International Journal of Security and Its Applications
Vol. 10, No. 4 (2016), pp.249-266
http://dx.doi.org/10.14257/ijsia.2016.10.4.24

An Effective Approach for Classification of Advanced Malware with High Accuracy

Ashu Sharma[1] and Sanjay K. Sahay[2,*]

[1]*Research scholar, Department of Computer Science and Information System,*
[2]*Assistant Professor, Department of Computer Science and Information System,*
[1,2]*Birla Institute of Technology and Science, K. K. Birla Goa Campus, NH-17B, By Pass Road, Zuarinagar-403726, Goa, India*
{[1]p2012011, [2]ssahay}@goa.bits-pilani.ac.in

Abstract

Combating malware is very important for software/systems security, but to prevent the software/systems from the advanced malware, viz. metamorphic malware is a challenging task, as it changes the structure/code after each infection. Therefore in this paper, we present a novel approach to detect the advanced malware with high accuracy by analyzing the occurrence of opcodes (features) by grouping the executables. These groups are made on the basis of our earlier studies [1] that the difference between the sizes of any two malware generated by popular advanced malware kits viz. PS-MPC, G2 and NGVCK are within 5 KB. On the basis of obtained promising features, we studied the performance of thirteen classifiers using N-fold cross-validation available in machine learning tool WEKA. Among these thirteen classifiers we studied in-depth top five classifiers (Random forest, LMT, NBT, J48 and FT) and obtain more than 96.28% accuracy for the detection of unknown malware, which is better than the maximum detection accuracy (~95.9%) reported by Santos et al (2013). In these top five classifiers, our approach obtained a detection accuracy of ~97.95% by the Random forest.

Keywords: *Anti-Malware, Static Analysis, WEKA, Machine Learning, Decision Tree*

1. Introduction

"Malware refers to a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system or of otherwise annoying or disrupting the victim" [2]. From the last four decades, malware is continuously evolving with high complexity to evade the available detection technique. These malware are basically classified as first and second generation malware. In the first generation, structure of the malware does not change, while in the second generation, structure changes to generate a new variant, keeping the action same [3]. On the basis of how variances are created in malware, second generation malware are further classified into Encrypted, Oligormorphic, Polymorphic and Metamorphic Malware [4]. These malware changes its structure in random and unpredictable ways each time it replicates, hence hard to detect. According to McAfee technical report of 2014, there are more than 200 million known malware samples [5]. The Symantec 2014 Internet Security Threat report states that 2013 was the mega breach year [6] (~62% more breaches then 2012). The F-secure document reported an increase in malware attacks against mobile devices based on Android and Apple iOS [7]. This increase in threat from malware is due to wide spread use of World Wide Web. An estimate shows that the web-based attacks were increased 36% with over 4,500 new

* *ssahay*@goa.bits-pilani.ac.in

ISSN: 1738-9976 IJSIA

attacks each day, annoying/disrupting the victim in terms of confidentiality, integrity, availability of the victims data *etc*. [8]. The malware attacks/threat are not only limited to individual level, but there are state sponsored highly skilled hackers developing customized malware to disrupt industries and for military espionage [9]. Such attacks can alter the operation of industrial systems, disrupt power plants, *e.g*. the StuxNet and Duqu malware [10]. The intrusion into Google's systems demonstrates how well-organized attacks are designed to maintain long-term access to an organization's network [11].

To combat threats/attacks from the malware, signature-based software (anti-malware) were widely deployed. However, its an indisputable fact that these traditional approach of combating the threats/attack with a technology-centric are ineffective to detect today's highly sophisticated customized malware. Hence attacks from such malware to the computing world are increasing every day. The consequence will be devastating if in time adequate measures had not been taken. Therefore, there is a need that both academia and anti-malware developers should continually work to combat the threats/attacks from the evolving malware. The most popular techniques used for the detection of malware are signature matching, heuristics-based detection, malware normalization, machine learning, *etc*. [4]. In recent years, machine learning techniques are studied by many authors and proposed different approaches [12] [13] [14], which can supplement traditional anti-malware system. For the detection of malware by machine learning technique, feature selection plays a vital role. In the literature, many feature selection approaches are discussed viz. Olivier Henchiri *et al*. 2006 [15], Siddiqui *et al*. 2008 [16], B. Mehdi *et al*. 2009 [17] and Santos *et al*. 2013 [18] used hierarchical, unary variable removal method, Goodness evaluator and Weighted Term Frequency (WTF) respectively for the feature selection. The maximum accuracy they obtained was 95.26%. In this paper, our approach outperforms the accuracy obtained by these authors by more than ~2%.

The paper is organized as follow, in next section related work is discussed and in section 3 we present our approach, The section 4 discuss the experimental results and finally section 5 contains the conclusion of the paper.

2. Related Work

The first virus was created in 1970 [19] and since then there is a strong contest between the attackers and defenders. This rat-race led to the improvement in both malware and its detection techniques. To defend the malware attacks, anti-malware groups are developing new techniques. On the other hand, malware developers are adopting new tactics/methods to avoid the malware detectors. Initially, the tools and techniques of malware analysis were in the domain of anti-malware vendors. However, the use of malware for espionage, sophisticated cyber-attacks and other crimes motivated the academicians and digital investigators to develop an advanced method to combat the threats/attacks from it. In the year 2001, Schultz *et al*. [20] were the first to introduce the concept of data mining for detecting malware. They used three different static features for malware classification (Portable Executable, strings and byte sequences). Kolter and Maloof (2004) evaluated data sets using Instance-Based Learning Algorithms (IBK), TF-IDF, Naive Bayes, Support Vector Machine (SVM) and Decision tree [21]. Among the classifiers used by them, Decision tree outperformed. In the year 2005, Karim *et al*. [22] addressed the tracking of malware evolution based on opcode sequences and permutations. O. Henchiri *et al*. (2006) proposed a hierarchical feature extraction algorithm and used ID3, j48, Naive Bayes and SMO classifier and obtained the maximum of 92.56% accuracy [15]. In the year 2007, Bilar uses small dataset (67 malware and 20 benign samples) to examine the difference in opcode frequency distribution between malicious and benign programs [23]. He found that malware opcode distribution differs significantly from benign programs and also observed that some opcodes

seen to be a stronger predictor of frequency variation. He however, did not apply it for the classification of advanced malware. In the year 2008, Tian *et al*. particularly classified Trojan malware using function length frequency [24]. Their results indicate that the function length along with its frequency is significant in identifying malware family and can be combined with other features for fast and scalable malware classification. In the year 2008, Siddiqui *et al*. [16] used variable length instruction sequence for detecting worms in the wild. They tested their method on a data set of 2774 (1444 worms and 1330 benign files) and got 95.6% detection accuracy. In the year 2008, Moskovitch *et al*. [25] [26] compared the different classifiers by byte-sequence n-grams (3, 4, 5 or 6). Among the classifiers they studied BDT, DT and ANN outperformed NB, BNB and SVM classifiers, exhibiting lower false positive rates. S. Momina Tabish (2009) used AdaBoostM1 algorithm for classification by taking n-gram frequency as a feature and reported 90% detection accuracy [27]. In the year 2010, Bilal Mehdi et. al. [28] used hyper-grams (generalized n-gram) and obtained 87.85% detection accuracy and claimed no false alarm. Santos *et al*. in the year 2011 pointed out that supervised learning requires a significant amount of labeled executables for both malware and benign programs, which is difficult to obtain, hence they proposed a semi-supervised learning method for detecting unknown malware, which does not require a large amount of labeled data [29]. They obtained 86% of accuracy by labeling only 50% of the selected data set. In the subsequent paper [18] in 2013, they used Term Frequency for modeling different classifiers and found that SVM outperforms with accuracy of 92.92% and 95.90% respectively for one opcode and two opcode sequence length respectively.

Recently in 2014, Kevin Allix *et al*. [12] took a size-able dataset of over 50,000 android applications and implemented using 4 well-known machine learning algorithms (RandomForest, J48, JRip and LibSVM) with ten-fold cross-validation. He claimed his approach outperformed existing machine learning approaches, however on usual size datasets performance does not translate in the wild.

3. Our Approach

In order to uncover the unknown malware with high accuracy, our novel approach as shown in Figure 1 involves finding the promising features (Algo. 1), training of classifiers and detection of unknown malware.

3.1. Building the Datasets and Feature Selection

To build the datasets, we downloaded 11088 malware from malacia-project [30] and collected 4006 benign programs (cross checked from virustotal.com [31]) from different systems.

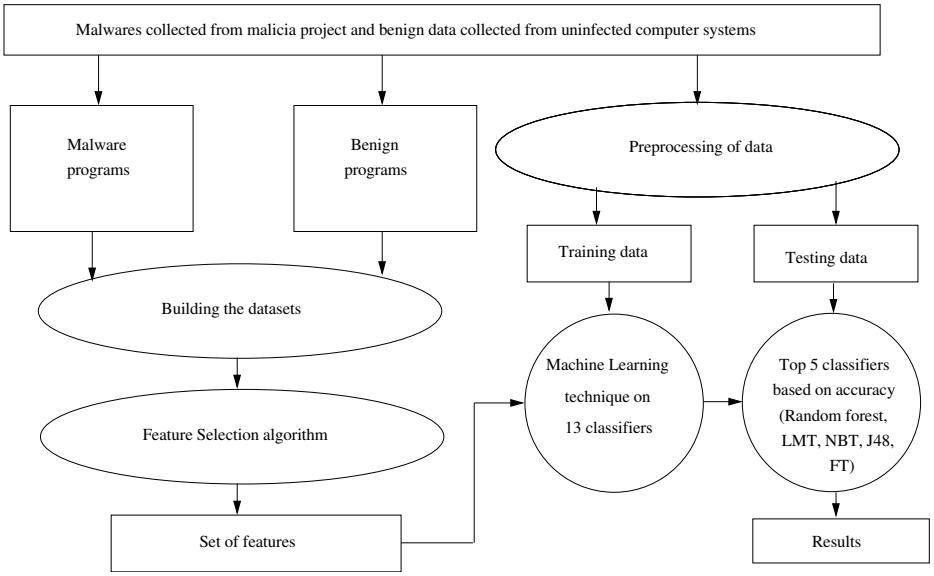

Figure 1. Flow Chart for the Detection of Unknown Malware

Algorithm 1: Feature Selection

INPUT: Pre-processed data of all groups,
$\mathbf{N_b}\rightarrow$ Number of benign executables and $\mathbf{N_m}\rightarrow$ Number of malware executables
OUTPUT: List of features.

BEGIN
for all groups $\mathbf{G_K}$ **do**
for all i_{th} benign executable in the group $\mathbf{G_K}$ **do**
Compute the normalized frequency F_{K_b} of each opcode **oj**

$$F_{K_b}(o_j) = \left(\quad f_i(o_j) \right) / N_b$$

end for
for all i_{th} malware executable in the group $\mathbf{G_K}$ **do**
Compute the normalized frequency F_{K_m} of each opcode **oj**

$$F_{K_m}(o_j) = \left(\quad f_i(o_j) \right) / N_m$$

end for
for all opcode $\mathbf{o_j}$ **do**
Subtract the frequencies F_{k_b} and F_{k_m}.

$$D_K(o_j) = \left| F_{K_b}(o_j) \quad F_{K_m}(o_j) \right|$$

end for
Sort the obtained $\mathbf{D_K(o)}$.

end for
Set a threshold on $\mathbf{D_k(o)}$ to select the promising opcodes features such that from each $\mathbf{G_k}$ at least 10 opcodes get selected.

return Union of the selected features of all the groups.

The promising features of the executables are obtained by clubbing the dataset in 5 KB size of 100 groups [1] as in the collected dataset ~97.18% malware are below 500 KB (Figure 2) and the difference between the sizes of any two malware generated by popular advanced malware kits viz. NGVCK [32], PS-MPC [33] and G2 [34] are within 5 KB. Hence, the features obtained will have a signature of maximum executables to detect the unknown malware. Our features are opcodes of the executables obtained by *objdump* utility available in the Linux system. To identify the each opcode we labeled it with a fixed integer as given in the Appendix A. To differentiate malware and benign programs we obtained the features as given in algo. 1.

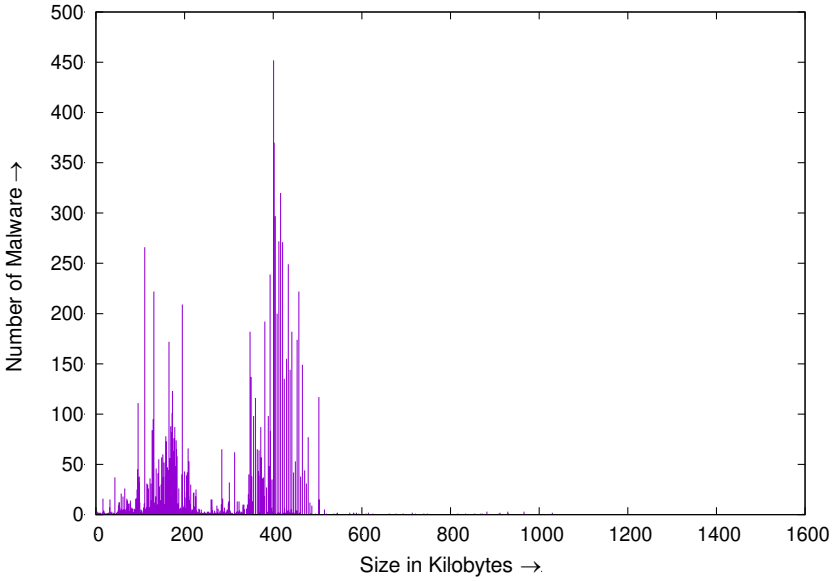

Figure 2. Number of Malware with Respect to File Size

3.2. Training of the Classifiers

To find the best classifiers for detection of unknown malware, we investigated thirteen tree based classifiers viz. Random forest, J48, REPTREE, LMT, Decision stump, ADT, NBT, FT, LAD, Random Tree, Simple CART, BFT and J48 Graft available in the popular and widely used suite of machine learning software known as *WEKA* (a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality). Then with the obtained features, we run the *WEKA* n-fold cross-validation to train all the selected classifiers. Figure 3 shows the accuracy obtained by all classifiers for n = 2,4,6...,16 folds. We observed that Random forest is the best classifier and its accuracy is almost flat after n = 2. Rest twelve classifiers accuracy fluctuates, however after ten-fold cross-validation the fluctuations are least and we observe maximum correctness in the accuracy.

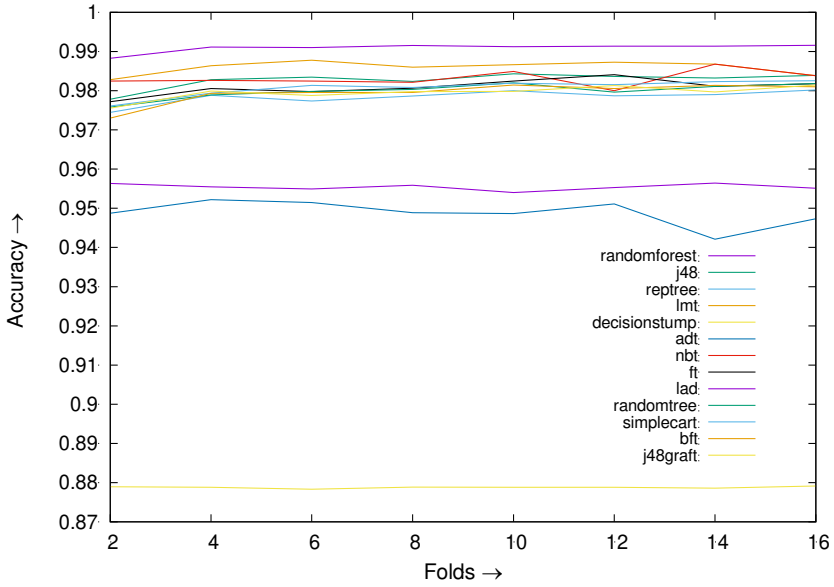

Figure 3. Accuracy of the Thirteen Classifiers with N-Fold Cross Validation

3.3 Detection of Unknown Malware

Among the thirteen studied classifier, we selected top five (Random forest, LMT, NBT, J48 and FT) for in-depth analysis. To study the overall performance of these classifiers, we randomly selected 750 malware and 610 benign programs from all the groups, such that at least five executables from each group can be randomly tested by the trained classifiers with ten-fold cross-validation for the detection of unknown malware.

4. Experimental Results

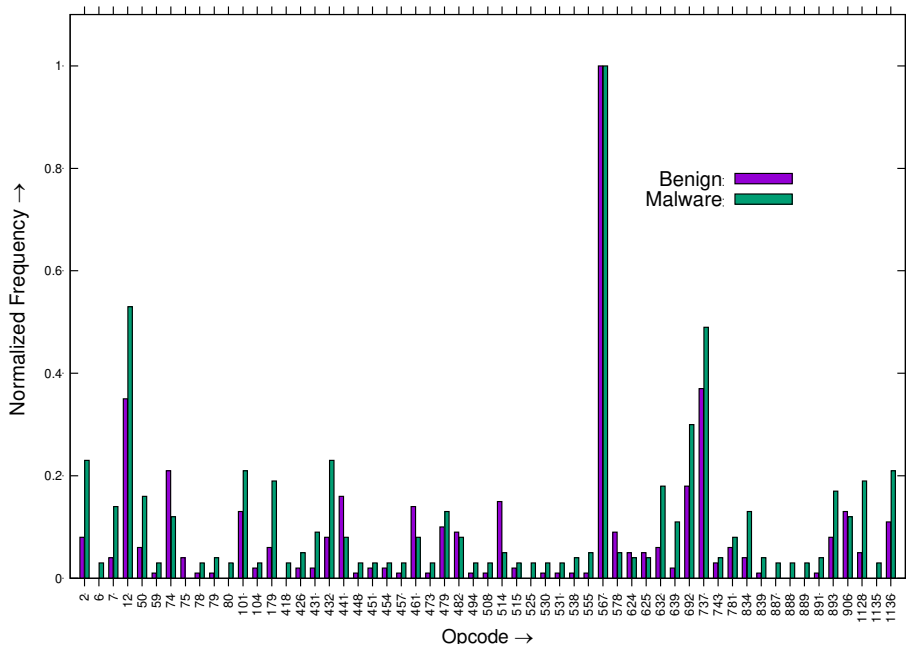

Figure 4. Normalized Opcode Occurrence of all the Collected Malware and Benign Program Keeping Threshold 0.02

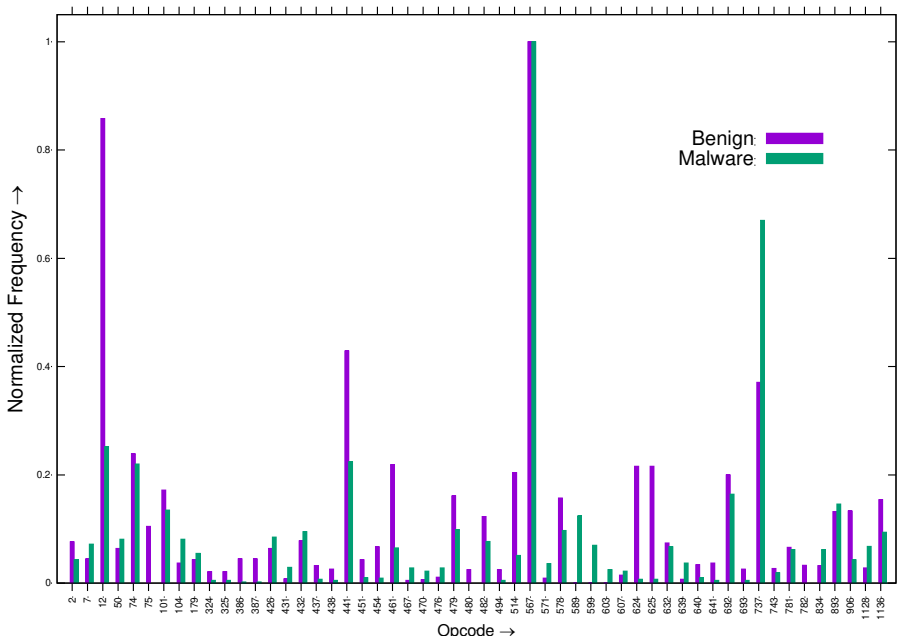

Figure 5. Normalized Opcode Occurrence of the Malware and Benign Program of Size 10-15 KB Keeping Threshold 0.02

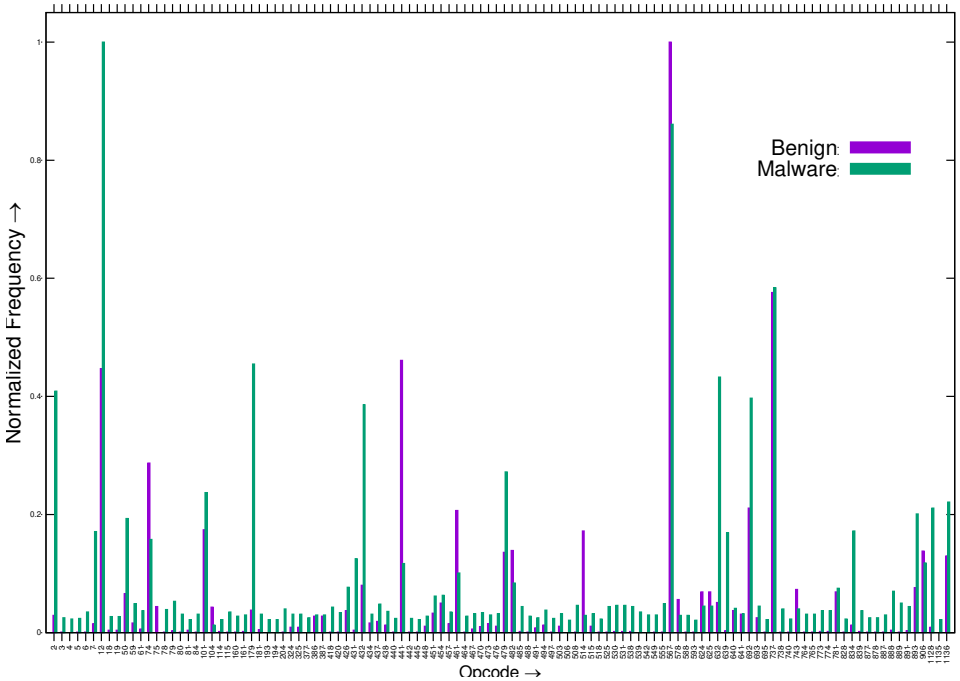

Figure 6. Normalized Opcode Occurrence of the Malware and Benign Program of Size 140-145 KB Keeping Threshold 0.02

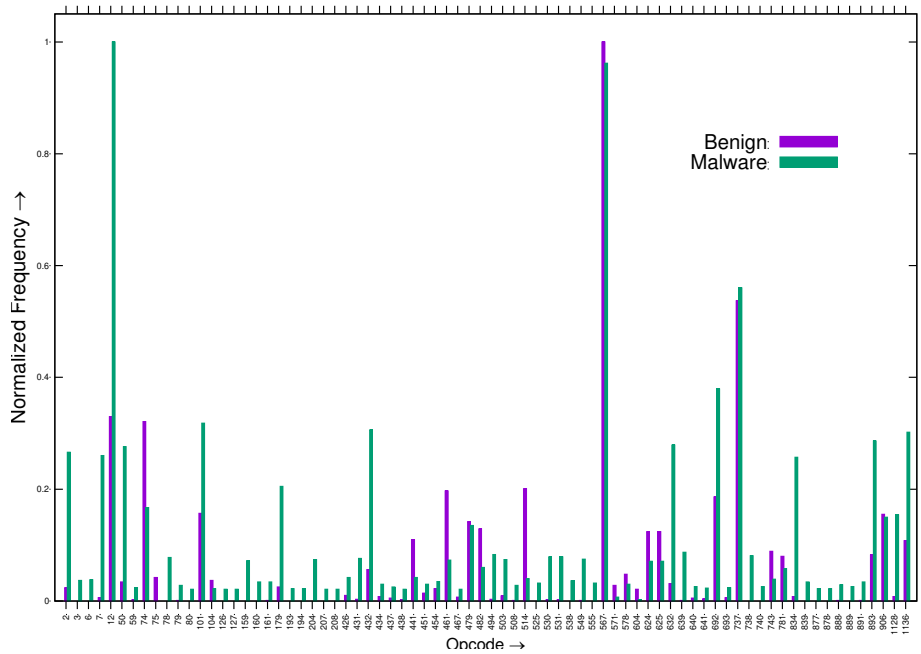

Figure 7. Normalized Opcode Occurrence of the Malware and Benign Program of size 240-245 KB Keeping Threshold 0.02

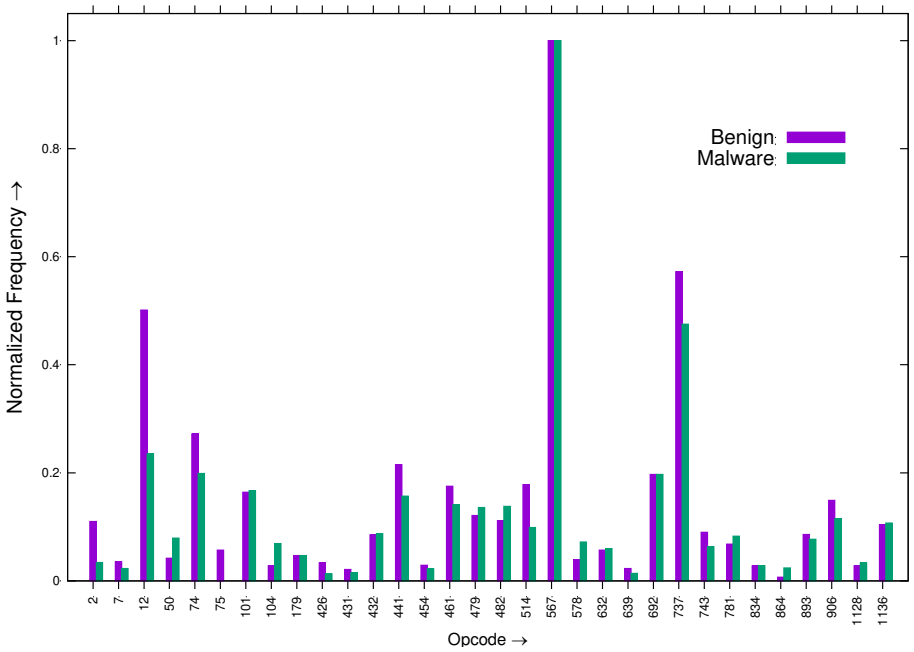

Figure 8. Normalized Opcode Occurrence of the Malware and Benign Program of size 415-420 KB Keeping Threshold 0.02

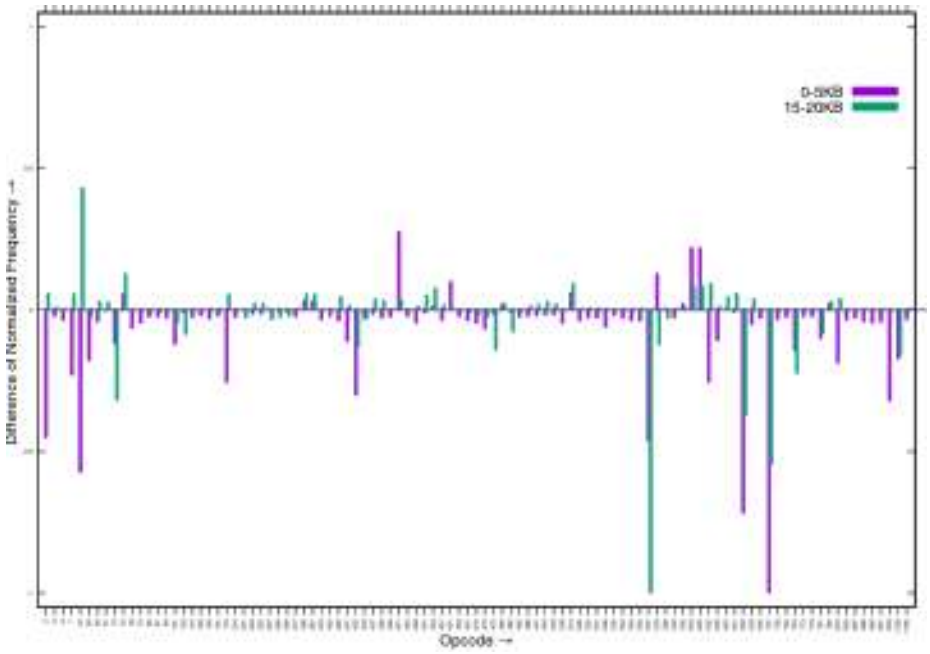

Figure 9. Difference in the Occurrence of Respective Opcodes between Benign and Malware Program of Size 0-5 KB and 15-20 KB Keeping Threshold 0.02

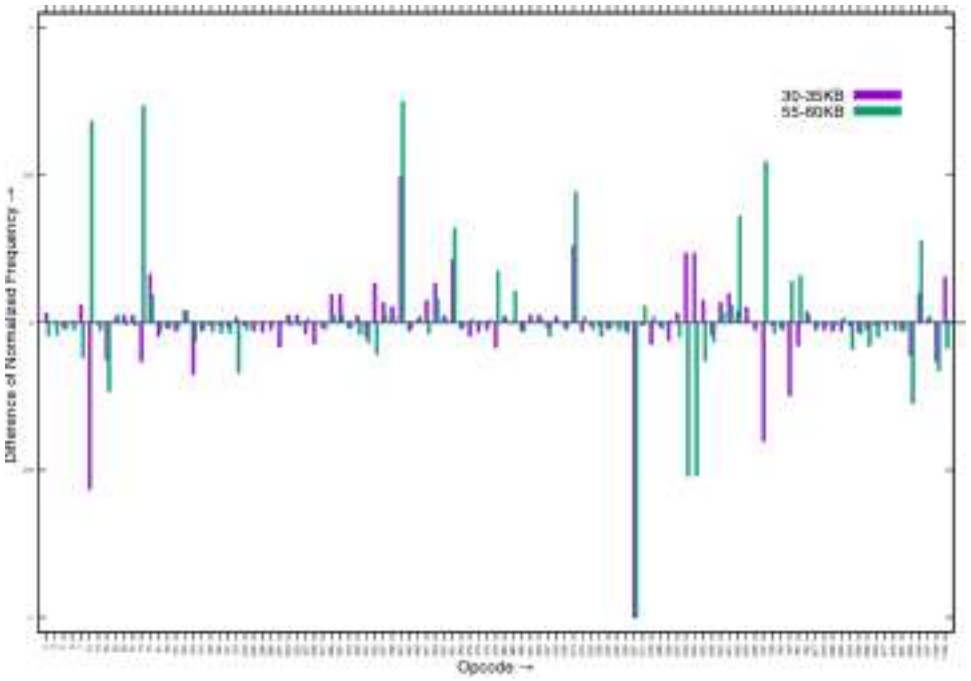

Figure 10. Difference in the Occurrence of Respective Opcodes between Benign and Malware Program of Size 30-35 KB and 55-60 KB Keeping Threshold 0.02

As discussed in section 3, we disassembled all the malware and benign programs of below 500 KB size. Then we computed the normalized opcode occurrence of all malware and benign programs, for each group separately. We observed that the opcode occurrence in the malware and benign programs differ in large. Fig. 4 shows the normalized opcode occurrence of all the malware and benign programs and Fiureg 5, 6, 7, & 8 shows the

normalized opcode occurrence for the group 10-15, 140-145, 240-245 & 415-420 KB size respectively keeping the lower threshold 0.02. To find the dominant features of malware and benign programs we computed the difference in opcode occurrence between the benign and malware programs of each group and found that dominant opcodes vary from group to group. Figure 9. shows the difference in the occurrence of respective opcodes between benign and malware program of size 0-5 KB and 15-20 KB size keeping the lower threshold 0.02. Similarly, Figure 10 is plotted for the size 30-35 and 55-60 KB size of the data set. In the Figure 9, 10 upper side shows the opcodes that dominate in benign and lower side shows the opcode that dominates in malware.

The effectiveness of the top five classifiers viz. Random forest Tree, LMT, NBT, J48 and FT has been studied with the randomly selected 750 malware and 610 benign programs. The analysis are done in *WEKA* with ten-fold cross-validation, in terms of True Positive Ratio (TPR), True Negative Ratio (TNR), False Positive Ratio (FPR), False Negative Ratio (FNR) and accuracy, defined as

$$TPR=\frac{TP}{TM};\quad TNR=\frac{TN}{TB};\quad FPR=\frac{FP}{TB};\quad FNR=\frac{FN}{TM}$$

$$Accuracy=\frac{TP+TN}{TM+TB}\times 100$$

where,

TP⟶ True positive, the number of malware correctly classified
TN⟶ True negative, the number of benign correctly classified.
FP⟶ False positive, the number of benign detected as malware.
FN⟶ False negative, the number of malware detected as benign.
TM⟶ Total number of malware.
TB⟶ Total number of benign.

From the analysis, it is clear that Random forest is the best classifier for identification of unknown malware. Nevertheless, the other classifiers are also reasonably good (> 96.2%) for the detection of unknown malware. The detail results obtained are shown in table 1.

Table 1. Performance of the Top 5 Classifiers

Classifiers	True positive	False negative	False positive	True Negative	Accuracy
Random forest	739 (98.53%)	11 (1.47%)	6 (2.81%)	554 (97.19%)	97.95%
LMT	734 (97.87%)	16 (2.13%)	23 (4.04%)	547 (95.96%)	97.04%
NBT	728 (97.07%)	22(2.93%)	19 (3.33%)	551 (96.67%)	96.89%
J48	729 (97.2%)	21 (2.8%)	23 (4.04%)	547 (95.96%)	96.66%
FT	729 (97.2%)	21 (2.8%)	8 (4.91%)	542 (95.09%)	96.28%

We found that the classifier NBT, J48 and FT have the almost same True positive ratio. In these, the overall accuracy of Functional Tree classifier is lowest, which is basically due to high False positive ratio. Figure 11 shows the variation of False positive and False negative of the studied classifiers. We found that the False positives ratio of Random forest and LMT are almost double than False positive ratio, however for overall accuracy both has to be low. From Figure 12 we observed that the True positives of all classifiers are more biased towards the detection of malware. The obtained accuracy for all five classifiers are shown in Figure 13 and the comparison of our results with Santos *et al*., Siddiqui *et al*., Asaf Shabtai et. al. for Random forest and Mehdi *et al*., Santos *et al*., Olivier Henchiri *et al*. for J48 are shown in Figure 14. Among these authors our approach uncover the malware with the best accuracy (~97.95%).

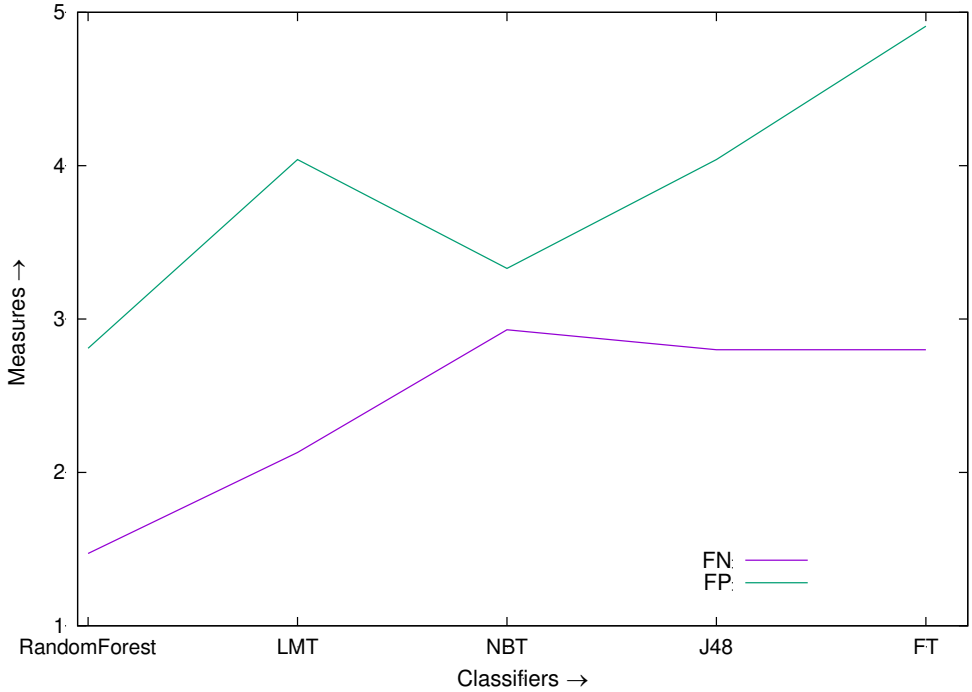

Figure 11. FP and FN of Top Five Classifiers

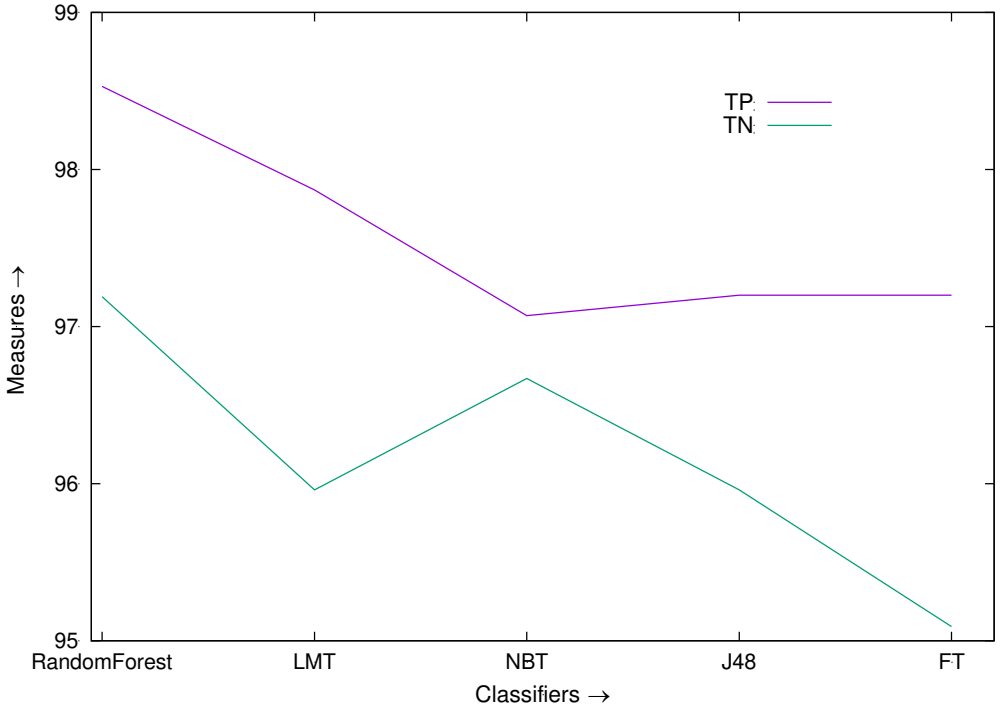

Figure 12. TP and TN of Top Five Classifiers

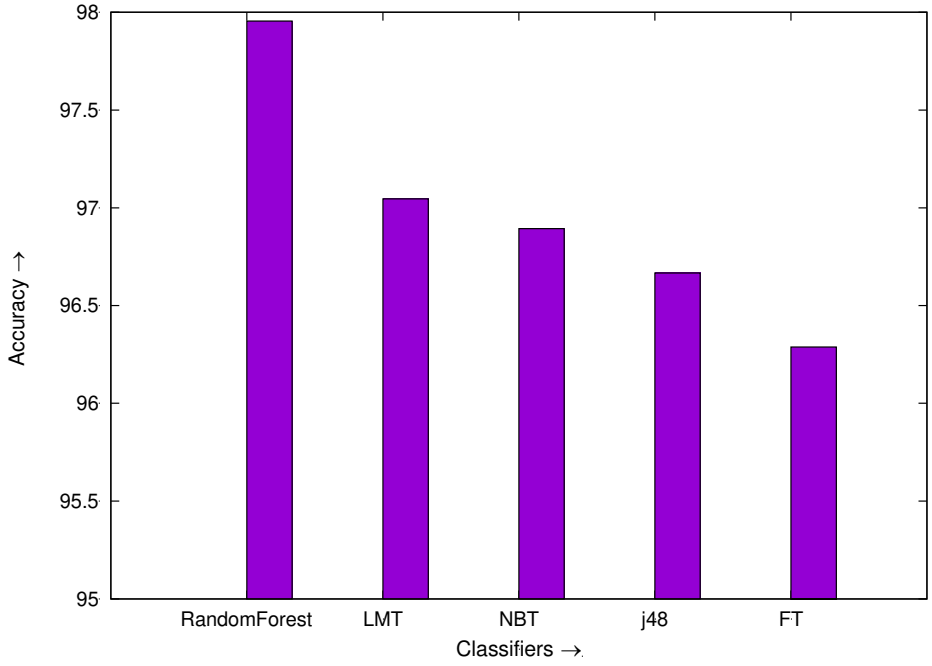

Figure 13. Accuracy of the Top Five Classifiers

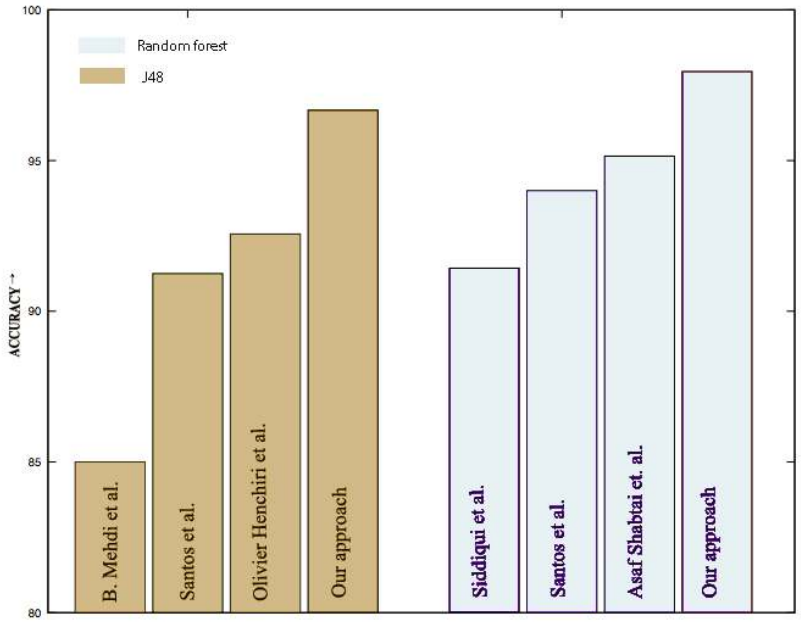

Figure 14. Comparison of the Accuracy Obtained by our Approach and Others

5. Conclusion

Traditional approach *i.e*. updating the signature database to combat advanced malware is ineffective. Therefore, in this paper, we presented a novel approach to detect the advanced malware with high accuracy. For the classification, we obtained the promising features (opcodes) by grouping the executables in 5 KB size. Extensive experiment has been done to study the performance of the classifiers viz. Random forest, LMT, NBT, J48 and FT in terms of TPR, TNR, FPR, FNR and accuracy by analyzing 11688 malware

downloaded from malicia-project and 4006 benign programs collected from different systems. By our approach all five classifiers are able to uncover unknown malware with greater than 96.28% accuracy, which is better than the detection accuracy (~95.9%) reported by Santos et. al. (2013). Among these classifiers, we found that Random forest is the best (~97.95%) classifier to detect the unknown malware. Thus, our approach outperforms to detect the unknown malware and hence can be an effective technique to complement the signature based mechanism or dynamic analysis. In future, we will collect more malware and benign and will perform in-depth size analysis for the classification of unknown malware.

Appendix

ID	Opcode	ID	Opcode	ID	Opcode	ID	Opcode	ID	Opcode
1	.byte	231	fcmovnbe	461	je	691	pmuludq	921	vaddsd
2	(bad)	232	fcmovne	462	je,pn	692	pop	922	vaddsubpd
3	aaa	233	fcmovnu	463	je,pt	693	popa	923	vaddsubps
4	aad	234	fcmovu	464	jecxz	694	popaw	924	vaesdec
5	aam	235	fcom	465	jecxz,pn	695	popf	925	vandnpd
6	aas	236	fcomi	466	jecxz,pt	696	popfq	926	vandnps
7	adc	237	fcomip	467	jg	697	popfw	927	vandpd
8	adcb	238	fcoml	468	jg,pn	698	popl	928	vandps
9	adcl	239	fcomp	469	jg,pt	699	popq	929	vcmpltpd
10	adcq	240	fcompl	470	jge	700	popw	930	vcmpngess
11	adcw	241	fcompp	471	jge,pn	701	por	931	vcmppd
12	add	242	fcomps	472	jge,pt	702	prefetch	932	vcmpps
13	addb	243	fcoms	473	jl	703	prefetchnta	933	vcmpsd
14	addl	244	fcos	474	jl,pn	704	prefetcht0	934	vcmpss
15	addpd	245	fdecstp	475	jl,pt	705	prefetcht1	935	vcomisd
16	addps	246	fdiv	476	jle	706	prefetcht2	936	vcomiss
17	addq	247	fdivl	477	jle,pn	707	prefetchw	937	vcvtdq2pd
18	addr16	248	fdivp	478	jle,pt	708	psadbw	938	vcvtdq2ps
19	adcx	249	fdivr	479	jmp	709	pshufb	939	vcvtpd2dq
20	addr32	250	fdivrl	480	jmpq	710	pshufd	940	vcvtpd2dqx
21	addsubps	251	fdivrp	481	jmpw	711	pshuflw	941	vcvtpd2dqy
22	addr32csrex.B	252	fdivrs	482	jne	712	pshufw	942	vcvtpd2psx
23	addr32csrex.RXB	253	fdivs	483	jne,pn	713	pslld	943	vcvtps2dq
24	addr32fsrex.RX	254	femms	484	jne,pt	714	psllq	944	vcvtps2pd
25	addr32gsrex.R	255	ffree	485	jno	715	psllw	945	vcvtps2ph
26	addr32gsrex.RXB	256	ffreep	486	jno,pn	716	psrad	946	vcvtsd2si
27	addr32gsrex.W	257	fiadd	487	jno,pt	717	psraw	947	vcvtsd2ss
28	addr32gsrex.WRB	258	fiaddl	488	jnp	718	psrld	948	vcvtsi2sd
29	addr32gsrex.WRX	259	ficom	489	jnp,pn	719	psrldq	949	vcvtsi2ssl
30	addr32rex	260	ficoml	490	jnp,pt	720	psrlq	950	vcvtss2sd
31	addr32rex.B	261	ficomp	491	jns	721	psrlw	951	vcvtss2si
32	addr32rex.RX	262	ficompl	492	jns,pn	722	psubb	952	vcvttpd2dq
33	addr32rex.RXB	263	fidiv	493	jns,pt	723	psubd	953	vcvttpd2dqy
34	addr32rex.WR	264	fidivl	494	jo	724	psubq	954	vcvttps2dq
35	addr32rex.WRB	265	fidivr	495	jo,pn	725	psubsb	955	vcvttsd2si
36	addr32rex.WRX	266	fidivrl	496	jo,pt	726	psubsw	956	vcvttss2si
37	addr32rex.WXB	267	fild	497	jp	727	psubusb	957	vdivpd
38	addr32rex.X	268	fildl	498	jp,pn	728	psubusw	958	vdivps
39	addr64	269	fildll	499	jp,pt	729	psubw	959	vdivsd
40	addsd	270	fimul	500	jrcxz	730	punpckhbw	960	vdivss
41	addss	271	fimull	501	jrcxz,pn	731	punpckhdq	961	vdpps
42	addsubpd	272	fincstp	502	jrcxz,pt	732	punpckhqdq	962	verr
43	addw	273	finit	503	js	733	punpckhwd	963	verw
44	aesdec	274	fist	504	js,pn	734	punpcklbw	964	vfmadd213pd
45	aesdeclast	275	fistl	505	js,pt	735	punpckldq	965	vfmadd231pd
46	aesenc	276	fistp	506	lahf	736	punpcklwd	966	vfmadd231ss

47	aesenclast	277	fistpl	507	lar	737	push	967	vfnmsubpd
48	aesimc	278	fistpll	508	lcall	738	pusha	968	vfrczsd
49	aeskeygenassist	279	fisttp	509	lcallq	739	pushaw	969	vfrczss
50	and	280	fisttpl	510	lcallw	740	pushf	970	vgatherqpd
51	andb	281	fisttpll	511	lddqu	741	pushfq	971	vhaddpd
52	andl	282	fisub	512	ldmxcsr	742	pushfw	972	vhaddps
53	andnpd	283	fisubl	513	lds	743	pushl	973	vhsubpd
54	andnps	284	fisubr	514	lea	744	pushq	974	vhsubps
55	andpd	285	fisubrl	515	leave	745	pushw	975	vlddqu
56	andps	286	fld	516	leaveq	746	pxor	976	vmaskmovdqu
57	andq	287	fld1	517	leavew	747	rcl	977	vmaxpd
58	andw	288	fldcw	518	les	748	rclb	978	vmaxps
59	arpl	289	fldenv	519	lfs	749	rcll	979	vmaxsd
60	blcfill	290	fldenvs	520	lgdt	750	rclq	980	vmaxss
61	bound	291	fldl	521	lgdtl	751	rclw	981	vminpd
62	bsf	292	fldl2e	522	lgs	752	rcpps	982	vminps
63	bsr	293	fldl2t	523	lidt	753	rcpss	983	vminsd
64	bswap	294	fldlg2	524	lidtl	754	rcr	984	vminss
65	bt	295	fldln2	525	ljmp	755	rcrb	985	vmload
66	btc	296	fldpi	526	ljmpq	756	rcrl	986	vmmcall
67	btcl	297	flds	527	ljmpw	757	rcrq	987	vmovapd
68	btl	298	fldt	528	lldt	758	rcrw	988	vmovaps
69	btr	299	fldz	529	lmsw	759	rdmsr	989	vmovd
70	btrl	300	fmul	530	loopew,pt	760	rdpmc	990	vmovddup
71	bts	301	fmull	531	lock	761	rdtsc	991	vmovdqa
72	btsl	302	fmulp	532	lockrex	762	rep	992	vmovdqu
73	btsq	303	fmuls	533	lockrex.B	763	repe	993	vmovhpd
74	call	304	fnclex	534	lockrex.WB	764	repne	994	vmovhps
75	callq	305	fndisi(8087	535	lockrex.WR	765	repnz	995	vmovlpd
76	callw	306	fneni(8087	536	lockrex.X	766	repnzrex.R	996	vmovlps
77	cbtw	307	fninit	537	lockrex.XB	767	repnzrex.RX	997	vmovmskpd
78	clc	308	fnop	538	lods	768	repnzrex.RXB	998	vmovmskps
79	cld	309	fnsave	539	loop	769	repnzrex.W	999	vmovntdq
80	cli	310	fnsaves	540	loop,pn	770	repnzrex.WRX	1000	vmovntpd
81	cltd	311	fnsetpm(287	541	loop,pt	771	repnzrex.WRXB	1001	vmovntps
82	cltq	312	fnstcw	542	loope	772	repnzrex.XB	1002	vmovq
83	clts	313	fnstenv	543	loope,pn	773	repz	1003	vmovsd
84	cmc	314	fnstenvs	544	loope,pt	774	rueu	1004	vmovshdup
85	cmova	315	fnstsw	545	loopel	775	repzcsrex.XB	1005	vmovsldup
86	cmovae	316	fpatan	546	loopew	776	repzrex.WRB	1006	vmovss
87	cmovb	317	fprem	547	loopew,pn	777	repzrex.WX	1007	vmovupd
88	cmovbe	318	fprem1	548	loopl	778	repzrex.WXB	1008	vmovups
89	cmove	319	fptan	549	loopne	779	repzrex.X	1009	vmptrld
90	cmovg	320	frndint	550	loopne,pn	780	repzrex.XB	1010	vmptrst
91	cmovge	321	frstor	551	loopne,pt	781	ret	1011	vmread
92	cmovl	322	frstors	552	loopnel	782	retq	1012	vmulpd
93	cmovle	323	frstpm(287	553	loopnew	783	retw	1013	vmulps
94	cmovne	324	fs	554	loopw	784	rex	1014	vmulsd
95	cmovno	325	fstp1	555	lret	785	retnw	1015	vmulss
96	cmovnp	326	fsave	556	lretq	786	rg	1016	vmwrite
97	cmovns	327	fscale	557	lretw	787	rgb	1017	vorpd
98	cmovo	328	fsfsrex	558	lsl	788	rex.R	1018	vorps
99	cmovp	329	fsfsrex.RX	559	lss	789	rex.R	1019	vpackssdw
100	cmovs	330	fsfsrex.WRB	560	ltr	790	rex.RB	1020	vpacksswb
101	cmp	331	fsfsrex.WXB	561	maskmovq	791	rex.RB	1021	vpackuswb
102	cmpb	332	fsgsrex.W	562	maxps	792	rex.RX	1022	vpaddb
103	cmpeqsd	333	fsgsrex.WB	563	minpd	793	rglpsz	1023	vpaddd
104	cmpl	334	fsgsrex.WR	564	minps	794	rex.RXB	1024	vpaddq
105	cmpltpd	335	fsin	565	minss	795	rguid	1025	vpaddsb
106	cmpltsd	336	fsincos	566	montmul	796	rex.W	1026	vpaddusb
107	cmpltss	337	fsqrt	567	mov	797	riid	1027	vpaddusw

108	cmpneqpd	338	fsrex	568	movabs	798	rex.WB	1028	vpaddw
109	cmpneqps	339	fsrex.B	569	movapd	799	rorx	1029	vpand
110	cmpnlepd	340	fsrex.R	570	movaps	800	rex.WR	1030	vpandn
111	cmppd	341	fsrex.RB	571	movb	801	roundpd	1031	vpavgb
112	cmpps	342	fsrex.RX	572	movd	802	rex.WRB	1032	vpavgw
113	cmpq	343	fsrex.RXB	573	movdq2q	803	roundps	1033	vpcmpeqb
114	cmpsb	344	fsrex.W	574	movdqa	804	rex.WRX	1034	vpcmpeqd
115	cmpsl	345	fsrex.WB	575	movdqu	805	roundsd	1035	vpcmpeqw
116	cmpsq	346	fsrex.WR	576	movhlps	806	rex.WRXB	1036	vpcmpgtb
117	cmpsw	347	fsrex.WRB	577	movhps	807	rsldt	1037	vpcmpgtd
118	cmpunordps	348	fsrex.WRX	578	movl	808	rex.WX	1038	vpcmpgtw
119	cmpw	349	fsrex.WRXB	579	movlhps	809	rsqrtss	1039	vpcomud
120	cmpxchg	350	fsrex.WX	580	movlpd	810	rex.WXB	1040	vpcomw
121	cmpxchg8b	351	fsrex.WXB	581	movlps	811	rsts	1041	vpextrw
122	comisd	352	fsrex.XB	582	movmskps	812	rex.X	1042	vphaddbd
123	comiss	353	fst	583	movntdq	813	rex.RXB	1043	vphaddbw
124	cpuid	354	fstcw	584	movnti	814	rex.XB	1044	vphaddubq
125	cqto	355	fstenv	585	movntps	815	rueu	1045	vphaddubw
126	cs	356	fstl	586	movntq	816	rol	1046	vphadduwq
127	cvtpd2dq	357	fstp	587	movq	817	rolb	1047	vphaddwq
128	csrex.B	358	fstpl	588	movsb	818	roll	1048	vphsubbw
129	csrex.R	359	fstps	589	movsbl	819	rolq	1049	vphsubdq
130	csrex.RB	360	fstpt	590	movsbq	820	rolw	1050	vphsubwd
131	csrex.RXB	361	fsts	591	movsbw	821	ror	1051	vpinsrw
132	csrex.W	362	fstsw	592	movsd	822	rorb	1052	vpmacsdqh
133	csrex.WB	363	fsub	593	movsl	823	rorl	1053	vpmacsdql
134	csrex.WR	364	fsubl	594	movsldup	824	rorq	1054	vpmacssdql
135	csrex.WRB	365	fsubp	595	movslq	825	rorw	1055	vpmacsswd
136	csrex.WRX	366	fsubr	596	movsq	826	rsm	1056	vpmacssww
137	csrex.WX	367	fsubrl	597	movss	827	rsqrtps	1057	vpmacswd
138	csrex.WXB	368	fsubrp	598	movsw	828	sahf	1058	vpmaddwd
139	csrex.X	369	fsubrs	599	movswl	829	sar	1059	vpmaxsw
140	csrex.XB	370	fsubs	600	movswq	830	sarb	1060	vpmaxub
141	cvtdq2pd	371	ftst	601	movupd	831	sarl	1061	vpminsw
142	cvtdq2ps	372	fucom	602	movups	832	sarq	1062	vpminub
143	cvtpd2ps	373	fucomi	603	movw	833	sarw	1063	vpmovmskb
144	cvtpi2ps	374	fucomip	604	movzbl	834	sbb	1064	vpmulhuw
145	cvtps2pd	375	fucomp	605	movzbq	835	sbbb	1065	vpmulhw
146	cvtps2pi	376	fucompp	606	movzbw	836	sbbl	1066	vpmulld
147	cvtsd2si	377	fwait	607	movzwl	837	sbbq	1067	vpmullw
148	cvtsd2ss	378	fxam	608	movzwq	838	sbbw	1068	vpmuludq
149	cvtsi2sd	379	fxch	609	movzww	839	scas	1069	vpor
150	cvtsi2sdq	380	fxrstor	610	mul	840	seta	1070	vpperm
151	cvtsi2ss	381	fxsave	611	mulb	841	setae	1071	vprotb
152	cvtss2sd	382	fxtract	612	mull	842	setb	1072	vprotd
153	cvttpd2dq	383	fyl2x	613	mulpd	843	setbe	1073	vpsadbw
154	cvttpd2pi	384	fyl2xp1	614	mulps	844	sete	1074	vpshab
155	cvttps2pi	385	getsec	615	mulq	845	setg	1075	vpshld
156	cvttsd2si	386	gs	616	mulsd	846	setge	1076	vpshlw
157	cvttss2si	387	gdiplus	617	mulss	847	setl	1077	vpshufb
158	cwtd	388	gsaddr32rex.R	618	mulw	848	setle	1078	vpshufhw
159	cwtl	389	gsaddr32rex.WXB	619	neg	849	setne	1079	vpshuflw
160	daa	390	gscsrex.RXB	620	negb	850	setno	1080	vpslld
161	das	391	gsdata16rex	621	negl	851	setnp	1081	vpsllq
162	data16	392	gsdsrex.R	622	negq	852	setns	1082	vpsllw
163	data16addr32rex.WRB	393	gsdsrex.RXB	623	negw	853	seto	1083	vpsrad
164	data16data16rex.R	394	gsdsrex.WRB	624	nop	854	setp	1084	vpsraw
165	data16data16rex.WXB	395	gsdsrex.WRX	625	nopq	855	sets	1085	vpsrld
166	data16gsrex.RXB	396	gsesrex.W	626	nopl	856	sgdt	1086	vpsrlq
167	data16gsrex.WRB	397	gsfsrex.B	627	nopw	857	sgdtl	1087	vpsrlw
168	data16rex	398	gsfsrex.RB	628	not	858	shl	1088	vpsubb

169	data16rex.B	399	gsfsrex.W	629	notb	859	shlb	1089	vpsubq
170	data16rex.R	400	gsfsrex.WR	630	notl	860	shld	1090	vpsubsb
171	data16rex.W	401	gsfsrex.WRB	631	notw	861	shll	1091	vpsubsw
172	data16rex.WB	402	gsgsrex.R	632	or	862	shlq	1092	vpsubusb
173	data16rex.WRB	403	gsgsrex.W	633	orb	863	shlw	1093	vpsubusw
174	data16rex.WRX	404	gsrex	634	orl	864	shr	1094	vpsubw
175	data16rex.WRXB	405	gsrex.B	635	orpd	865	shrb	1095	vpunpckhbw
176	data16rex.WXB	406	gsrex.R	636	orps	866	shrd	1096	vpunpckhdq
177	data16rex.XB	407	gsrex.RB	637	orq	867	shrl	1097	vpunpckhqdq
178	data32	408	gsrex.RXB	638	orw	868	shrq	1098	vpunpckhwd
179	dec	409	gsrex.W	639	out	869	shrw	1099	vpunpcklbw
180	decb	410	gsrex.WB	640	outsb	870	shufps	1100	vpunpckldq
181	decl	411	gsrex.WR	641	outsl	871	sidt	1101	vpunpcklqdq
182	decq	412	gsrex.WRB	642	outsw	872	sidtl	1102	vpunpcklwd
183	decw	413	gsrex.WRX	643	packssdw	873	sldt	1103	vrcpps
184	div	414	gsrex.WRXB	644	packsswb	874	smsw	1104	vrcpss
185	divb	415	gsrex.WXB	645	packuswb	875	sqrtpd	1105	vrsqrtps
186	divl	416	gsrex.X	646	paddb	876	sqrtps	1106	vrsqrtss
187	divpd	417	gsrex.XB	647	paddd	877	ss	1107	vshufpd
188	divps	418	hlt	648	paddq	878	sz	1108	vshufps
189	divq	419	hsubps	649	paddsb	879	ssfsrex.W	1109	vsqrtpd
190	divsd	420	icebp	650	paddsw	880	ssrex.B	1110	vsqrtps
191	divss	421	idiv	651	paddusb	881	ssrex.RXB	1111	vsqrtsd
192	divw	422	idivb	652	paddusw	882	ssrex.WB	1112	vsqrtss
193	d3d8	423	idivl	653	paddw	883	ssrex.WRB	1113	vsubpd
194	ds	424	idivq	654	palignr	884	ssrex.WX	1114	vsubps
195	dsrex	425	idivw	655	pand	885	ssrex.WXB	1115	vsubsd
196	dsrex.R	426	imul	656	pandn	886	ssrex.X	1116	vsubss
197	dsrex.RXB	427	imulb	657	pause	887	stc	1117	vucomisd
198	dsrex.WB	428	imull	658	pavgb	888	std	1118	vucomiss
199	dsrex.WR	429	imulq	659	pavgw	889	sti	1119	vunpckhps
200	dsrex.WRB	430	imulw	660	pcmpeqb	890	stmxcsr	1120	vunpcklpd
201	dsrex.WRX	431	in	661	pcmpeqd	891	stos	1121	vunpcklps
202	dsrex.WRXB	432	inc	662	pcmpeqw	892	str	1122	vxorpd
203	emms	433	incb	663	pcmpgtb	893	sub	1123	vxorps
204	enter	434	incl	664	pcmpgtd	894	subb	1124	vzeroupper
205	enterq	435	incq	665	pcmpgtw	895	subl	1125	wbinvd
206	enterw	436	incw	666	pcmpistri	896	subpd	1126	wrmsr
207	es	437	insb	667	pextrw	897	subps	1127	xadd
208	encls	438	insl	668	pf2id	898	subq	1128	xchg
209	esrex.RB	439	insw	669	pf2iw	899	subsd	1129	xcrypt-cbc
210	esrex.RX	440	int	670	pfcmpeq	900	subss	1130	xcrypt-cfb
211	esrex.W	441	int3	671	pfcmpgt	901	subw	1131	xcrypt-ctr
212	esrex.WB	442	into	672	pfrsqit1	902	syscall	1132	xcrypt-ecb
213	esrex.WRX	443	invd	673	phaddbd	903	sysenter	1133	xcrypt-ofb
214	esrex.X	444	invlpg	674	phaddd	904	sysexit	1134	xgetbv
215	esrex.XB	445	iret	675	phadduwq	905	sysret	1135	xlat
216	extrq	446	iretq	676	phsubbq	906	test	1136	xor
217	f2xm1	447	iretw	677	pi2fd	907	testb	1137	xorb
218	fabs	448	ja	678	pi2fw	908	testl	1138	xorl
219	fadd	449	ja,pn	679	pinsrb	909	testq	1139	xorpd
220	faddl	450	ja,pt	680	pinsrw	910	testw	1140	xorps
221	faddp	451	jae	681	pmaddubsw	911	ucomisd	1141	xorq
222	fadds	452	jae,pn	682	pmaddwd	912	ucomiss	1142	xorw
223	fbld	453	jae,pt	683	pmaxsw	913	ud1	1143	xrstor
224	fbstp	454	jb	684	pmaxub	914	ud2	1144	xsave
225	fchs	455	jb,pn	685	pminsw	915	unpckhpd	1145	xsaveopt
226	fclex	456	jb,pt	686	pminub	916	unpckhps	1146	xsha1
227	fcmovb	457	jbe	687	pmovmskb	917	unpcklpd	1147	xsha256
228	fcmovbe	458	jbe,pn	688	pmulhuw	918	unpcklps		
229	fcmove	459	jbe,pt	689	pmulhw	919	vaddpd		

230	fcmovnb	460	jcxz	690	pmullw	920	vaddps		

Acknowledgments

Mr. Ashu Sharma is thankful to BITS, Pilani, K.K. Birla Goa Campus for the support to carry out his work through Ph.D. scholarship No. Ph603226/Jul. 2012/01. We are also thankful to IUCAA, Pune for providing hospitality and computation facility where part of the work was carried out.

References

[1] A. Sharma, S. K. Sanjay, "Improving the detection accuracy of unknown malware by partitioning the executables in groups", Proceedings of the 9th international confrence on advanced computing and comunication technologies, Nagpur, India, **(2015).**
[2] P. Mell, K. Kent, J. Nusbaum, "Guide to malware incident prevention and handling", US Department of Commerce, Technology Administration, National Institute of Standards and Technology, **(2005)**.
[3] Govindaraju, "Exhaustive statistical analysis for detection of metamorphic malware", Master's thesis, San Jose State University **(2010)**.
[4] A. Sharma, S. K. Sahay, "Evolution and detection of polymorphic and metamorphic malware: A survey", International Journal of Computer Applications., vol. 90, no. 2, **(2014)**, pp 7-11.
[5] V. Weafer, Editor, "Mcafee labs threats report june 2014", McAfee Labs threat report, **(2014)**.
[6] P. Wood, Editor, "Internet security threat report 2014", Symantec Corporation, USA, **(2014)**.
[7] M. HyppÖnen, "Threat report H1 2013", F-Secure labs, Malaysia, **(2013)**.
[8] P. Wood, Editor, "Internet security threat report", Symantec Corporation, USA **(2012)**.
[9] R. Stone, "A call to cyber arms", Science., vol. 339, no. 6123, **(2013),** pp. 1026–1027.
[10] B. Bencsáth, G. Pék, L. Buttyán, M. Félegyházi, "Duqu: A stuxnet-like malware found in the wild", CrySyS Lab, Hungary, **(2011).**
[11] M. K. Daly, "Advanced persistent threat", Proceedings of the 23rd Large Installation System Administration Conference, Usenix, Baltimore, USA, vol. 35, no. 1 **(2009)**, pp. 104-105.
[12] K. Allix, T. F. Bissyandé, Q. Jérome, J. Klein, Y. Le Traon, *et al*., "Large-scale machine learning-based malware detection: confronting the 10-fold cross validation scheme with reality", Proceedings of the 4th ACM conference on Data and application security and privacy, ACM, **(2014)**, pp. 163–166.
[13] E. Gandotra, D. Bansal, S. Sofat, "Malware analysis and classification: A survey", Journal of Information Security., vol. 5, **(2014)**, pp. 56-64.
[14] H. V. Nath, B. M. Mehtre, "Static malware analysis using machine learning methods", Proceedings of the Recent Trends in Computer Networks and Distributed Systems Security, Springer Berlin Heidelberg, **(2014)**, pp. 440–450.
[15] O. Henchiri, N. Japkowicz, "A feature selection and evaluation scheme for computer virus detection", Proceedings of the Sixth International Conference on Data Mining, IEEE, Hong Kong, **(2006)**, pp. 891–895.
[16] M. Siddiqui, M. C. Wang, J. Lee, "Detecting internet worms using data mining techniques", Journal of Systemics, Cybernetics and Informatics., vol. 6, no. 6, **(2009),** pp. 48–53.
[17] S. B. Mehdi, A. K. Tanwani, M. Farooq, "Imad: in-execution malware analysis and detection", Proceedings of the 11th Annual conference on Genetic and evolutionary computation, ACM, Montreal, **(2009)**, pp. 1553–1560.
[18] I. Santos, F. Brezo, X. Ugarte-Pedrero, P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection", Information Sciences., vol 231 **(2013)** pp. 64–82.
[19] K. Gettman, Editor, "The art of computer virus research and defense", Pearson Education, **(2005)**.
[20] M. G. Schultz, E. Eskin, E. Zadok, S. J. Stolfo, "Data mining methods for detection of new malicious executables", Proceedings of the IEEE Symposium on Security and Privacy, IEEE, **(2001)**, pp. 38–49.
[21] Z. Kolter, M. A. Maloof, "Learning to detect malicious executables in the wild", Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, **(2004)**, pp. 470–478.
[22] M. E. Karim, A. Walenstein, A. Lakhotia, L. Parida, "Malware phylogeny generation using permutations of code", Journal in Computer Virology., vol. 1, **(2005),** pp. 13–23.
[23] D. Bilar, "Opcodes as predictor for malware", International Journal of Electronic Security and Digital Forensics., vol. 1, no. 2, **(2007)**, pp. 156–168.
[24] R. Tian, L. M. Batten, S. Versteeg, "Function length as a tool for malware classification", Proceding of the 3rd International Conference on Malicious and Unwanted Software, IEEE, **(2008)**, pp. 69–76.
[25] R. Moskovitch, D. Stopel, C. Feher, N. Nissim, Y. Elovici, "Unknown malcode detection via text categorization and the imbalance problem", Proceedings of the IEEE International Conference on Intelligence and Security Informatics, IEEE, **(2008)**, pp. 156–161.

[26] R. Moskovitch, C. Feher, N. Tzachar, E. Berger, M. Gitelman, S. Dolev, Y. Elovici, "Unknown malcode detection using opcode representation", Proceedings of the Springer on Intelligence and Security Informatics, **(2008)**, pp. 204–215.
[27] S. M. Tabish, M. Z. Shafiq, M. Farooq, "Malware detection using statistical analysis of byte-level file content", Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics, ACM, **(2009)**, pp. 23–31.
[28] B. Mehdi, F. Ahmed, S. A. Khayyam, M. Farooq, "Towards a theory of generalizing system call representation for in-execution malware detection", Proceedings of the IEEE International Conference on Communications (ICC), IEEE, **(2010)**, pp. 1–5.
[29] I. Santos, J. Nieves, P. G. Bringas, "Semi-supervised learning for unknown malware detection", Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence, Springer, **(2011)**, pp. 415–422.
[30] A. Nappa, M. Z. Rafique, J. Caballero, "Driving in the cloud: An analysis of drive-by download operations and abuse reporting", Proceedings of the Detection of Intrusions and Malware, and Vulnerability Assessment, Springer Berlin Heidelberg, **(2013)**, pp. 1–20.
[31] J. Canto, M. Dacier, E. Kirda, C. Leita, "Large scale malware collection: lessons learned", Proceedings of the 27th International Symposium on Reliable Distributed Systems and Experiment Measurements on Resilience of Distributed Computing Systems, **(2008)**.
[32] A. Venkatesan, "Code obfuscation and virus detection", Ph.D. thesis, San Jose State University **(2008)**.
[33] P. Beaucamps, "Advanced metamorphic techniques in computer viruses", Proceedings of the International Conference on Computer, Electrical, and Systems Science, and Engineering-CESSE'07, **(2007)**.
[34] T. H. Austin, E. Filiol, S. Josse, M. Stamp, "Exploring hidden markov models for virus analysis: a semantic approach", Proceedings of the 46th Hawaii International Conference on System Sciences, IEEE, Havaii, **(2013)**, pp. 5039–5048.

Authors

Ashu Sharma is a full time research scholar in Department of Computer Science and Information Systems, BITS-Pilani, K.K. Birla Goa Campus, India and perusing for his Ph.D degree on malware analysis under the supervision of Dr. Sanjay K. Sahay. He has published couple of papers in malware analysis and two papers are accepted in conference proceedings and will appear in Springer Verlag and Elsevier Procedia CS.

Sanjay K. Sahay received his Ph.D in 2003 and currently working as an Assistant Professor in Computer Science and Information System at BITS-Pilani, K.K. Birla Goa Campus, India. His research interest includes malware analysis, data mining, gravitational waves and machine learning. He has published papers on topics like malware analysis, gravitational waves, machine learning and data mining.