12	34
ELM	69.3	15	72	65	21	72
NB	72	12	55	68	14	48

In Table 2 the result actually reflects the complexity of both data sets, mainly in terms of accuracy. The DR was obtained by dividing the number of traces detected as attack by the total number of test attack traces. The FAR is a joint average error related to an FNR and FPR [3]. An FNR was obtained by dividing the number of incorrectly detected traces as attacks by the total number of test attack traces, and FPR by dividing the number of incorrectly detected traces as attacks by the total number of normal validation traces. Therefore, FAR was given by:

$$FAR = (FPR + FNR)/2 \times 100$$

It can be observed that both the linear and non-linear classifiers were unable to achieve the optimum performance for either data set for three main possible reasons: firstly, both data sets included sophisticated attacks, which resulted in a high similarity between the data points of normal and attacked traces; secondly, the feature construction scheme DDLLC was unable to assist classifiers in ascertaining natural differences between attacked and normal traces; and thirdly, an un-normalized distance was observed in the data points of the *FV* for all traces, which might have affected the classification capability of a particular classifier. Moreover, it can be observed in Table 2 that all the classifiers performed better for ADFA-WD than ADFA-WD-SAA which demonstrates the sophistication of stealth attacks.

6. Conclusion

In order to tackle the challenge of zero-day and stealth attacks on Windows OS, in this paper two major contributions in the domain of HADS were discussed. Firstly, two comprehensive Windows OS- based data sets (ADFA-WD and ADFA-WD: SAA) with informative description were compiled and released for the IDS research community. Secondly, a preliminary analysis using the frequency distribution method and machine learning algorithms integrated with a novel DDLLC based feature construction methodology was performed. The results demonstrated that, in order to design an effective HADS for Windows OS, attention must be paid to the selection/construction of features and adaption of the decision engine due to the observed complexity of the data sets. The work in this paper was supported by the Australian Research Council (ARC) linkage projects LP100200538 and LP110100602.

Author Contributions: Study design, survey design, data analysis and interpretation and manuscript writing: Haider and Hu. Data collection: Creech. Concept evaluation: Xie.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hu, J. Host-based anomaly intrusion detection. In *Handbook of Information and Communication Security*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 235–255.
2. Paul, B. Application firewalls in a defence-in-depth design. *Netw. Secur.* **1983**, *9*, 9–11.
3. Creech, G. Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks. Ph.D. Thesis, University of New South Wales, Sydney, Australia, 2014.
4. Creech, G.; Hu, J. Generation of a new ids test dataset: Time to retire the kdd collection. In Proceedings of the 2013 IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, China, 7–10 April 2013; pp. 4487–4492.
5. Creech, G.; Hu, J. A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns. *IEEE Trans. Comput.* **2014**, *4*, 807–819.
6. Information Systems Security, Assurance, and Privacy. Available online: http://aisel.aisnet.org/amcis2014/ISSecurity/GeneralPresentations/12 (accessed on 5 March 2015).
7. Osanaiye, O.; Cai, H.; Choo, K.K.R.; Dehghantanha, A.; Xu, Z.; Dlodlo, M. Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP J. Wirel. Commun. Netw.* **2016**, *1*, doi:10.1186/s13638-016-0623-3.
8. Osanaiye, O.; Choo, K.K.R.; Dlodlo, M. Distributed Denial of Service (DDoS) Resilience in Cloud: Review and Conceptual Cloud DDoS Mitigation Framework. *J. Netw. Comput. Appl.* **2016**, *67*, 147–165.
9. Walls, J.; Choo, K.K.R. A Review of Free Cloud-Based Anti-Malware Apps for Android. In Proceedings of the IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 20–22 August 2015; pp. 1053–1058.
10. Do, Q.; Martini, B.; Choo, K.K.R. Exfiltrating data from Android devices. *Comput. Secur.* **2016**, *48*, 74–91.

11. Dorazio, C.J.; Choo, K.K.R.; Yang, L.T. Data Exfiltration from Internet of Things Devices: iOS Devices as Case Studies. *IEEE J. Internet Things* **2016**, doi:10.1109/jiot.2016.2569094.
12. Windows based IDS data sets. Available online: https://www.unsw.adfa.edu.au/school-of-engineering-and-information-technology /professor-jiankun-hu (accessed on 21 March 2015).
13. Smith, C.L. Understanding concepts in the defence in depth strategy. In Proceedings of the IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, Taipei, Taiwan, 14–16 October 2003; pp. 8–16.
14. Common Vulnerabilities and Exposures. Available online: http://cve.mitre.org/data/refs/refmap/source-ISS.html (accessed on 12 January 2015).
15. Xie, M.; Hu, J.; Yu, X.; Chang, E. Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to ADFA-LD. In *Network and System Security*; Springer: Xi'an, China, 2014; pp. 542–549.
16. Haider, W.; Hu, J.; Xie, M. Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems. In Proceedings of the 2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), Auckland, New Zealand, 15–17 June 2015; pp. 513–517.
17. Haider, W.; Hu, J.; Yu, X.; Xie, Y. Integer Data Zero-Watermark Assisted System Calls Abstraction and Normalization for Host Based Anomaly Detection Systems. In Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 3–5 November 2015; pp. 349–355.
18. Amari, S.I.; Wu, S. Improving support vector machine classifiers by modifying kernel functions. *Neural Netw.* **1999**, *12*, 783–789.
19. Quinlan, J.R. *C4. 5: Programs for Machine Learning*; Elsevier: San Diego, CA, USA, 2014.
20. Huang, G.B.; Zhou, H.; Ding, X.; Zhang, R. Extreme learning machine for regression and multiclass classification. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2012**, *42*, 513–529.
21. Chen, W.-H.; Hsu, S.-H.; Shen, H.-P. Application of svm and ann for intrusion detection. *Comput. Oper. Res.* **2005**, *32*, 2617–2634.
22. Heckerman, D.; Geiger, D.; Chickering, D.M. Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.* **1995**, *20*, 197–243.