CCS: Collaborative Malware Clustering and Signature Generation using Malware Behavioral Analysis

Huabiao Lu, Xiaofeng Wang and Jinshu Su

School of Computer, National University of Defense Technology, Changsha, China
ccmaxluna@gmail.com, {xf_wang, sjs}@nudt.edu.cn

Abstract

The sheer volume of new malware found each day is growing at an exponential pace. Centralized systems that collect all malware samples to central severs can cause problems of single point of failure as well as processing bottlenecks. Previous works on distributed and scalable malware analysis are mainly applied for specific or simple malware. This paper presents CCS, a collaborative online malware analysis system which is applied for various malware and well scalable. Each sensors in CCS analysis their own malware samples accurately in-situ and then CCS aggregates those analyses among sensors in a load-balance way. We implemented a proof-of-concept version of CCS and performed experiments with 917 real-world malware samples; preliminary results from our evaluation confirm that CCS has comparable performance with centralized system, but much better scalability, and is approximately consistent with the result of AV scanners.

Keywords: *collaborative malware analysis; malware behaviors; local analysis; global aggregation, signature generation.*

1. Introduction

The battle against malicious software (a.k.a. malware) is becoming more difficult. The volume of new malware, fueled by easy-to-use malware morphing engines, is growing at an exponential pace [1]. In 2010, Symantec detected more than 286 mil-lion unique by hash malware samples, average 783,562 unique malware samples per day [1]. At the same time, security organizations usually deploy numerous wide-spread sensors to monitor the comprehensive malware samples. How to organize those large number of sensors and analysis such many malware samples is a challenge.

Centralized systems that collect all malware samples to central severs [2, 4, 6] can cause problems of single point of failure as well as processing bottlenecks. Previous works [18] on distributed and scalable malware analysis are initial works aiming at specific or simple malware. This paper presents CCS, a collaborative malware clustering and signature generation framework which is applied for various malware and well scalable. Sensors in CCS share their local view of malware behaviors through an information-driven accumulation structure (INFOACC) to gain global view. Based on the global view, each sensor analyzes its own malware samples locally and accurately. Finally, CCS aggregates those analyses in a load-balance way. Each sensor in CCS undertakes only a small part of computation and communication, therefore, CCS has well scalability.

The remainder of this paper is organized as follows. In Section 2, we firstly describe the system overview of CCS, and then present its main components. In Section 3, we validate our method through experiment on real malware samples and conclude the paper in Section 4.

2. Overview of CCS Malware Analysis Framework

2.1 Architecture of CCS

Figure 1 shows the architecture of our CCS, which consists of sensors and an infor-mation sharing structure INFOACC (as Figure 1(a) describes). Each sensor contains two phases: local analysis stage and global aggregation stage (as Figure 1(b) describes).

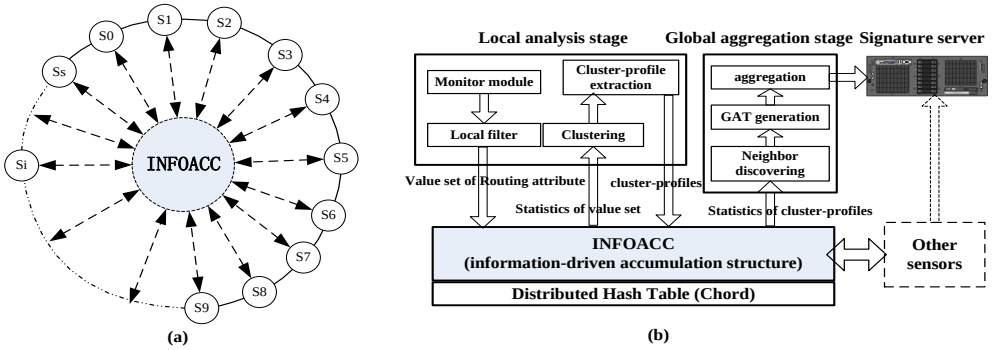

Figure 1. Architecture of our CCS

The INFOACC receives malware behaviors from all sensors and returns global statistics of those behaviors to source sensors. INFOACC is a concept component, and each sensor in CCS undertakes part of its work. In local analysis stage, the monitor module is responsible for logging suspicious malware behaviors; the local filter module selects representative behavior set among those logged behavoirs; then it sends the set to INFOACC, and in return gains global statistics; the clustering module clusters local malware samples based on those global statistics; at last, each sensor extracts profiles for each cluster, and sends the profiles (cluster-profiles) to INFOACC. After receiving global statistics of cluster-profiles from INFOACC, sensor enters into the global aggregation stage, each sensor computes neighbor sensors for each of its cluster-profiles; then sends the cluster-profile to its neighbor sensors. The neighbor finds the most similar cluster-profile as the cluster-profile's neighbor-cluster and returns their similarity weight; CCS builds HDHTs (Heuristic Distributed Hierarchical aggregation Trees) for each kind of malware; finally, sensors aggregate their local cluster-profiles along those HDHTs; and global cluster-profiles will be generated at the root of each HDHTs, and the root sent global cluster-profiles to signature server. In ideal situation, sensors should be distributed on the Internet and monitor module runs in real-time while other components run periodically.

2.2 INFOACC (information-driven accumulation structure)

Definition 1: *routing attribute*. The attribute used by INFOACC mapping a malware behavior to a key to route the malware behavior to the sensor responsible for it is called *routing attribute*.

Definition 2: *accumulating attributes*. The attributes whose values are accumulated by INFOACC among the malware behaviors with the same value of routing attribute are called *accumulating attributes*, e.g. the number of occurrences, and the source sensor set.

Figure 2 describes the principle of INFOACC structure. Each sensor has representative values of a routing attribute extracting from malware behaviors, as Figure 2(a) describes. In Figure 2(a), there are 16 sensors: s_0, s_1, s_2, ….. , s_{15} ; each sensor $s_i (0 \le i \le 15)$ extracts $k_i (0 \le i \le 15)$ distinct representative values, e.g., sensor s_0 extracts:

$value_{s0\_1}, value_{s0\_2}, value_{s0\_3}, \ldots\ldots, value_{s0\_k0}$. Firstly each sensor sends the value set to INFOACC according to DHT. As DHT implied, the same values will be sent to the same sensor, as Figure 2(b) describes, $value_{s0\_2}$, $value_{s3\_1}$, $value_{s5\_3}$, $value_{s8\_1}$, $value_{s13\_3}$ are same values and converged to sensor s_6. Then, each sensor accumulates the accumulating attributes among its received values. Assume that the values set of routing attribute received by sensor j is presented as $Acc_j = \{value_{j\_0}, value_{j\_1}, ..., value_{j\_rj}\}$, $attr(value_{j\_i})(0 \le i \le rj)$ is the accumulating attribute value carried by $value_{j\_i}$, and $sensor(value_{j\_i})(0 \le i \le rj)$ is the source sensor ID of $value_{j\_i}$. Then each sensor uses algorithm 1 to compute global statistics and returns the statistics to the source sensors. As a result, all sensors know the statistics of its own values, as Figure 2(c) describes, $Stat(value_{sj\_h})$ presents the global statistics of $value_{sj\_h}$.

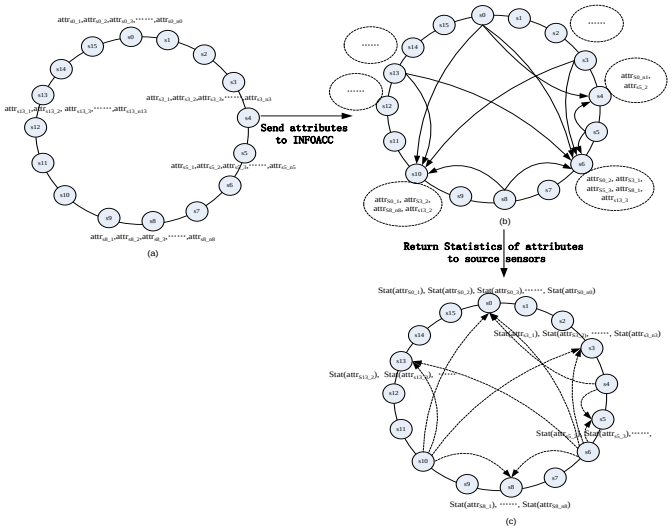

Figure 2. Principle of INFOACC Structure

2.3 Local Analysis Stage

In local analysis stage, the monitor module logs the behaviors generated by malware samples; local filter module selects representative value set from value set of routing attribute of logged behaviors, and then sends the set to INFOACC, carrying with number of samples that the value occurs. After receiving the global numbers from INFOACC, the clustering module calculates a weight for each value using the TF*IDF weight [10, 14], and transforms each sample as a weighted vector composed of weights of values the sample contains, then clusters malware samples utilizing the agglomerative hierarchical clustering algorithm [3] and Davies-Bouldin index [13]. Finally, cluster-profile extraction module generates cluster-profile for each cluster, and sends the routing attributes of cluster-profiles to INFOACC.The TF*IDF value increases proportionally to the number of samples the value occurs, but is offset by the frequency of the value in benign software. The weight of value i is defined as:

$$W_i = (1 + \log(cnt_i)) \log(\frac{1}{frequency_i}) \qquad (1)$$

Where cnt_i denotes the number of samples the value i occurs globally in CCS, and $frequency_i$ denotes the frequency of the value i occurs in benign software.

Each sensor in CCS is agnostic to the number of clusters of its own malware samples. We use agglomerative hierarchical clustering algorithm to build a hierarchy of clusters; then utilize Davies-Bouldin index [13] to find an appropriate location to cut the hierarchy.

Algorithm 1 INFOACC accumulating algorithm

```
1:  INPUT: value set Acc_j = {value_{j_0}, value_{j_1}, ....., value_{j_tj}}
2:  OUTPUT: messages sent to source sensor set
3:  for t = 0 -> value_Max do
4:      Stat[t] = 0;    SrcID[t]=NULL;
5:  end for
6:  for all value i in value set do
7:      Stat[i] += attr(i);    add sensor(i) to SrcID[i];
8:  end for
9:  for t= 0 -> value_Max do
10:     if(Stat[t] > threshold)do
11:         send t and Stat[t] to the sensors in SrcID[t]
12:     end if
13: end for
```

2.4 Global Aggregation Stage

Each sensor in INFOACC uses the similar edition of **algorithm 1** to accumulate global statistics (source sensor set). After receiving statistics from INFOACC, sensor enters into global aggregation stage. Neighbor discovering module computes the similarity between local cluster-profile and remote sensors; calculates a threshold using interval estimation; selects the sensors within the threshold as the neighbor sensor set of the cluster-profile.

Definition 3: *neighbor_cluster*. Given a cluster-profile c_s in sensor n_i, n_j is c_s's neighbor and c_t is the most similar cluster-profile in n_j with c_s, we call c_t is c_s's neighbor_cluster.

Definition 4: *cluster_couple*. Given cluster-profiles c_s and c_t, if c_t is c_s's neighbor_cluster and c_s is c_t's neighbor_cluster. We call c_s and c_t are cluster_couple.

Neighbor discovering module of sensor s_i sends cluster-profile to the profile's neighbor set. Then, each neighbor s_j selects the most similar one $P_{s_j\_c_b}(s_j \in neighborSet_{s_i\_c_a})$, and returns the ID and their similarity. The module of sensor s_i knows which remote cluster-profiles treat the cluster-profile as neighbor_cluster, and can deduce out which are its couple_clusters.

Assuming each cluster-profile as a vertex, and an edge connecting the cluster_couple, and then CCS turns into a graph. Each cluster-profile runs a revised distributed algorithm for spanning trees [5] daemon to generate a HDHT. As a result, each HDHT is composed of similar cluster-profiles. Ideally each malware correspond to a HDHT, and like a hierarchy, the most similar couples are at the bottom. Then, aggregation module sends its cluster-profile to its father which generalizes a new cluster-profile and sends the new one to its father again. Finally, a global cluster-profile is generated at root of each HDHT.

3. Evaluation

In this section we present an evaluation of the effectiveness of our CCS malware analysis system. In section 3.1, we describe the data source, a reference clustering generated based on

multi-AV scanning results, and the setup of CCS. Then, in Section 3.2, we compare our solution with centralized system and the reference clustering.

3.1 Experimental Setup

We implemented CCS system based on MIT- Chord, and used the n-gram as the routing attribute. Without live experiment environment for CCS, we deploy CCS in a virtual network based on CORE emulator [15] and dispatch the malware samples to machines randomly. We run CCS system in 10 machines and centralized in 1 machine with same parameters.

We obtained a set of almost 400,000 malware samples from mwanalysis.org [16] in the period from January 16, 2011 to March 21, 2011. We selected only those samples for which three out of five of the AV scanners reported as the same malware family, the number of samples in the family must greater than 20, and the pcap file of sample greater than 1KB. This resulted in a total of 917 samples. Those families are used as a reference clustering. We turn the global cluster-profile to a cluster whose elements are samples that match it.

3.2 Comparing with Centralized System and AV Scanners

We setup three groups of comparison. Group 1: centralized system as reference, estimate the performance of CCS. Group 2: reference clustering as reference, estimate our CCS. Group 3: reference clustering as reference, estimate the centralized system. Table 1 describes the performance of each group using precision, recall (higher the value, better the performance) as metrics. From the table 1, we can see that the performance of CCS is comparable to the performance of centralized system and approximately consistent with reference clustering.

Table 1. Malware Analysis Perfomance of CCS

	precision	*recall*
group 1	***0.98***	***0.83***
group 2	***0.67***	***0.81***
group 3	***0.54***	***0.93***

4. Conclusion

We propose CCS, the first collaborative malware clustering and signature generation framework which is applied for various malware and scalable. Our experiment with 917 real-world malware samples shows that CCS has comparable performance with centralized system, buts better scalability. CCS shows a very promising direction for scalable malware analysis.

Acknowledgements

This work is supported by PCSIRT (No.IRT 1012), and the National Science Foundation of China under Grant No. 61003303.

References

[1] Internet Security Threat Report, Vol. 16. http://www.symantec.com/business/threatreport/.

[2] J. Jang, D. Brumley and S. Venkataraman, "BitShred: Feature Hashing Malware for Scalable Triage and Semantic Analysis", CCS'11, **(2011)** October 17–21; Chicago, Illinois, USA.

[3] A.K. Jain, M. N. Murty and P. J. Flynn, "Data Clustering: A Review", ACM Computing Surveys, Vol. 31, No. 3, **(1999)**.

[4] U. Bayer, P. M. Comparetti, C. S. Hlauschek, C. Kruegel and E. Kirda, "Scalable, Behavior-Based Malware Clustering", NDSS, **(2009)** February 8-11 San Diego, USA.

[5] R. G. Gallager, P. A. Humblet and P. M. Spira, "A Distributed Algorithm for Spanning Trees", ACM Transactions on Programming Languages and Systems, Vol. 5, No. 1, **(1983)**.

[6] R. Perdisci, W. Lee and N. Feamster, "Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces", NDSS'10, **(2010)** February 28- March 3, San Diego.
[7] http://www.honeynet.org/.
[8] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", Computer Networks, 31(23-24), pp. 2435-2463, **(1999)**.
[9] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks", LISA'99, **(1999)** Berkeley, USA.
[10] http://en.wikipedia.org/wiki/Tf*idf.
[11] J. Newsome, B. Karp and D. Song, "Polygraph: Automatic Signature Generation for Polymorphic Worms", IEEE Security and Privacy Symposium, **(2005)** May 8-11, California, USA.
[12] A. Valdes and K. Skinner, "Probabilistic Alert Correlation", RAID, **(2000)** Oct. 2-4, Toulouse, France.
[13] M. Halkidi, Y. Batistakis and M. Vazirgiannis, "On Clustering Validation Techniques", Journal of Intelligent Information Systems, Vol. 17, No. 2-3, pp.107-145, **(2001)**.
[14] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval", Information Processing and Management, 24(5):513-523, **(1988)**.
[15] http://cs.itd.nrl.navy.mil/work/core/.
[16] http://mwanalysis.org/.
[17] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework", IEEE Symposium on Security and Privacy, **(2002)** May 12-15, Oakland, California, USA.
[18] M. Cai, K. Hwang, J. Pan, "WormShield: Fast Worm Signature Generation with Distributed Fingerprint Aggregation", IEEE TDSC, Vol. 4, No. 2, pp. 88-104, **(2007)**.

Authors

Huabiao Lu received the B.S. degree and M.S. degree from the National University of Defense Technology (NUDT) in 2006 and 2008, respectively, all in school of computer. He is a PhD candidate in Institute of Network and Information Security, National University of Defense Technology (NUDT) since March 2009. His current research interests are in malware detection, especially distributed and collaborative malware detection, and distributed computing.

Xiaofeng Wang is an assistant professor in School of Computer, National University of Defense Technology (NUDT), China. He completed his PhD at NUDT in 2009. His current research interests are in trust and security of networking systems, distributed and intelligent date processing. He has published several papers in renowned journals and conferences like IEEE/ACM CCGrid, AINA, IEEE Transactions on Services Computing and Elsevier FGCS etc.

Jinshu Su received the B. Sc degree in Mathematics from Nankai University in 1983, the M.S. degree and PhD degree in Computer Science, NUDT in 1989 and 1999, respectively. He is a full professor, the head of the Institute of network and information security, NUDT, and the academic leader of the State Innovative Research Team in University ("Network Technology") awarded by the Ministry of Education, CHINA. He has lead several national key projects of CHINA, including national 973 projects, 863 projects and NSFC Key projects. His research interests include high performance routers, high performance computing, wireless networks and information security. He is a member of the ACM and IEEE.