A Lower Bound on Effective Performance Testing for Digital Forensic Tools

Lei Pan Lynn M. Batten

School of Engineering and Information Technology,
Deakin University, Melbourne, Australia
E-mail: {ln,lmbatten}@deakin.edu.au

Abstract

The increasing complexity and number of digital forensic tasks required in criminal investigations demand the development of an effective and efficient testing methodology, enabling tools of similar functionalities to be compared based on their performance. Assuming that the tool tester is familiar with the underlying testing platform and has the ability to use the tools correctly, we provide a numerical solution for the lower bound on the number of testing cases needed to determine comparative capabilities of any set of digital forensic tools. We also present a case study on the performance testing of password cracking tools, which allows us to confirm that the lower bound on the number of testing runs needed is closely related to the row size of certain orthogonal arrays. We show how to reduce the number of test runs by using knowledge of the underlying system.

Index Terms: Abstraction Layer Model, Orthogonal Arrays, Partition Testing, SADFE, Software Performance.

1 Introduction

With the increase of the storage volume on a home computer from a couple of Gigabytes to a few hundred Gigabytes in the last several years, and the introduction of anti-forensic tools freely available on the market [1], the workload for the digital forensic investigator has increased drastically in a short time. It is therefore necessary to introduce an effective and efficient testing methodology to help identify those tools which have best performance.

A recent testing case [2] has revealed that a widely used disk imaging tool, referred to as *dd*, copies an incomplete number of disk sectors under the Linux 2.4 kernel. This result indicates that ignoring the underlying computer system in assessing tools may lead to incorrect conclusions. Because software programs are high-level applications which rely on the underlying computer system to produce correct and instantaneous results, we argue that it is important to consider the underlying computer system performance alongside that of the forensic tool.

We will use the common approach of measuring performance based on the execution time [3] of a digital forensic tool installed on a computer system, and henceforth refer to the conjunction of the tool and computer as a "system". Thus, the less time it takes to finish a task, the better the performance of the system.

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

We require fairness and blindness of the testing process; that is, every tool and every system value should be considered equally, and no performance conclusion should be drawn before the end of the entire testing process. The fairness requirement prevents the setting up of a biased testing situation — each parameter is considered in every testing run and their values will appear an equal number of times; the blindness requirement prevents any advantage gained by deliberately aborting the testing cases.

Due to the fact that most design specifications of digital forensic tools are not publicly available, we assume that the tool tester knows how to use the tools and uses them correctly during the testing phase. Secondly, we assume that the tester has a clear understanding of the formats of inputting and outputting data for each forensic tool to be tested. Finally, we assume that the tester is capable of tuning the underlying computer system by replacing hardware components or adjusting system settings.

In this paper, we cite the existing research results that partition testing [4-7] is the best method of performance testing and that, used in conjunction with orthogonal arrays [8, 9], is the best method for satisfying our requirements and assumptions in the digital forensic context. Furthermore, by examining the existence of orthogonal arrays of certain sizes, we are able to derive a general lower bound on the number of testing cases needed. We also present a case study where the testing runs are reduced before they are tested by using the well-known Taguchi method [10-12].

The focus of this paper is to derive the numerical lower bound of testing cases where all the testing parameters are given. We show that this problem is equivalent to deriving the exact size of orthogonal arrays, which is in turn proportional to the experimental cost. We also illustrate how to normalize the parameters in a system to reduce the number of testing runs, a point often missed in the experimental literature.

This paper is organized as follows: in section 2, we give the required background on partition testing and orthogonal arrays and explain why they are suitable for our purpose. In section 3, we numerically derive lower bounds for the partition testing cases using orthogonal arrays and discuss the possibilities of reducing the testing complexity. In section 4, we present a performance testing example to illustrate our methodology. We conclude in section 5.

2 Partition Testing And Orthogonal Arrays

2.1 Partition Testing vs. Random Testing

Partition testing, in which a tester divides a system's input domain according to some rule and then tests within the subdomains, is a common software testing approach [4-7]. On the other hand, random testing, in which a tester randomly tests the system's input domain, is also widely used in software testing [14]. So which approach best meets the fairness and blindness requirements?

Duran and Ntafos [13] showed that random testing can be as effective as partition testing, under very restrictive conditions; Hamlet and Taylor later in [16] found Duran and Ntafos's conditions are rarely met in real testing cases; Gutjahr [7] followed Hamlet and Taylor's work and theoretically proved that "partition testing compares more favorably to random testing" when no assumptions are made before the test. Furthermore, every case in [5] and [6] showed that random testing is less effective than the investigated partition testing

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

methods.

Under the constraint of the blindness requirement, the tester should complete the entire testing process before comparing the testing results and drawing conclusions. However, in order to fulfill the fairness requirement, a tester using random testing must test all possible combinations, which has the complexity of $\mathcal{O}(l^k)$ for testing a system with k parameters each of which has l values. On the other hand, Cohen et al [15] proved that the complexity of completely testing the same system by using partition testing can be reduced to $\mathcal{O}(l \times \log_2 k)$ provided that a greedy algorithm always exists.

The performance of a computer system may be influenced by a number of "parameters", such as CPU working frequency, choice of memory volume, chip working voltage, settings of OS and settings of the software tools, etc. Normally, each of these parameters can be assigned several "values" by the tester; for instance, the CPU working voltage often can be increased by $0.1v$ or $0.2v$ and the memory volume of a system can be enlarged by adding a new RAM chip. We shall use the term "parameter list" for the set of all pairs (p, v) where p is a parameter of the system to be considered in testing and v is a value associated with p. For the purposes of applying partition testing, the system's input domain will be this parameter list and each parameter along with its associated values will be considered as a subdomain.

According to Hamlet and Taylor, partition testing must sample the input domain often so that a uniform distribution across each subdomain is obtained, because "were the appropriate distribution skewed in any way, it would be a bias" [4]. Essentially, this remark is a restatement of our fairness requirement in which each value of every parameter of the system should be tested for an equal number of times.

Later in this section, we will introduce a method of obtaining a set of testing suites meeting the fairness and blindness requirements.

2.2 Orthogonal Arrays (OAs)

The partition testing strategy can be related to the concept of matrices. Let each parameter correspond to a column in a matrix and fill the column entries with the parameter's associated values. The question is, how should these values be placed in the columns? Our aim is to establish a placement in such a way that the resultant rows contain entries which meet the fairness and blindness requirements and thus establish a suite of tests (one for each row) which completes our test target. Thus, an effective partition testing suite is constructed when the entries of the corresponding matrix have an even distribution, according to Hamlet and Taylor's work mentioned in the previous subsection.

The focus now becomes a search for appropriate matrix constructions, resolved by the use of Orthogonal Arrays. Orthogonal arrays (OAs) are matrices with the following two properties [17] —

1. Those elements appearing in a column occur the same number of times.
2. If all possible ordered pairs appear across two or more arbitrary columns, then the ordered pairs of elements in these columns occur the same number of times.

The two properties of an orthogonal array ensure that the fairness condition holds — the first property suggests an equal chance of appearance of every parameter value; the second one suggests that all possible values of one parameter appear against their counterparts of any other parameters. And the blindness requirement is satisfied if all the rows of an OA

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

are tested as testing suites before drawing conclusions. Thus, given a set of parameters and corresponding values for a system, an appropriate testing suite can be found if an orthogonal array exists with the same number of columns as parameters, and matrix entries correspond to parameter values.

Two example sets of ordered pairs which are OAs are listed in Figure 1 — the left is based on values from the set $\{0, 1\}$ and the right on values from the set $\{0, 1, 2\}$. In both cases, elements appearing in each column are evenly distributed; note that the value 2 does not appear at all in column 1 of example 2. (This will be useful for our application as it indicates that the corresponding parameter only has two values.)

0	0
0	1
1	0
1	1

0	0
0	1
0	2
1	0
1	1
1	2

Figure 1. Examples of Ordered Pairs accross 2 Columns

An orthogonal array allows great flexibility. For instance, any column can be exchanged with another, or even omitted, without violating either of the two properties. Other systems such as Latin Squares and Mutually Orthogonal Latin Squares (MOLS) [9] also satisfy the fairness requirement, but they are more rigid to use since the fairness requirement will not hold if one column is removed.

An additional reason for preferring OAs is a simulation study carried out by Maity and Nayak [18] demonstrating that the number of test cases using an OA "is never higher and in some cases lower than using two popular computer-based design generator AETG or IPO", where information about AETG and IPO can be found in [15] and [19] respectively.

2.3 Our Testing Methodology

As mentioned in the previous section, we will use OAs in a partition testing approach for performance testing over a computer system where digital forensic tools are installed. Our objective is to distinguish which tool performs better by measuring and comparing the execution time over a given workload. Because many aspects of an underlying system affect the execution time of a tool, our methodology needs to be able to process tools and system settings at the same time.

Obviously, it is of little interest to compare the performance of two different tools built for completely different tasks. To group tools by using precise software engineering concepts such as "feature" [20] is not viable, unless we possess the tools design specifications. We therefore focus on the abstraction level inspired by Carrier's model of abstraction layers.

Carrier [21] proposes the concept of abstraction layers as a means of categorizing digital forensic tools. He states, "The custom format is a layer of abstraction" and "each abstraction layer can be described as a function of inputs and outputs." The inputs are further split into the input data and a rule set, and the outputs consist of the output data and

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

a margin of error (as shown in Figure 2). Therefore, an open-source forensic tool can be modeled by using a stack of such single layers, which as a whole takes the input data and produces the output data.

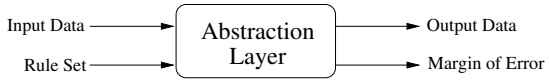

Figure 2. A Single Abstraction Layer [21]

However, one of the key elements in this model, the rule set, (referred to as "design specifications" in Carrier's paper) is assumed to be empty in our case, since, as stated earlier, most design specifications of digital forensic tools are not publicly available. We do assume, however, that the tool tester knows how to use the tools and uses them correctly during the testing phase; that the tester has a clear understanding of the formats of inputting and outputting data for each forensic tool to be tested; and that the tester is capable of tuning the underlying computer system by replacing hardware components or adjusting system settings.

We also omit the concept of margin of error because Carrier's objective was to measure error resulting from the use of the tool, while this is not associated in any way with our objective.

We therefore base our choice of tools to be tested, based on Carrier's results, on the formats of the input and output data. Consequently, we choose a selection of tools which perform similar forensic tasks, have identical input data and have identical output data formats. For example, *John The Ripper* and *OphCrack* are used to recover Windows logon passwords. Both tools have as input a Windows password file and as output as ASCII string.

Having constructed a parameter list by including all possible values of all identified parameters, we use the same strategy as described in the Taguchi method [10-12] to relate the list to an OA — each parameter is represented by a distinct column of the selected OA; each value of a parameter is randomly linked to a symbol in the corresponding column which represents that parameter. Then, each testing suite is represented by a row of the OA. Therefore, the number of rows of the OA becomes the number of testing runs for the given parameter list.

After the testing phase is completed, we again use the Taguchi method to process the resulting data and conduct the proper statistical analysis. The Taguchi method is not the focus of this paper and so a description of it is omitted. The interested reader can find good examples and full references in [10], an informal approach in [11], and complete theoretical background and insightful discussions on technical details from [22] and [12].

The testing cost is now directly proportional to the total number of tests, which is the number of rows of the chosen OA. Hence, the problem of deriving the lower bound of a testing case is the problem of deriving the number of rows in the OA which represents the parameter list. In the next section, we will derive the numerical solution for this problem.

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

3 The Lower Bound

In this section, we will numerically derive the lower bound of the row size of an OA which is capable of accommodating a given list of parameters. As we have shown in the previous section, this lower bound is actually directly proportional to the minimal cost required to carry out a testing case satisfying our requirements.

3.1 Definitions and Notations

In the following definition, we have replaced the word 'factor' by the word 'parameter' in order to remain consistent with our terminology.

Definition 3.1 *An orthogonal array* $OA(N, s_1^{k_1} s_2^{k_2} \dots s_v^{k_v}, t)$ *is an array of size* $N \times k$*, where* $k = k_1 + k_2 + \cdots + k_v$ *is the total number of parameters, in which the first* k_1 *columns have symbols from* $\{0, 1, \dots, s_1 - 1\}$*, the next* k_2 *columns have symbols from* $\{0, 1, \dots, s_2 - 1\}$*, and so on, with the property that in any* $N \times t$ *subarray every possible* t*-tuple occurs an equal number of times as a row. [8] (pp 200)*

We list all the notations used in this paper in Table 1. We set $t = 2$ by following the observation of Hedayat et al. in [8] (pp 199), which says the best value of t is 2 when not all s_i are equal.

Symbol	Meaning
N	The number of tows of an OA
N_{min}	The least possible number of rows of an OA, known as the Rao bound.
N_{max}	The upper bound of N, which guarantees an OA exists of that row size.
N_i	A candidate choice for N, which satisfies $N_{min} \leq N \leq N_{max}$.
s_i	The number of settings of the i-th parameter, also referred to as "levels" in other literature.
$\max(s_i)$	The maximum of s_i.
s	The maximum of s_i in an existing OA, which can be greater than $\max(s_i)$.
k_i	The number of parameters having s_i settings.
k	The sum over k_i.
t	Strength value indicating the evenly distributed pairs across t columns in an OA. $t = 2$ is used when $s_i \neq s_j$.
$\{s_1^{k_1} s_2^{k_2} \dots s_v^{k_v}\}$	The parameter list accommodating all k parameters with their associated settings. We remove the curly brackets when it is in an OA expression.

Table 1. Notations

3.2 Existing Research Results

According to the definition of OA, one could identify an OA by its row size N, its parameter list $\{s_1^{k_1} s_2^{k_2} \dots s_v^{k_v}\}$ and the t value. Given a parameter list and a fixed t value, we only need to derive an exact minimum value for N to identify an OA.

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

Rao in [17] found an inequality relation in which the value of N has to be equal or larger than an integer N_{min}:

$$N \geq N_{min} = 1 + k_1(s_1 - 1) + k_2(s_2 - 1) + \cdots + k_v(s_v - 1)$$

for a given parameter set $\{s_1^{k_1} s_2^{k_2} \ldots s_v^{k_v}\}$. The Rao inequality is a necessary condition on N. For example, the Rao inequality for $\{2^4\}$ is 4, but an $OA(4, 2^4, 2)$ does not exist. Dios and Chopra [23] showed that for some values of N, corresponding OAs do not exist.

Many OAs with small row sizes have been constructed, although the general existence problem is open. Colbourn and Dinitz [9] (Table 4.19 and Table 4.20, pp 174-176) provide a list of OAs for various values of N covering up to 50 parameters; Hedayat et al. [8] (section 12) list OAs with up to several thousand rows; Sloane hosts an online library (`http://www.research.att.com/~njas/oadir/index.html`) containing all known OAs with up to 100 rows.

Remarkably, the following three theorems [8] assure the existence of OAs which can be constructed algebraically.

Theorem 3.1 *For s a prime power, there exists an $OA(s^2, s^{s+1}, 2)$. [8] (pp 27)*

Theorem 1 guarantees that there is an OA which can accommodate a given parameter set of any size, because any parameter list $\{s_1^{k_1} s_2^{k_2} \ldots s_v^{k_v}\}$ can be contained by a larger list $\{s^{s+1}\}$, as long as we find a prime power s such that $s \geq \max(k-1, \max(s_i))$. However, the parameter list $\{s_1^{k_1} s_2^{k_2} \ldots s_v^{k_v}\}$ may be covered by an OA whose row size N is between N_{min} and s^2. Theorems 2 and 3 below give us a finer grain.

Theorem 3.2 *If s is an odd prime power, then for any $n \geq 2$ there exists an $OA(2s^n, s^{\frac{2(s^n-1)}{s-1}-1}, 2)$. [8] (pp 45 Theorem 3.16)*

Theorem 3.3 *If s is a prime power, then for any $n \geq 2$ there exists an $OA(s^n, s^{\frac{s^n-1}{s-1}}, 2)$. [8] (pp 49 Theorem 3.20)*

3.3 Deriving the Lower Bound Numerically

Because there is no known algorithm to generate all existing OAs and the complexity of generation is still unknown [8], we include a library of OAs as a part of the input of our algorithm. In practice, the best computer OA generator was developed by Xu [24], but it does not generate all existing OAs. In this paper, we use Sloane's online library to assist with the solution because it is sufficient for daily performance testing work — an OA with 100 rows can accommodate up to 99 parameters with binary values.

Our algorithm follows:

Algorithm input: a parameter list $\{s_1^{k_1} s_2^{k_2} \ldots s_v^{k_v}\}$ and a library of OAs.
Algorithm output: an integer N which guarantees $OA(N, s_1^{k_1} s_2^{k_2} \ldots s_v^{k_v}, 2)$ exists.
Begin Algorithm

1) If $\max(s_i)$ is a prime power and $\max(s_i)$ divides any $s_j \neq s_i$, then let $s = \max(s_i)$ and go to the next step; otherwise, round $\max(s_i)$ up to the next smallest prime power greater than $\max(s_i)$ and which divides some s_i. We denote this prime power as s.
2) Calculate the Rao inequality bound N_{min}.

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

3) If $k < s+1$, set $N_{min} = s^2$; if s is an odd prime power, calculate the smallest integer $n \geq \log_s(k(s-1+1) \geq 2$, and set $N_{max} = s^n$; otherwise, calculate the smallest integer $n \geq \log_s\left(\frac{k(s-1)}{2}+1\right) \geq 2$, and set $N_{max} = 2s^n$.

4) Iterate an integer N_i from N_{min} to N_{max}, record those dividing s.

5) Go through the OA library to check if there is an OA of N_i-rows that accommodates the given parameter list. If so, choose N as the least N_i from the list obtained in step (4); otherwise, set $N = N_{max}$.

End Algorithm

We are now able to prove the following theorem.

Theorem 3.4 *The above algorithm guarantees the existence of an OA with the smallest number of rows covering a given parameter list* $\{s_1^{k_1} s_2^{k_2} \dots s_v^{k_v}\}$.

Proof: From Theorems 2 and 3, an $OA(N_{max}, s^k, 2)$ exists. The parameter list $\{s^k\}$ contains the given parameter list $\{s_1^{k_1} s_2^{k_2} \dots s_v^{k_v}\}$ because $s \geq s_i$ and $k = k_1 + k_2 + \cdots + k_v$. In our algorithm, the default value of N_i is N_{max}. Because the Rao inequality is a necessary condition, the actual lower bound N must exist in the exhaustive list of consecutive integers between N_{min} and N_{max}. According to the definition of OA, N divides s because each symbol needs to appear an equal number of times in every column. Therefore, iterating the list of candidate integers between N_{min} and N_{max} at s'th step produces the lower bound of N. □

As an example, we now derive the lower bounds for two parameter lists $\{7^1 4^1 2^5\}$ and $\{8^1 2^6\}$ by following our algorithm. For the list $\{7^1 4^1 2^5\}$, $\max(s_i) = 7$ does not divide $s_i = 2$ nor $s_i = 4$. So we round $\max(s_i)$ up to 8, which is a prime power and divides $s_i = 2$ and $s_i = 4$. So take $s = 8$, $N_{min} = 14$ and $N_{max} = 64$. Iterating between 14 and 64, we find $OA(32, 8^1 4^8)$, so that $N = 32$. This suggests we need 32 testing cases to complete testing for $\{7^1 4^1 2^5\}$ satisfying the fairness and blindness requirements. For the list $\{8^1 2^6\}$, $\max(s_i) = 8$ divides $s_i = 2$. So take $s = \max(s_i) = 8$, $N_{min} = 14$ and $N_{max} = 64$. Iterating between 14 and 64, we find $OA(16, 8^1 2^8)$, so that $N = 16$. This suggests we need 16 testing cases to complete testing for $\{8^1 2^6\}$ satisfying the fairness and blindness requirements.

A comparison result is listed in Table 2 where several other algorithms have been used to generate testing suites regarding these two example parameter set, such as IPO [19], recursive block [25], pair-wise [25] and full combinatorial [9]. The results reveal that the use of orthogonal arrays is the most efficient approach to generate balanced testing suites.

Parameter list	OA	IPO [19]	Recursive Block [25]	Pair-wise [25]	Full Combinatorial [9]
$\{7^1 4^1 2^5\}$	Balanced	Unbalanced	Unbalanced	Unbalanced	Balanced
	32-row design	28-row design	40-row design	28-row design	898-row design
$\{8^1 2^6\}$	Balanced	Balanced	Unbalanced	Balanced	Balanced
	16-row design	16-row design	18-row design	16-row design	512-row design

Table 2. Testing Suites Generated by Using Different Schemes

3.4 A Strategy to Further Reduce the Testing Cost

The lower bound of N depends on two bounds N_{min} and N_{max} because $N_{min} \leq N \leq N_{max}$. The value of N_{min} decreases whenever any parameter or any of their values is

dropped; the value of N_{max} decreases when s divides more other parameters' values s_i until s becomes the least common multiplier of all s_i.

The tester can remove parameters from the parameter list by asserting that those parameters and their interaction will definitely have no impact on the performance. Dropping out many parameters in the performance testing for digital forensic tools is not acceptable, because one often does not know whether there is interaction between the eliminated parameter and other parameters. For instance, one cannot claim there is no interplay between a parameter and a tool when the tool's design specification is unknown.

Hence, we advocate possibly reducing the number of values from a parameter whose performance impact is evident. A tester should be able to remove those values of a parameter, if they affect the performance in a consistent manner. For example, in general, the performance of a modern computer system can be boosted by increasing the CPU working frequency, reducing the memory latency, maintaining good voltage on CPU, memory and motherboard chips. On the OS level, any inappropriate configurations of system services, device drivers, storage systems and runtime libraries may drug down the performance significantly.

4 Case Study — Performance Testing on Password Cracking Tools

To validate our methodology, we conducted a performance test on password cracking tools. We searched tools whose input data format is small file containing encrypted information and whose output data format is an ASCII string as the encryption key. Seven well-known and easily accessible tools were therefore selected — *LC5*, *John the Ripper*, *OphCrack*, *Advanced Archive Password Recovery*, *Advanced Office Password Recovery*, *Zip Key* and *Office Key*. These tools can be used to recover the encryption keys (passwords) from system logon password files, encrypted archive files and encrypted Office document files.

We installed those tools on a PC with a CPU of AMD Athlon XP 1800+, Hyundai DDR400 memory chip, a Gigabyte GA-7N400 mother board, dual-boot system of Windows XP SP2 and Debian 3.0. Some parameters were tuned to study their impact on the system performance — the CPU working frequency could be adjusted by tuning the frequency of Front-Side-Bus (FSB) to 100MHz, 133MHz, 166MHz or 200MHz; the RAM volume could be 512MB or 1GB by plugging two different RAM chips; the latency of the memory chips could be set to 2 clock cycles or 3 clock cycles; the working voltage of the CPU could be set to $1.7v$ or $1.8v$; the execution priority of all the tools could be adjusted to the highest or the lowest level in the OS; all tools could be configured to include or exclude special symbols other than English alphanumerical ones.

Hence, the parameter list of our system was $\{7^1 4^1 2^5\}$ — the selection of cracking tool had 7 values, FSB had 4 values, and the rest had 2 values each. The lower bound for the parameter list $\{7^1 4^1 2^5\}$ was calculated as 32 in Section 3.

It was still possible for us to further reduce the testing runs. According to our knowledge of the computer architecture [26], CPU working frequency consistently influences the system's performance: the higher the frequency the faster the system. And we asserted that the interaction between CPU working frequency and other parameters in the list was negligible. And the CPU working frequency was proportional to the FSB frequency in our system. Therefore, we reduced the number of values by removing intermediate values

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

(133MHz and 166MHz) of the FSB frequency parameter and the FSB frequency became a binary parameter with values 100MHz and 200MHz while the updated parameter list became $\{7^1 2^6\}$.

By following our algorithm again, we derived that the updated parameter list $\{7^1 2^6\}$ actually had the same lower bound as the list $\{8^1 2^6\}$, which was 16 as calculated in Section 3. Hence, an orthogonal array $OA(16, 8^1 2^6, 2)$ was chosen. And, we had an option to add an additional tool or a new cracking scenario to the chosen OA. We chose to use a new scenario because *John the Ripper* could crack multiple types of system password files. The testing runs were arranged as following — in testing runs 1 and 2, we used *LC5* to crack Windows logon password files; in testing runs 3 and 4, we used *John the Ripper* to crack Linux system password files; in testing runs 5 and 6, we used *OphCrack* to crack Windows logon password files; in testing runs 7 and 8, we used *ArchPR* to crack encrypted ZIP archives; in testing runs 9 and 10, we used *John the Ripper* to crack SUN Solaris system password file; in testing runs 11 and 12, we used *Zip Key* to crack encrypted ZIP archives; in testing runs 13 and 14, we used *Office Key* to crack encrypted Word documents; in testing runs 15 and 16, we used *AOPR* to crack encrypted Word documents.

Each file had three copies and was encrypted by three different alphanumerical keys "Russia", "ifyinn" and "Lyin05". The average execution time in each testing run is listed in Figure 3, where the execution time varied drastically from a dozen of seconds to a few days. (Please refer to Table 3 in Appendix for the complete list of observed execution time and the setup details.)

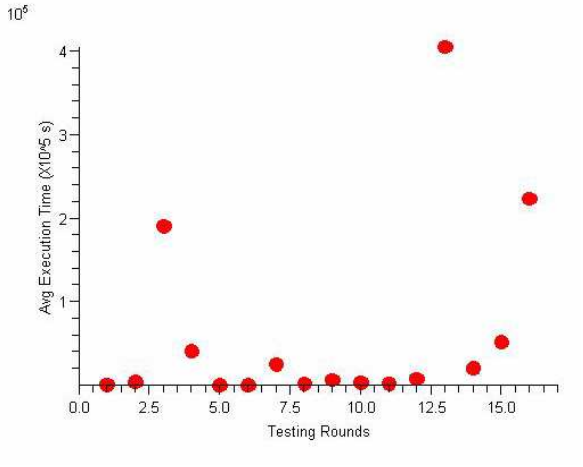

Figure 3. Average Execution Time(in 10^5 seconds) Observed in The 16 Testing Runs

Having processed the experimental data and conducted a statistical analysis in [27] in accordance with the Taguchi method, we identified the tool choice and the value of including/excluding special symbols were the two main causes of the huge difference in the observed execution time; the statistical analysis also indicated that other parameters only impacted the performance difference as negligible random noises. The detailed statistical analysis has been published in [27].

We concluded that the real lower bound in this case study was 16 runs, the same as our prediction. Given the testing result — the tool selection and the symbol selection affected the system performance, a tester must perform a complete random testing with at least

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

$8 \times 2 = 16$ runs to check our result without bias.

5 Conclusion

In this study, we develop a numerical algorithm to derive the lower bound of carrying out effective performance testing cases for digital forensic tools. Our assumptions are that the tester uses tools correctly during the testing phase and knows the data formats of inputting and outputting data. No design knowledge about tools is required.

We also show that the knowledge of the underlying system can be used to further reduce the testing runs which are proportional to the total testing cost. Our methodology is supported by the results from a case study of testing password cracking tools.

In addition, we describe the potential of extending this work to conduct correctness testing cases on digital forensic tools. But there are two obstacles to conquer:

1. The unavailability of design specifications of many forensic tools prevents the tester from determining what is to be tested. As argued by Carrier [28], the release of design specifications will help the tester understand better how the tool should work and thereafter develop more effective and more specific testing suites. A tool tester should be able to identify the testing cost by following our algorithm if a design specification helped him/her realize which parts of the tool are to be tested.
2. Without an effective metric to measure the correctness of tools, comparative testing is not a viable approach. The large amount of information generated by correctness testing has to be quantified before the comparison becomes possible.

Ideally, the design specification of a forensic tool should describe exactly how the tool should behave and how it should not. The unavailability of the vendor's specification can be partially overcame by using some generic tool specifications, such as for disk imaging tools [29] and for write blocking tools [30]. However, such generic specifications may be incomplete or excessively redundant — an incomplete specification will make the tester neglect to test some important tool functionalities and a redundant specification will unnecessarily incur too many testing cases. The generic specifications can be published, discussed, revised and later standardized through peer-reviewing. If vendors decide to use such standardized specifications for developing forensic tools, then the tool tester can develop a uniform set of testing data and a methodology for testing those tools. Consequently, a reliable and trustworthy conclusion can be drawn by referring to independent testing reports from different parties.

In order to develop an effective and efficient testing scheme, an accurate and precise measurement metric is also necessary. The accuracy and precision of the metric affects the quality of testing results. In performance testing, an accurate and precise clock should be used to measure the execution time — we used a tested stopwatch with the precision of 0.5 seconds for testing password cracking tools in the case study. In correctness testing, it is very difficult to find a comparable measurement tool.

Appendix

The testing procedure in this section has been published in [27]. Our experiment consisted of 16 testing rounds, each of which complied with the value in the corresponding row

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

in Table 3 below. The values for the software tools are mentioned in Section 4.

Runs	Software	FSB	RAM Size	Latency	Voltage	Priority	Speical symbol	Unused columns		Execution time (sec) T_1	T_2	T_3
1	0	0	0	0	0	0	0	0	0	4	261	576
2	0	1	1	1	1	1	1	1	1	1	3934	8974
3	1	0	0	0	0	1	1	1	1	10	383850	187637
4	1	1	1	1	1	0	0	0	0	4	27793	94418
5	2	0	1	0	1	0	0	1	1	182	190	3
6	2	1	0	1	0	1	1	0	0	92	97	4
7	3	0	1	0	1	1	1	0	0	20645	40477	13608
8	3	1	0	1	0	0	0	1	1	1353	2626	894
9	4	0	1	1	0	0	1	0	1	1	12514	6474
10	4	1	0	0	1	1	0	1	0	1	6268	3039
11	5	0	1	1	0	1	0	1	0	43	61	5467
12	5	1	0	0	1	0	1	0	1	30	60	22634
13	6	0	0	1	1	0	1	1	0	1	23	1216932
14	6	1	1	0	0	1	0	0	1	6	19	61565
15	7	0	0	1	1	1	0	0	1	11	115152	39516
16	7	1	1	0	0	0	1	1	0	6	513828	157788

Table 3. Testing Rounds and Results of the Case Study

The remaining columns were arranged as follows — FSB frequency was set to 100MHz or 200MHz denoted as treatment "0" and "1" respectively; RAM size was either 512MB or 1GB denoted as treatment "0" or "1"; memory latency was set to 2 clock cycles or 3 clock cycles denoted as treatment "0" or "1"; voltage was set to $1.7v$ or $1.8v$ denoted as treatment "0" or "1"; priority was either the lowest or the highest allocated by the operating system denoted as treatment "0" or "1"; character set included either special symbols or not denoted as treatment "0" or "1". Two columns in the original OA were not used because all the parameters of interest have been arranged. Three passwords "Russia", "ifyinn", and "Lyin05" were used and their execution time (in seconds) was recorded in column T1, T2, and T3, respectively.

Acknowledgment

The authors are grateful to Chris Charnes, Charles Colbourn and Neil Sloane for inspirational discussions on combinatorial designs. The authors wish to thank the anonymous referees for useful comments.

References

[1] R. Harris. "Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem". *Digital Investigation*. 3S pp S44–S49. 2006.

[2] J. D. Kornblum. (2004). "The Linux Kernel and the forensic acquisition of hard discs with an odd number of sectors". *International J. of Digital Evidence*. Vol 3(2). Online proceeding `http://www.utica.edu/academic/institutes/ecii/ijde/articles.cfm?action=issue&id=10`.

[3] R. Jain. *The art of computer systems performance analysis: techniques for experimental design*. Measurement, Simulation, and Modeling. Wiley. 1991.

[4] D. Hamlet, R. Taylor. "Partition testing does not inspire confidence". *IEEE Transactions on Software Engineering* 1990; 16(12):1402–1411.

[5] S. Reid. "An empirical analysis of equivalence partitioning, boundary value analysis and random testing". *Proceedings of the 4th International Software Metrics Symposium (METRICS'97)*, Albuquerque, NM, 5-7 November 1997. IEEE Computer Society Press: Los Alamitos, CA, 1997; pp. 64–73.

[6] H. Yin, Z. Lebne-Dengel, Y.K. Malaiya. "Automatic test generation using checkpoint encoding and antirandom testing". *Technical Report CS-97-116*, Colorado State University, 1997.

[7] W.J. Gutjahr. "Partition testing vs. random testing: The influence of uncertainty". *IEEE Transactions on Software Engineering* 1999; 25(5):661–674.

[8] A. Hedayat, N.J. Sloane and J. Stufken. *Orthogonal arrays: theory and applications*. Springer. 1999.

[9] C.J. Colbourn and J.H. Dinitz. *CRC Handbook of Combinatorial Designs*. CRC Press, Boca Raton, FL. 1996.

[10] G. Taguchi, S. Chowdhury and Y. Wu. *Taguchi's Quality Engineering Handbook*. Wiley. 2004.

[11] S.H. Park. *Robust design and analysis for quality engineering*. Springer. 1996.

[12] V.N. Nair, B. Abraham, J. MacKay, J.A. Nelder, G. Box, M.S. Phadke, R.N. Kacker, J. Sacks, W.J. Welch, T.J. Lorenzen, A.C. Shoemaker, K.L. Tsui, J.M. Lucas, S. Taguchi, R. H. Myers, G.G. Vining, C.F.J. Wu. "Taguchi's parameter design: a panel discussion". *Technometrics*. 1992, Vol 34(2), pp. 127–161.

[13] J.W. Duran and S.C. Ntafos. "An evaluation of random testing". *IEEE Transactions on Software Engineering* 1984; 10(4):438–444.

[14] W.R. Adrion, M.A. Branstad and J.C. Cherniavsky. "Validation, verification, and testing of computer software". *ACM Comput. Surv.* 14, 2. 1982. pp. 159–192.

[15] D.M. Cohen, S.R. Dalal, M.L. Fredman and G.C. Patton. "The AETG system: An approach to testing based on combinatorial design". *IEEE Transactions on Software Engineering* 1997; 23(7):437–444.

[16] D. Hamlet and R. Taylor. "Partition testing does not inspire confidence". *IEEE Transactions on Software Engineering* 1990; 16(12):1402–1411.

[17] C.R. Rao. "Factorial experiments derivable from combinatorial arrangements of arrays". *J. Royal Statist. Soc. (Suppl.)*, 9, 1947, pp. 128–139.

[18] S. Maity and A. Nayak. "Improved test generation algorithms for pair-wise testing". *Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on* 8-11 Nov. 2005.

[19] K.C. Tai and Y. Lei. "A test generation strategy for pairwise testing". *IEEE Transactions on Software Engineering* 2002; 28(1):109–111.

[20] D. Kafura. "A survey of software metrics". In Proceedings of the 1985 *ACM Annual Conference on the Range of Computing:* Mid-80's Perspective: Mid-80's Perspective (Denver, Colorado, United States). ACM '85. ACM Press, New York, NY. 1985. pp.502–506.

[21] B. Carrier. (2002) "Defining digital forensic examination and analysis tools". *2nd Digital Forensic Research Workshop (DFRWS 2002)*. Online proceeding: `http://www.dfrws.org/2002/papers/Papers/Brian_carrier.pdf`.

[22] G. Taguchi. *Introduction to Quality Engineering: Designing Quality into Produces and Processes*. White Plains. 1986. NY: Kraus International Publications.

[23] A. Dios and D.V. Chopra. "On the non-existence of some orthogonal arrays". *J. Combin. Math. Combin. Comput.* 54, 2005 pp. 129–136.

[24] H. Xu. "An algorithm for constructing orthogonal and nearly-orthogoanl arrays with mixed levels and small runs". *Technometrics*, 44. 2002. pp. 356–368.

[25] A. W. Williams. "Determination of Test Configurations for Pair-wise Interaction Coverage". *IFIP TC6/WG6.1 13th International Conference on Testing Communicating Systems (Test-Com 2000)*, 2000. pp. 59–74.

[26] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 3rd edition. 2002.

[27] L. M. Batten and L. Pan. "Development of a theoretical framework for the comparative performance testing of forensic tools". *Technical report TR M06/02*, Deakin University, Melbourne, Australia. 2006.

[28] B. Carrier. "Open source digital forensic tools: the legal argument". *@stake Research Report*. October 2002. Available online: `http://www.digital-evidence.org/papers/index.html`.

[29] J. Lyle. (2002) "Testing Disk Imaging Tools". *2nd Digital Forensic Research Workshop (DFRWS 2002)*. Online proceeding: `http://www.dfrws.org/2002/papers/Papers/James_Lyle.pdf`.

[30] J. Lyle. (2006) "A Strategy for Testing Hardware Write Block Devices". *6th Digital Forensic Research Workshop (DFRWS 2006)*. Online proceeding: `http://www.dfrws.org/2006/proceedings/1-Lyle.pdf`.

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE

Lei Pan was born in Datong, P.R.China on May 21, 1978. He received a MSc degree in dependable computer systems from Chalmers University of Technology, Gothenburg, Sweden, in 2004. He is currently a PhD student in forensic computing at Deakin University, Melbourne, Australia. His current research interest topics include software testing, combinatorial designs, and cryptography. He has extensive experience with the testing of forensic tools and presented his work at the 5th Digital Forensic Research Workshop (DFRWS05) in New Orleans.

Lynn M. Batten is a member of the IEEE and has numerous publications in many areas of information security. She holds the Deakin Chair in Mathematics and is Director of the Information Security Research group at Deakin University responsible for conducting research and training, as well as providing consultancies, across a spectrum of digital security areas. Prof. Batten is seconded one day per week to SECIA, an organization concerned with research, development and training in the information security field.

Proceedings of the Second International Workshop
on Systematic Approaches to Digital Forensic Engineering (SADFE'07)
0-7695-2808-2/07 $20.00 © 2007 IEEE