Applied Mathematics & Information Sciences
An International Journal

Decision Support System for Zero-day Attack Response

Huy Kang Kim[1], Soo-Kyun Kim[2] and Seok-Hun Kim[3*]
[1]Graduate School of Information Security, Korea University, Republic of Korea (First Author)
Email Address:cenda@korea.ac.kr
[2]Department of Game Engineering, Pai Chai University, Republic of Korea
Email Address:kimsk@pcu.ac.kr
[3]Division of Computer Engineering, Mokwon University, Republic of Korea (Corresponding Author)
Email Address:shkim@mokwon.ac.kr

Received: Received May 02, 2011; Revised July 25, 2011; Accepted September 12, 2011
Published online: 1 January 2012

Abstract: Regardless of the existence of the various information security safeguards, many companies remain vulnerable to the unknown attack, which is known as the zero-day attack. In this study, we develop the decision support system (DSS) using case-based reasoning (CBR) for zero-day attack response. Also, our proposed system divides the unknown attack into atomic attacks for zero-day attack detection. Then, this proposed system analyzes the similarity between the new zero-day attack pattern and the known attack patterns. Finally, it suggests the most similar cases with applying similarity functions and CBR. The effectiveness of our system is further shown in the empirical test.

Keywords: zero-day attack, attack similarity, case-based reasoning, decision support system.

1 Introduction

The unknown attack so called zero-day attack is a buzzword of information security area. It is known that zillions of the new attack patterns are produced every year. The problem associated with the zero-day attack is that attack patterns cannot be easily detected with a signature-based intrusion detection system (IDS), because the new attacks' countermeasures and attack signatures are not readily known. Hence, it is very difficult for companies to respond to every new attack and variations of the existing known attacks.

To overcome the limitation of the current IDS and to protect assets from the zero-day attacks, we develop the decision support system (DSS) using three methodologies - similarity functions, case-based reasoning (CBR) and atomic attack analysis.

To find the most similar patterns from comparing the new attack's signature in packet payloads with that of the known attacks, our DSS adopts text mining and string search algorithms of information retrieval and text mining areas.

In order to develop more customized response strategy for a company, additional information such as attacker's IP address, target system's information and target application are required. For this, CBR is conducted. CBR is the process of solving new problems based on the solutions of the similar past problems. CBR can be easily implemented to the real world since most companies have the known attack signature database and they manage the previous incident case records. CBR is very effective to response the zero-day attack because the series of hackings have commonness. The attack targets are mostly predetermined and have commonness with the previous attacks even though conducted attack techniques are changed. Usually, the primary attack targets have valuable information (worth of hacking) or have severe vulnerabilities (ease of hacking). Thus, target system and exploited

[*] **Corresponding Author: SeokHun Kim, shkim@mokwon.ac.kr**

applications will not be changed unless the configuration of the existing systems is completely changed or the new valuable server systems which include sensitive data are changed. Hence, if the attack history is well recorded in the case-base, we can estimate the most similar prior attack cases. When the symptom for the zero-day attack is detected, it will be agile response if security officers can respond to the attack based on the information of the most similar case. Actually, many security advisories suggest similar workarounds where the attack type is similar with the known prior case.

Hence, if new attacks are detected and the attacks' target application are attacked by the previous similar case, then, known workaround for the prior cases are still valid for reacting to the new attacks. If there is attack signature database and case-base of the previous incident, companies can find the most similar attack pattern from the signature database. Then, security officers can search the case related to the same attack pattern and finally they can decide how-to-respond for the new attack based on the past similar cases.

The concept of atomic vulnerability and atomic attack is that any attack can be divided into single or multiple atomic attacks. As the same, vulnerability can be divided into single or multiple atomic vulnerabilities. These atomic vulnerabilities are exploited by the related atomic attacks, and these atomic attack and its vulnerability relationship can be 1:1 or 1:N. Hence, we can define attack as a set of atomic vulnerabilities, which are applied to compromise a certain system with exploiting single vulnerability or multiple vulnerabilities. If the symptom for the zero-day attack is detected, then we divide the unknown attack into several atomic attacks and search for a similar attack case to increase search ability. Even though the IDS does not know all of the atomic attack patterns, if IDS knows some of the atomic attacks then it can detect part of the zero-day attack.

With these mechanisms, we also propose zero-day attack response procedures in this paper. Finally, we estimate our proposed DSS's performance.

2 Literature Review

2.1 Attack and vulnerability

Attacks and vulnerabilities have been studied by many researchers. Studies on attacks and vulnerabilities can be classified into three approaches. The first approach is the state-transition analysis. The state-transition analysis depicts attacks with a state-transition diagram, which is a set of goals and transitions. Any event that triggers an attack state is considered as an intrusion. State-transition analysis employs the rule-based analysis of the audit trails of multi-user computer systems. In state-transition analysis, an intrusion is identified as a sequence of state changes that lead the computer system from some initial states to a target compromised state. NetSTAT uses state transition diagrams to describe network attacks to support network intrusion analysis [4]. The second approach is based on the probabilistic learning and various machine learning techniques. It can be considered as a variation of the state-transition analysis. Table.1 shows approaches with the probabilistic model and Bayesian network. The third approach is based on simulation methodology. This approach tries to formalize attack events discretely. Several studies apply DEVS (Discrete Event System Specification) formalism.

Algorithm	Application and related paper
modeling attack transition with a probabilistic model	Applied to intrusion detection by examining intrusions manifested as anomalies in UNIX system call traces[5]
employing Hidden Markov Model as the finite state machine	Applied to intrusion detection by examining intrusions manifested as anomalies in UNIX command execution log[6]
constructing a Bayesian network for reasoning in the domain of system call execution	Applied to intrusion detection by examining intrusions manifested as anomalies in UNIX system call traces[7]

Table.1 Approaches with the probabilistic model and Bayesian network

Kim *et al.* [8] introduced the concept of VX (vulnerability expression), CV (compound Vulnerability) and AV (Atomic Vulnerability). Kim *et al.* assumed that an attack is composed of a sequence of actions that cause state transition, where each action incurs vulnerability that can be exploited. They call each action an atomic vulnerability, which is an undividable cause-unit of cause-effect model. An example is shown as follows.

Atomic Vulnerability: AV = {Iav, Qav, δav, Type}
Where,
Iav = {Iav1, Iav2, …, Iavn}, external input of AV
Qav = Q(initial state) ∪ Q(final state), state set of AV
δav : Iav × Q(initial state) → Q(final state), AV's

state transition Type : {Fact, Deterministic, Probabilistic}, type of AV

Even though the above three approaches employ different techniques and tools, there exist commonalities.

An attack or attacks can be decomposed into multiple states, beginning with an initial state (start of attack) and ending with the final state (success of attack – which means the system has been compromised finally). When a state transition occurs, some symptoms appear as the result of the attack. By observing these symptoms, we can estimate the probability that the state can be transited to the next state; it can be estimated with stochastic process or Hidden Markov Model or Bayesian rule.

Still, the three approaches differ in understanding the meaning of state transition. In the third approach, Kim *et al.* made a hypothesis that a state transition is created by an attack. They assumed that an attack can be decomposed into the atomic level that can cause state transitions. Therefore, an attack can be expressed as the compound of the atomic attacks. This hypothesis is based on the concept developed by Bishop [9]. The other two approaches, which are the first and second approach, did not definitely state that only an attack can change the states. They utilize a model that explains the normal trends and abnormal trends of system calls when attacks occur. In these approaches, the state is just one of the waypoints between 'beginning of attack' – where the system is still normal – and 'end of attack' – where the system is compromised. Systems can be compromised by a single attack or multiple attacks. However, the latter approaches do not care about the number of attacks or the relationship between attacks that can lead to state transitions. This is because these approaches are developed from based on observation of the real world's audit trail. However, since not every state transition that invokes system calls can be logged, they can miss some changes in state transition.

In fact, some exploitation can be caused by the combination of multiple attacks, while some are caused by a single attack. The combination of multiple attacks can be both sequential and parallel forms. For example, 'GoodTech ssh remote exploit attack' [10] can be done as the following procedure.

- In initial state, server runs a vulnerable version of sshd.
- A hacker begins to do network port scanning at a remote site.
- A hacker attempts to execute 'telnet target server IP target_port' for grabbing the version of ssh daemon.
- A hacker sends strings so that the target server can experience buffer overflow on the ssh daemon.
- A hacker changes 'eip address' and 'offset' for loading a command.
- Finally, a hacker sends a malicious command to the server after the daemon's memory has been overflowed.

In the above example, among the six procedures, the second procedure can be skipped and the third procedure can be skipped by a hacker. In the third procedure, a hacker just can send strings to make overflow the sshd brute-forcedly without confirmation of versions. So, some hackers skip some of the procedures and directly execute the fourth procedure. However, when hackers fail to overflow the sshd, they need to go back to the second procedure and then execute the third procedure to gain the exact version information of sshd. After that, they execute the fourth procedure once again with the exact information, which was acquired in the third procedure. In any case, the fifth procedure should always be followed by the fourth procedure.

As we can see the above example, attacks can be divided into several sub-attacks. These atomic attacks might have sequential relationships among them, that is, they have dependency between them.

We developed a new approach that could fully explain the 6 procedures. We illustrated a new attack and state transition model as in the Fig.1. Our model encompasses all the different viewpoints of researchers including that of Eskin [5], Warrender *et al.* [6], Kruegel et al. [7], Kim *et al.* [8], and Bishop [9].

The common points are fact that an attack can be decomposed into atomic attacks, and the fact that every atomic attack changes the state. However, the symptom caused by this atomic attack is not always observable. Therefore, there is possibility that the attack will not to be detected until the system is fully compromised. IDS can miss the detection of every symptom caused by an atomic attack when the atomic attack does not show remarkable symptoms. (e.g., definitive signatures on packet payload and anomalous action on a process leave logs in the system's log files.) Moreover, some poorly designed applications or OS cannot properly react to every atomic attack because they might not have adequate functions of audit or defensive mechanisms. In addition, attackers can modify their attacks with many evasion techniques described in the study of Cohen [11]. In these cases, interim state transition cannot be shown.

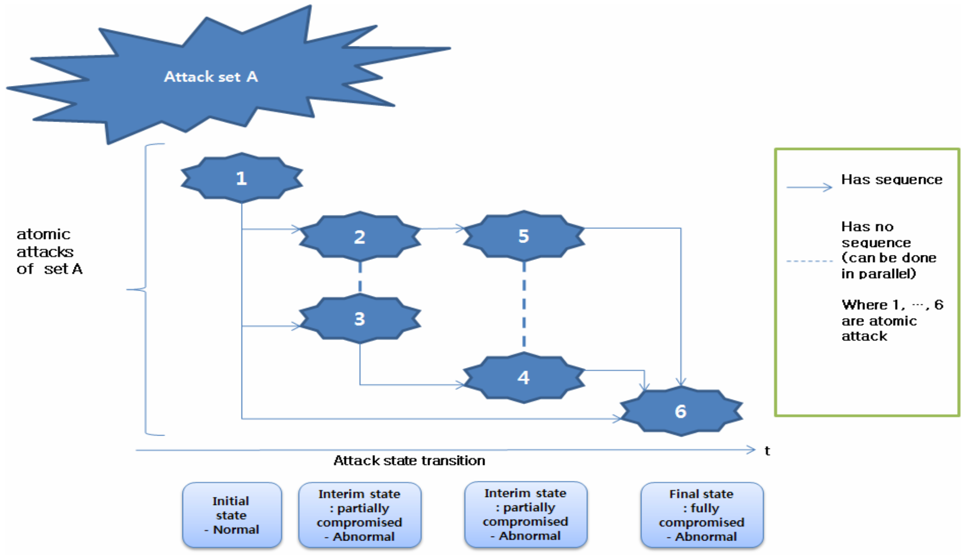

Attack set A
atomic attacks of set A
1
2
5
3
4
6
Has sequence
Has no sequence (can be done in parallel)
Where 1, ···, 6 are atomic attack
Attack state transition
t
Initial state - Normal
Interim state : partially compromised - Abnormal
Interim state : partially compromised - Abnormal
Final state : fully compromised - Abnormal

Fig.1 Atomic attacks and state transition

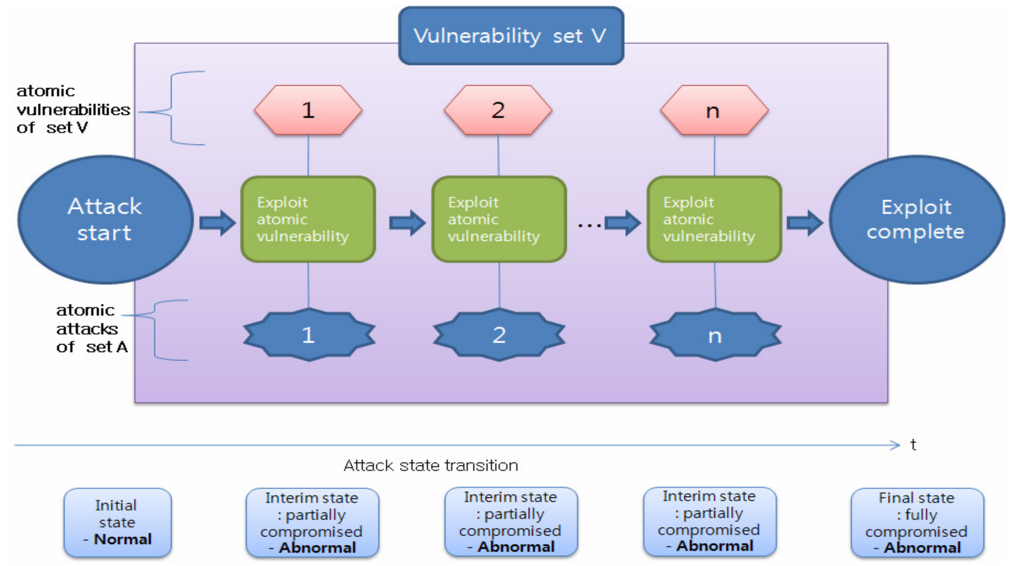

Vulnerability set V
atomic vulnerabilities of set V
1
2
n
Attack start
Exploit atomic vulnerability
Exploit atomic vulnerability
Exploit atomic vulnerability
Exploit complete
atomic attacks of set A
1
2
n
Attack state transition
t
Initial state - Normal
Interim state : partially compromised - Abnormal
Interim state : partially compromised - Abnormal
Interim state : partially compromised - Abnormal
Final state : fully compromised - Abnormal

Fig.2 Atomic vulnerabilities and relevant atomic attacks

As an attack is divided into atomic attacks, vulnerability can be divided into sub-vulnerabilities, that is, atomic vulnerabilities. These atomic vulnerabilities are exploited by the related atomic attacks. These relations are illustrated in the Fig.2.

2.2 CAPEC, CWE, and CVE

There are other approaches that are based on software engineering, CAPEC (Common Attack Pattern Enumeration and Classification) [12], CWE (Common Weakness Enumeration) [13], and CVE (Common Vulnerability Exposure) [14] by MITRE cooperation. They are the de-facto standards of expressing vulnerabilities and attacks with unique identification indexes.

CAPEC is a measurable catalog of attack patterns along with detail schema and classification taxonomy. CAPEC categorizes attacks and break down attacks into an atomic level. Every attack pattern is divided into undividable attack patterns. These undividable attack patterns have unique CAPEC ID. CAPEC shows the full description, related weakness (CWE) and related known vulnerabilities (CVE) of undividable attack.

CWE is a measurable category of software weaknesses that exist in source code, operational systems, and other software. This weakness has a more specific and detail meaning of vulnerability. Vulnerability can be divided into undividable union weakness. This undividable weakness has unique CWE ID. For each undividable weakness, CWE provides full description about the weakness itself, the related attack (CAPEC) and the related known vulnerabilities (CVE). For examples, CWE-306, which describes 'No Authentication for critical function', can be exploited by CAPEC-62, CAPEC-36, CAPEC-12 and CAPEC-40.

CVE is a dictionary of publicly known information security vulnerabilities and exposures. CVE Identifiers are common identifiers publicly known for identifying vulnerabilities. Unique CVE ID is assigned to the each vulnerability. CVE ID enables data exchange between security products and provides a baseline index point for evaluating coverage of tools and services. For the each vulnerability that has CVE ID, CVE shows the attack title, attack description and related reference. Some of the vulnerabilities shown in the CVE list are atomic vulnerabilities by themselves, while other vulnerabilities shown in the CVE list are the combinations of several atomic vulnerabilities.

In conclusion, CAPEC is good for expressing atomic attacks and their relations. CWE is good for expressing atomic vulnerabilities. CVE can be assumed as a set of atomic vulnerabilities.

2.3 Attack graph

Noel *et al.* [16] introduce the concept of the attack graph. The attack graph shows the relationship between sub-attacks, which is in the form of vertex, and node. Fig.3 shows an example of an attack graph. The Graph theory is applied for the expression of sub-attacks and its condition to transit.

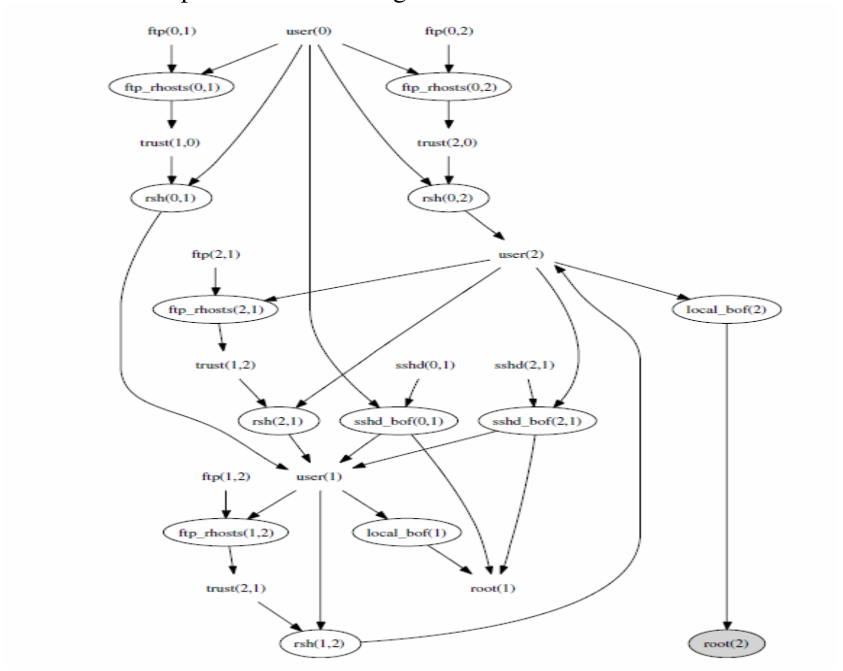


Fig.3 Attack graph example (source: Wang et al. [15])

There are several related studies done by Noel and colleagues [15-17]. These studies commonly called as attack graph. The attack graph can be applied to calculate the optimal hardening cost. Reike [18] utilized the attack graph to visualize the enterprise network vulnerabilities.

The attack graph theory adopts the concept that an attack can be divided into smaller ones and the exploitation process can be expressed in the series of state transition.

2.4 Case-based Reasoning

CBR's overall process is described in the Fig.4. The process of CBR is very simple and comprehensible. CBR also doesn't need prior-training of data for solving a new problem.

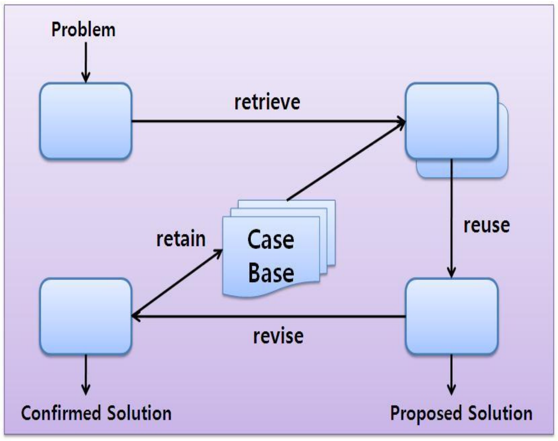


Fig.4 General cycle of a CBR system

Studies that aim at detecting anomalous attacks with CBR are as follows. Musman *et al.* [19] developed SoSMART with applying CBR. It relies on string match of rule-base to detect suspicious symptom of attack on the log files. However, not all attacks make application to generate log events. Hong *et al.* [20] proposed the CBR model for intrusion detection. In their model, CBR and decision tree C4.5 are used for misuse detection, and neural networks are used for anomaly detection. Kim *et al.* [21] apply CBR to find out similarities between already known hacking patterns and new hacking patterns. With the RFM (Recency, Frequency, Monetary) analysis methodology and CBR, they develop DSS which can minimize false alarms and decrease the time to respond to hacking events.

Similar case retrieval in the CBR process plays an important role in reasoning. Hence, how to design the similarity metrics in the case retrieval process is critical for increasing accuracy. Usually the k-nearest neighbor (k-NN) method or its variants are widely used as the retrieval mechanism.

2.5 Similarity and distance metrics

Information retrieval is a research area of searching for documents, for information within documents, and for metadata about documents, as well as that of searching databases. In the middle of information retrieval process, similarity and distance metrics are usually used to enrich search ability. These metrics are also applied to an Internet search engine application for searching similar words, sentences, and documents. Also, this string similarity metric can be applied to search attack variants by analyzing strings in the attack packets.

Many similarity and distance algorithms have been developed to support the string search and calculation of similarity between documents or sentences. The representative string comparator algorithms are as follows.

①Edit-distance (Levenshtein distance)

The edit-distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. A generalization of the Levenshtein distance (Damerau-Levenshtein distance) allows the transposition of two characters as an operation. This algorithm is frequently used in applications that need to determine how similar words are, such as spelling-check programs. The edit-distance can be considered as a generalized version of Hamming distance.

② Jaro distance and Jaro-Winkler distance

Jaro-Winkler distance [23] is a variant of the Jaro distance [22] metric. They are mainly used in the area of record linkage (duplicate detection). The higher the Jaro-Winkler distance for two strings is, the more similar the strings are. The Jaro-Winkler distance metric is designed and best suited for short strings such as person names.

③Smith-Waterman distance

Smith-Waterman distance [24] is a well-known algorithm for performing local sequence alignment, that is, for determining similar regions between two nucleotide or protein sequences. Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure.

A comparison between metrics is well summarized in the research of Cohen et al. [25]. Cohen et al. investigate the overall performance of several string distance metrics, edit-distance-like functions, token-based distance functions, and hybrid distance functions.

3 Methodologies

3.1 Similarity metrics between a known attack and an unknown attack

Considering the atomic attack analysis, the zero-day attack response procedure can be proposed as follows.

① When an anomalous symptom happens in network or server systems, then capture packets at that time.

② Input the captured packets into IDS. IDS can determine whether the captured packet as an attack, which means the known atomic attacks are included in the captured packets.

A. In this case, find the vulnerability that can be exploited by the found atomic attacks.

B. Find the assets that have those vulnerabilities in A.

Then, how can the organization response to the zero-day attack when a known atomic attack is not found in procedure ②? Now we need algorithms for searching similar attack patterns with the patterns in the captured packets.

As most detecting systems do, when deciding the similarity between attacks, the first important factor is considering the strings in the packet payload. In most cases of variant attacks in the same attack category and same attack target application, the attack signature strings have similar patterns.

For example, Klez worm type A and its variant attack, Klez worm type B, are not much different. From this viewpoint, Table.2 and Table.3 show the variant attacks in RPC and NETBIOS SMB protocol have own attack patterns. We can see that similar strings repeatedly occurred in attack signatures in the packets.

Snort ID	category	Attack title	Attack signatures in packet	Related CVE	Disclosure date
6200	RPC	RPC portmap 390113 udp request	|00 01 86 A0|)|00 00 00 03|)|00 05 F3 E1|)|00 00 00 00|	2007-3618,2000-0719	2000-08-10
6201	RPC	RPC portmap 390113 udp procedure 4 attempt	|00 05 F3 E1|)|00 00 00 04|)*sn_sub_rqst*	2007-3618, 2000-0902	2000-09-07
6202	RPC	RPC portmap 390113 udp procedure 5 attempt	|00 05 F3 E1|)|00 00 00 05|)*sn_sub_rqst*	2007-3618, 2000-0988	2000-10-13

Table.2 Attack signature in the snort ruleset: RPC portmapper

Snort ID	category	Attack title	Attack signatures in packet
1429	NETBIOS SMB	NETBIOS SMB ISystemActivator WriteAndX unicode alter context attempt	*|00|)|FF|SMB/)|05|)|0E|)|A0 01 00 00 00 00 00 00 C0* 00 00 00 00 00 00|F
1430	NETBIOS SMB	NETBIOS SMB-DS ISystemActivator WriteAndX alter context attempt	*|00|)|FF|SMB/)|05|)|0E|)|A0 01 00 00 00 00 00 00 C0* 00 00 00 00 00 00|F
1431	NETBIOS SMB	NETBIOS SMB-DS ISystemActivator unicode alter context attempt	*|00|)|FF|SMB*%)&*|00|)|05|)|0E|)|A0 01 00 00 00 00 00 00 C0* 00 00 00 00 00 00|F
1432	NETBIOS SMB	NETBIOS SMB-DS ISystemActivator WriteAndX unicode little endian alter context attempt	*|00|)|FF|SMB/)|05|)|0E|)|A0 01 00 00 00 00 00 00 C0* 00 00 00 00 00 00|F

Table.3 Attack signature in the snort ruleset: NETBIOS SMB ISystemActivator

Attack signatures in packet	Hex stream after data preprocessing
|00|)|FF|SMB/)|05|)|0E|)|A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00|F	00 FF534D422F 05 0E A001000000000000C000000000000046
|00|)|FF|SMB/)|05|)|0E|)|A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00|F	00 FF534D422F 05 0E A001000000000000C000000000000046
|00|)|FF|SMB%)&|00|)|05|)|0E|)|A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00|F	00 FF534D4225 2600 05 0E A001000000000000C000000000000046
|00|)|FF|SMB/)|05|)|0E|)|A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00|F	00 FF534D422F 05 0E A001000000000000C000000000000046

Table.4 Example – original NETBIOS related attack signature and preprocessed signature into HEX codes

For the better search performance, we transform the attack signatures into series of hex strings, and it then becomes easier to find the similarity as shown in Table.4. We can find that there are many common features by protocol and target applications. These attack string patterns have various characteristics – duplicated strings pattern, repetitive strings pattern, and so on. So, we assume that there are optimal similarity measure for each specific protocol and application.

To find this optimal similarity function per protocol, we designed an experiment. In addition, we estimate the detection power of zero-day attacks with those similarity functions. We choose the widely used similarity and distance metrics for calculating similarity between attack strings. The following representative functions are used for this experiment.

① Jaro distance

② Jaro-Winkler distance

③ Edit-distance (also known as Levenshtein distance)

④ modified Edit-distance (we set the transposition cost 1 instead of 2)

⑤ Bag distance (also known as approximate Levenshtein distance) – this distance measure is proposed by Bartolini et al. [27].

⑥ Smith-Waterman distance

Each function has a special characteristic. Jaro and Jaro-Winkler distance are fitted to search duplicated strings, especially in case of short strings. Edit-distance is a generic algorithm, but in case that the number of substitution and insertion is unexpectedly big, this algorithm reports that the two input strings have low similarity rather than Jaro-Winkler-kind distance measures. We normalize the search result as 0 to 1. The meanings of the value can be implied as Table.5.

similarity value S Between attack pattern A and attack pattern B	Meaning
0	“Completely different attack” Attacks A and B have no relationship.
0<S<1	“similar attack” At least the workaround can be similar or the same. So, the countermeasure can be referred the previous cases.
1	Attacks A and B are completely same. So, the countermeasure of attack A can be applied attack B without any modifications.

Table.5 Similarity values and their meaning

Now, the zero-day response procedure can be updated as follows.

① When anomalous symptom occurs in the network or on server systems, capture packets at that time.

② Input the captured packets into IDS. IDS can detect the captured packet as being an attack, which means the known atomic attacks are included in the captured packets.
   A. In case of this, find the vulnerability set (CVE) that can be exploited by the found atomic attacks.
   B. find the assets that have those vulnerabilities in A

③ When a known atomic attack is not found in procedure②, find the similar attacks based on the string similarity search. The attack string in the captured packets and the strings in the attack signature database (that is, misuse signatures in the ruleset) are compared.

④ Find the vulnerability set (CVE) that can be exploited by the most similar attack set found in procedure③.

⑤ Find the assets which have those vulnerabilities set (CVE) in procedure④.

3.2 Case-based reasoning

In section 3.1, we show the way to find out the most similar attack patterns with various string similarity metrics. In case that there is no findings by string similarity search, or in case that the queried result need to be improved, we need to search with more specific information which can affect the search accuracy. So, with adding up more specific information such as source IP address and destination IP address and destination port, more accurate result can be derived. Comparing strings in the packet level is important but it is micro view. In a point of macro view, hackers IP address, attack target’s OS and application information are well-fitted information for the organizations.

CBR is very powerful methodology when decision maker need to solve problem from the information of the domain knowledge. For this, as like the other CBR applications, it is required to define a distance measure for case retrieval. Symbolic data for explaining a case are coded into vector form for convenience of calculating distance, and then the distance between cases are measured. Usually Euclidean distance and cosine distance are used to calculate distance of two vectors.

When we describe a case, 5W1H (when, who, where, what, which, how) information are required intuitively. Considering 5W1H, attack case can be coded into the following vector form.

Case vector =

(attack date, attackers’ IP, attackers src port, target IP, target port, target application, target application’s version, target OS, target OS version, target OS architecture, attack pattern string)

- The vector is designed for expression of these following assumptions.
- The attacks happened in adjacent time line are similar.
- The attacks from similar IP address range are similar.
- The attacks which seek to exploit the same target servers are similar.
- The attacks which seek to exploit the same OS and same application are similar.

Assuming that there are {Solaris, HP-UX, AIX, Windows, Linux, SGI} OS. When coding the OS symbol information into numeric one, it cause an error to measure distance that if the value is set as {Solaris = 1, HP-UX = 2, AIX=3, Windows =4, Linux = 5, SGI = 6}. It is a just a coded value for distinguishing with other symbol values. If we set values from 1 to 5 then that cannot explain the distance between OS. (The value of SGI – the value of Windows is 2, and the value of SGI – the value of Solaris is 5. It does not imply that SGI is more similar with Windows OS rather than Solaris OS.)

For resolving this problem, we limit OS information only for (Solaris, HP-UX, AIX, Windows, Linux, SGI) and set the sub-vector form as follows.

Solaris = (1, 0, 0, 0, 0, 0)
HP-UX = (0, 1, 0, 0, 0, 0)
AIX = (0, 0, 1, 0, 0, 0)
Windows = (0, 0, 0, 1, 0, 0)
Linux = (0, 0, 0, 0, 1, 0)
SGI = (0, 0, 0, 0, 0, 1)

This vector form is quite exact but it cannot be avoided from the curse of dimensionality when considering all OS in the world. Hence, we only consider the top 6 OS at this time. In addition, the OS version information is highly correlated with the OS information and OS architecture. Likewise, the application's version information is highly correlated with the application information also. Finally, the application information can be derived by port number because the port numbers of the well-known Internet services are static.

The vector for expressing attack case is simplified as below. If not we should consider the vector whose dimension is at least 1024. (The number of port for well-known Internet services)

Simplified Case vector = (attack date, attackers' IP, target IP, target port, target OS, OS architecture, attack pattern string)
Where,
target OS = {Solaris, HP-UX, AIX, Windows, Linux, SGI}
target OS architecture = {x86, non x86}

Suppose the three attack case as follows.

Case 1 is the attack that is from 118.175.151.128, happened at March-3-2008, the target host's OS is Solaris 10, architecture is x86, the target program is bind ver. 9.3, and the target host's IP address is 143.248.1.17. At that time, IDS detected the attack string in the attack packet as "ABCD098000000000100000000000000100012020 20200261".

Case 2 is the attack that is from 153.23.4.5, happened at Feb-8-2006, the target host's OS is HP-UX 11i, architecture is PA_RISC(non x86), the target program is bind ver. 8, and the target host's IP address is 143.248.3.9. At that time, IDS detected the attack string in the attack packet as "2E2E2F2E2E2F2E2E2F".

Case 3 is the attack that is from 118.175.151.3, happened at Dec-12-2007, the target host's OS is Solaris 9, architecture is sparc (non x86), the target program is bind ver. 9, and the target host's IP address is 143.248.1.1. At that time, IDS detected the attack string in the attack packet as "800007000000000013F000102". Then these attack cases are coded as follows.

Case 1
=(2008-03-03, 118.175.151.128, 143.248.1.177, 53, {Solaris}, {x86}, ABCD09800000000010000000000000001000120202020200261)
= (2008-03-03, 118.175.151.128, 143.248.1.177, 53, 1, 0, 0, 0, 0, 0, 1, 0, ABCD09800000000010000000000000001000120202020200261)

Case 2
=(2006-02-08, 153.23.4.5, 143.248.3.9, 53, {HP-UX}, {non x86}, 2E2E2F2E2E2F2E2E2F)
=(2006-02-08, 153.23.4.5, 143.248.3.9, 53, 0, 1, 0, 0, 0, 0, 0, 1, 2E2E2F2E2E2F2E2E2F)

Case 3
=(2007-12-12, 118.175.151.3, 143.248.1.1, {Solaris}, {non x86}, 800007000000000013F000102)
=(2007-12-12, 118.175.151.3, 143.248.1.1, 1, 0, 0, 0, 0, 0, 0,1 , 800007000000000013F000102)

The distance between binary values can easily calculated, but the distance of date , distance of IP address, distance of port and distance of strings should be defined here.

The distance of date can be defined as follows.

$$d(date1, date2) = \frac{|date1 - date2|}{365 \times (|topyear - bottomyear| + 1)}$$

e.g.

- date1=2008-03-03
- date2=2006-02-08
- date3=2007-12-12

Then *topyear = 2008 and bottomyear = 2006*

$$\therefore d(date1, date2) = d(2008-03-03, 2006-02-08)$$

$$= \frac{390}{365 \times (2008 - 2006 + 1)} = 0.356$$

$$\therefore d(date1, date3) = d(2008-03-03, 2007-12-12)$$

$$= \frac{83}{365 \times (2008 - 2006 + 1)} = 0.076$$

Distance of IP address is defined as follows.

$d(IPaddress1, IPaddress2)$

is 0, where the IPaddress1 and IPaddres2 are same. (e.g. 143.248.1.1, 143.248.1.1)

is 0.25, where the IPaddress1 and IPaddress2 are matched up to the C Class IP address range (e.g. 143.248.1.1, 143.248.1.8)

is 0.5, where the IPaddress1 and IPaddress2 are matched up to the B Class IP address range (e.g. 143.248.1.1, 143.248.3.8)

is 0.75, where the IP address1 and IPaddrerss2 are matched up to the A Class IP address range (e.g. 143.248.1.1, 143.2.3.4)

is 1, where the IPaddress1 and IPaddress2 do not have any common segment.

Distance of port is defined as follows.

$d(port\#1, port\#2)$

is 0, where the port#1 and port#2 are same. (e.g. 1433, 1433)

is 0.5, where the port#1 and port#2 are highly correlated, the correlated port lists are as follows.

- (80, 8080) : http , http's another port
- (80, 443) : http , https
- (8080,443) : http's another port , https
- (119,563) : nntp, nntps
- (194,994) : irc, ircs
- (1433, 1434) : ms-sql-s, ms-sql-m
- (137, 138): netbios-ns, netbios-dgm
- (137, 139): netbios-ns, netbios-ssn
- (138,139) : netbios-dgm, netbios-ssn
- (161, 162): snmp, snmp-trap
- (143, 220): imap2, imap3
- (21, 22): ssh, telnet
- (67, 68): bootps, bootpc
- (109, 110): pop2, pop3
- (109,995) : pop2, pop3s
- (110,995): pop3, pop3s

is 1, others

The distance between attacks strings are calculated with the similarity measure described in section 3.1. Distance of attack strings is defined as follows.

$d(string1, string2)$ has a value from 0 to 1. The distance is opposite concept of similarity.

$\therefore d(string1, string2) = 1 - similarity(string1, string2)$

Especially,

$d(string1, string2)$

is 1, where the string1 and string2 are different completely.

is 0, where the string1 and string2 are same.

Finally, the proposed distance between cases is summarized as follows.

D(Case1, Case2)

$=sqrt\{\ d(date1, date2)^2 + d(\text{attackerIP1,attackerIP2})^2 + d(\text{targetIP1,targetIP2})^2 + d(\text{port\#1,port\#2})^2 + d(OS1, OS2)^2 + d(architecture1, architecture)^2 + \{1 - similarity(string1, string2)\}^2\ \}$

Where similarity function is Jaro-Winkler, then

Jaro.winkler(ABCD0980000000001000000000000010001202020200261, E2E2F2E2E2F2E2E2F) = 0.436

$(1 - similarity)$ *= 0.564*

D(Case1, Case2)

$= \sqrt{0.127 + 0 + 0.25 + 2 + 2 + 0.318}$ *=2.167*

Likewise above,

D(Case1, Case3) =1.481

D(case1, case) ≥ D(case1, case3)

So it says, "Case1 is more similar with Case3 rather than Case2."

With the proposed Euclidean-like distance, the similarity or the distance between attack cases are derived.

This process is automated in our DSS application shown in section 4. To summarize, based on the methodologies described in this section, we propose DSS which works with following logic shown in the Fig.5.

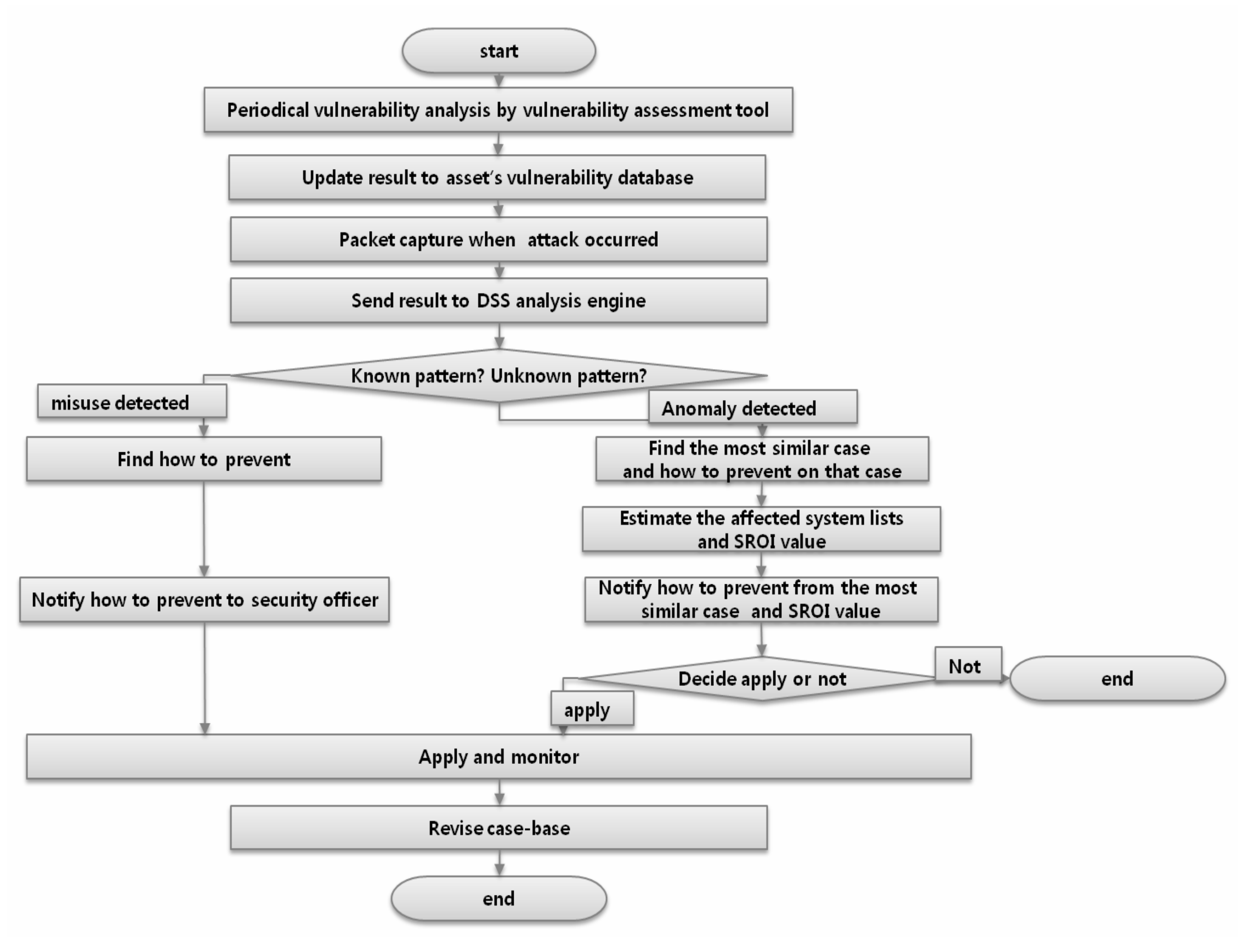

start
Periodical vulnerability analysis by vulnerability assessment tool
Update result to asset's vulnerability database
Packet capture when attack occurred
Send result to DSS analysis engine
Known pattern? Unknown pattern?
misuse detected
Anomaly detected
Find how to prevent
Find the most similar case and how to prevent on that case
Estimate the affected system lists and SROI value
Notify how to prevent to security officer
Notify how to prevent from the most similar case and SROI value
Decide apply or not
Not
end
apply
Apply and monitor
Revise case-base
end

Fig.5 proposed Decision Support System's working procedures

4 Framework and architecture of our DSS

4.1 Proposed zero-day attack response procedures

Our proposed procedures of zero-day attack are composed of four phases. Phase 1 is when zero-day attack is outbreak, and then anomalous symptom is detected. In Phase 2, security administrators capture the inbound and outbound packets on the network. In Phase 3, security administrators can find the most similar attack and case with similarity and distance metrics. In the first part of Phase 3, the strings on the captured packets will be compared with the known attack signature database. After searching the most similar attack signature, security administrators can find the safeguards from the most similar attack. And then, to acquire the prior domain knowledge, security administrator can find the most similar case with similarity and distance metrics. Searching the most similar attack case by CBR is performed. After searching the most similar case, security administrator can find the safeguards from the most similar case on that company. Phase 4 is to make a decision based on results of Phase 3.

These zero-day attack response procedures are summarized in the Fig.6. This DSS application is designed for supporting all methodologies proposed in section 3. Our DSS application mainly works on the phase 3 and phase 4. We assume that the anomalous symptoms can be detected by other monitoring systems which are deployed previously in the organizations. (e.g. MRTG, CPU monitoring system, process monitoring system and log analysis system). For working these procedures well, information asset database is required, which possess the every information asset's vulnerability information with CVE ID.

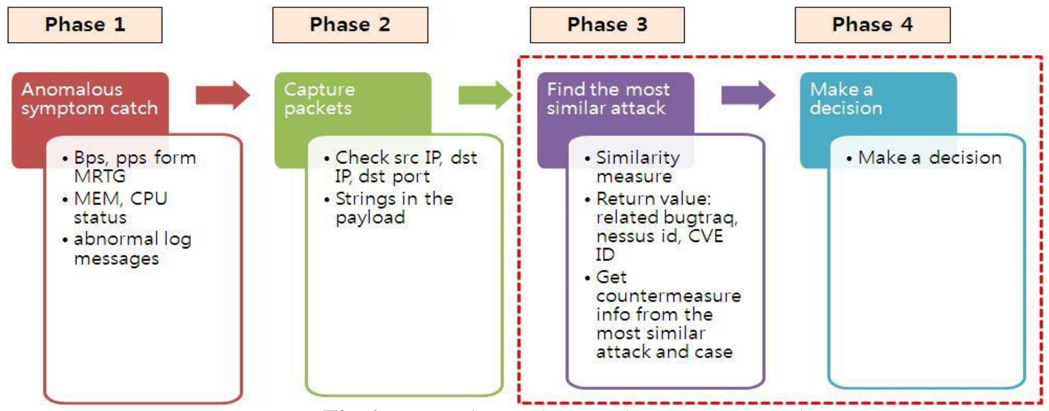


Fig.6 Proposed zero-day attack response procedures

4.2 Implemented DSS applications

This DSS is composed of the following 3 components as shown in the Fig.7. The first are databases. DSS has asset inventory database, atomic vulnerability database, atomic attack database, atomic safeguard database, case-base database and attack signature database for similar attack search. Asset inventory database includes vulnerability information; this vulnerability information is routinely updated by Nessus, the most famous vulnerability scanner program [28]. Second, there is DSS engine to find affected system. Third, in presentation layer, the search result will be displayed to security administrators. The screenshot of main GUI of our DSS is presented in the Fig.8. This program is developed with Delphi language, Python scripting language and MySQL database. It supports easy update method for the attack signature update, CWE/CAPEC/CVE database and information asset database.

Vulnerability scanning results of the information assets are parsed and saved into database as shown in Table.6. The gathered IP address, OS, open port information will be referred for CBR.

DSS parses the gathers vulnerabilities information and makes it referable from the information asset database as Table.7.

It extracts CVE information and then put that information into vulnerability database periodically. Especially, for calculating distance and case relation, the asset database should include IP address, OS version and running applications' information as shown in the Table.7.

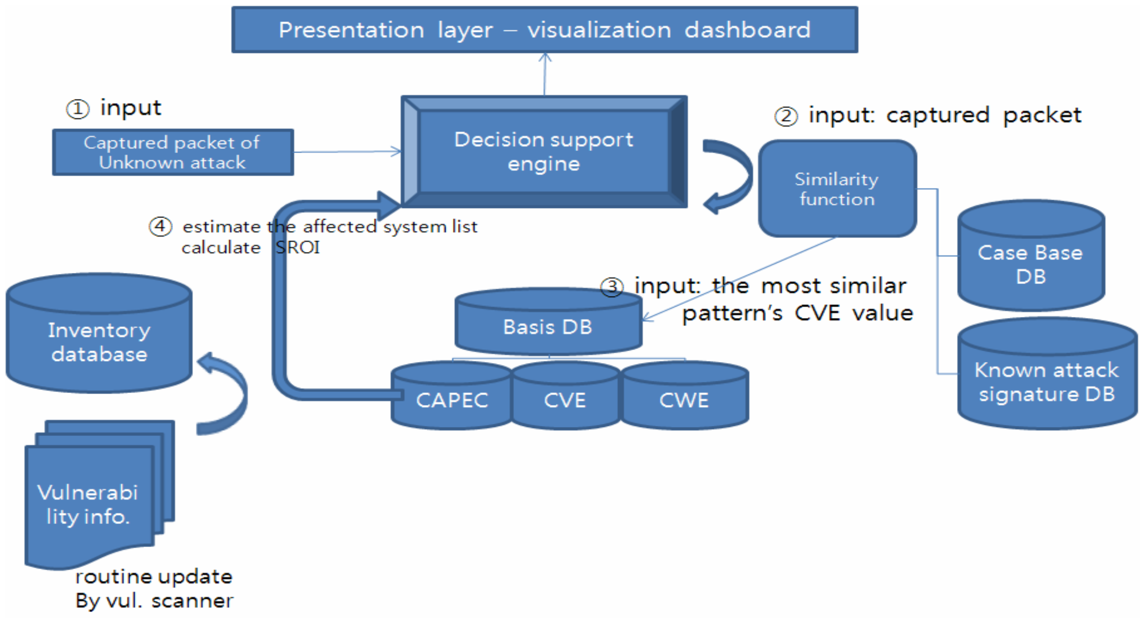
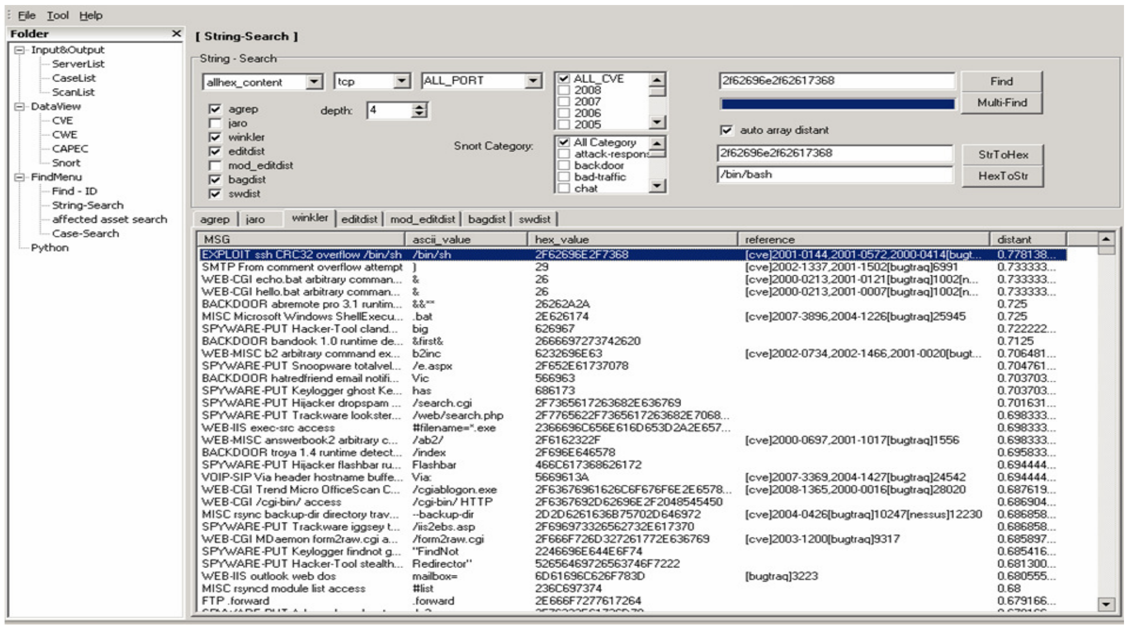


Fig.7 Architecture of overall system



[String-Search]

String - Search

allhex_content | tcp | ALL_PORT

[✓] agrep
[ ] jaro
[✓] winkler
[✓] editdist
[ ] mod_editdist
[✓] bagdist
[✓] swdist

depth: 4

[✓] ALL_CVE
[ ] 2008
[ ] 2007
[ ] 2006
[ ] 2005

Snort Category:
[✓] All Category
[ ] attack-respons
[ ] backdoor
[ ] bad-traffic
[ ] chat

2f62696e2f62617368

[✓] auto array distant

2f62696e2f62617368

/bin/bash

```
MSG                                    ascii_value         hex_value                          reference                                        distant
EXPLOIT ssh CRC32 overflow /bin/sh     /bin/sh             2F62696E2F7368                     [cve]2001-0144,2001-0572,2000-0414[bugt...       0.778138...
SMTP From comment overflow attempt     )                   29                                 [cve]2002-1337,2001-1502[bugtraq]6991            0.733333...
WEB-CGI echo.bat arbitrary comman...   &                   26                                 [cve]2000-0213,2001-0121[bugtraq]1002[n...       0.733333...
WEB-CGI hello.bat arbitrary comman...  &                   26                                 [cve]2000-0213,2001-0007[bugtraq]1002[n...       0.733333...
BACKDOOR abremote pro 3.1 runtim...    &&**                26262A2A                                                                            0.725
MISC Microsoft Windows ShellExecu...   .bat                2E626174                           [cve]2007-3896,2004-1226[bugtraq]25945           0.725
SPYWARE-PUT Hacker-Tool cland...       big                 626967                                                                              0.722222...
BACKDOOR bandook 1.0 runtime de...     &first&             2666697273742620                                                                    0.7125
WEB-MISC b2 arbitrary command ex...    b2inc               6232696E63                         [cve]2002-0734,2002-1466,2001-0020[bugt...       0.706481...
SPYWARE-PUT Snoopware totalvel...      /e.aspx             2F652E61737078                                                                      0.704761...
BACKDOOR hatredfriend email notifi...  Vic                 566963                                                                              0.703703...
SPYWARE-PUT Keylogger ghost Ke...      has                 686173                                                                              0.703703...
SPYWARE-PUT Hijacker dropspam ...      /search.cgi         2F7365617263682E636769                                                              0.701631...
SPYWARE-PUT Trackware lookster...      /web/search.php     2F7765622F7365617263682E7068...                                                     0.698333...
WEB-IIS exec-src access                #filename=*.exe     2366696C656E616D653D2A2E657...                                                      0.698333...
WEB-MISC answerbook2 arbitrary c...    /ab2/               2F6162322F                         [cve]2000-0697,2001-1017[bugtraq]1556            0.698333...
BACKDOOR troya 1.4 runtime detect...   /index              2F696E646578                                                                        0.695833...
SPYWARE-PUT Hijacker flashbar ru...    Flashbar            466C617368626172                                                                    0.694444...
VOIP-SIP Via header hostname buffe...  Via:                5669613A                           [cve]2007-3369,2004-1427[bugtraq]24542           0.694444...
WEB-CGI Trend Micro OfficeScan C...    /cgiablogon.exe     2F63676961626C6F676F6E2E6578...    [cve]2008-1365,2000-0016[bugtraq]28020           0.687619...
WEB-CGI /cgi-bin/ access               /cgi-bin/ HTTP      2F6367692D62696E2F2048545450                                                        0.686904...
MISC rsync backup-dir directory trav...  --backup-dir      2D2D6261636B75702D646972           [cve]2004-0426[bugtraq]10247[nessus]12230        0.686858...
SPYWARE-PUT Trackware iggsey t...      /iis2ebs.asp        2F69697332656562732E617370                                                          0.686858...
WEB-CGI MDaemon form2raw.cgi a...      /form2raw.cgi       2F666F726D327261772E636769         [cve]2003-1200[bugtraq]9317                      0.685897...
SPYWARE-PUT Keylogger findnot g...     "FindNot            2246696E644E6F74                                                                    0.685416...
SPYWARE-PUT Hacker-Tool stealth...     Redirector"         526564697265637466F7222                                                             0.681300...
WEB-IIS outlook web dos                mailbox=            6D61696C626F783D                   [bugtraq]3223                                    0.680555...
MISC rsyncd module list access         #list               236C697374                                                                          0.68
FTP .forward                           .forward            2E666F7277617264                                                                    0.679166...
```

Fig.8 Screenshot of the proposed DSS

IP address	DNS name	operating system	open port list	vulnerability list	related reference (CVE ID, CAN ID, nessus id)	risk factor	last scan end time
143.248. 1.177	unknown	Sun Solaris 10, Sun Solaris 9	22/tcp	DNS bind buffer overflow	CVE : CVE-2005-1794 BID : 13818 Nessus ID : 18405	Medium / CVSS Base Score : 5.1	Tue Jun 24 14:13:20 2008
143.248. 90.220	Apa01	Microsoft Windows XP Service Pack 2	3389/tcp	Microsoft Windows Remote Desktop Protocol Server Private Key Disclosure Vulnerability	CVE : CVE-2005-1794 BID : 13818 Nessus ID : 18405	Medium / CVSS Base Score : 5.1	Tue Jun 24 14:13:20 2008

Table.6 the parsed data from the vulnerability scanning result

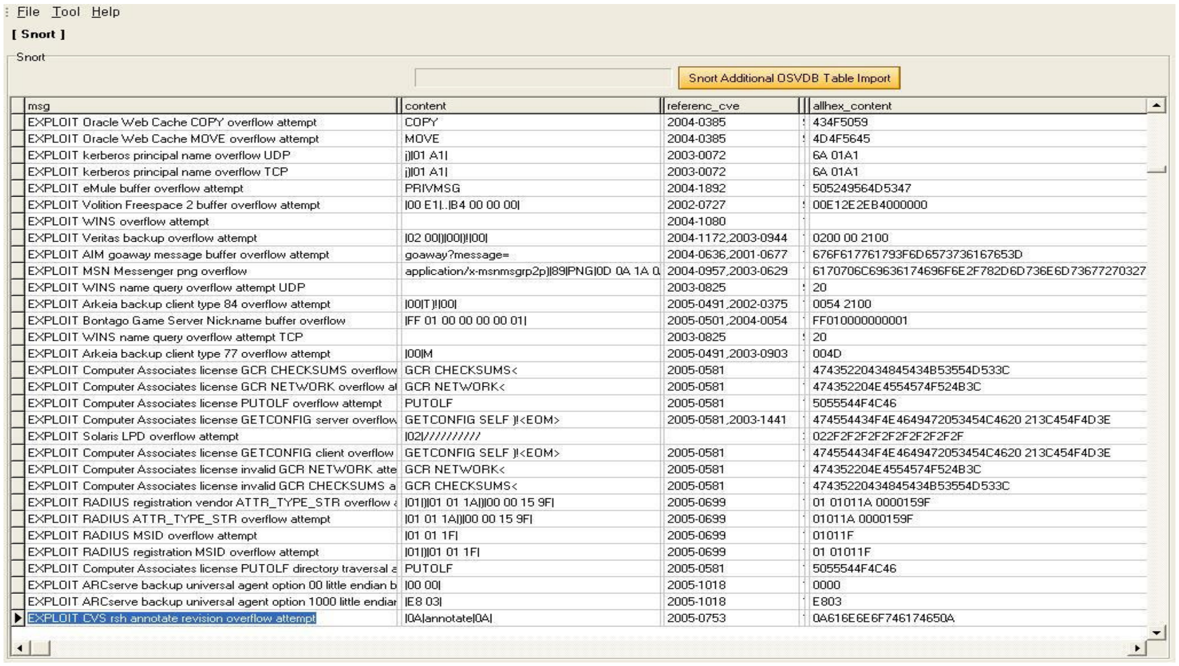
IP address	Subnet mask	Gateway	DNS name	operating system	Major DB /application	Asset Value	Vulnerability found
143.248. 90.2	255.255. 255.0	143.248. 90.1	Host1	Windows 2003 server	MS-SQL 2000 server IIS 6.0	$ 3,000	CVE-1999 -0103
143.248. 90.101	255.255. 255.0	143.248. 90.1	Host2	Windows 2000 server	MS-SQL 2005 server IIS 5.0	$ 4,850	CVE-1999 -0104

Table.7 example data scheme in the information asset database

Fig.9 is a screenshot of the attack signature database based on snort ruleset. It includes known attack signatures, attack category, reference CVE ID and preprocessed hex streams.

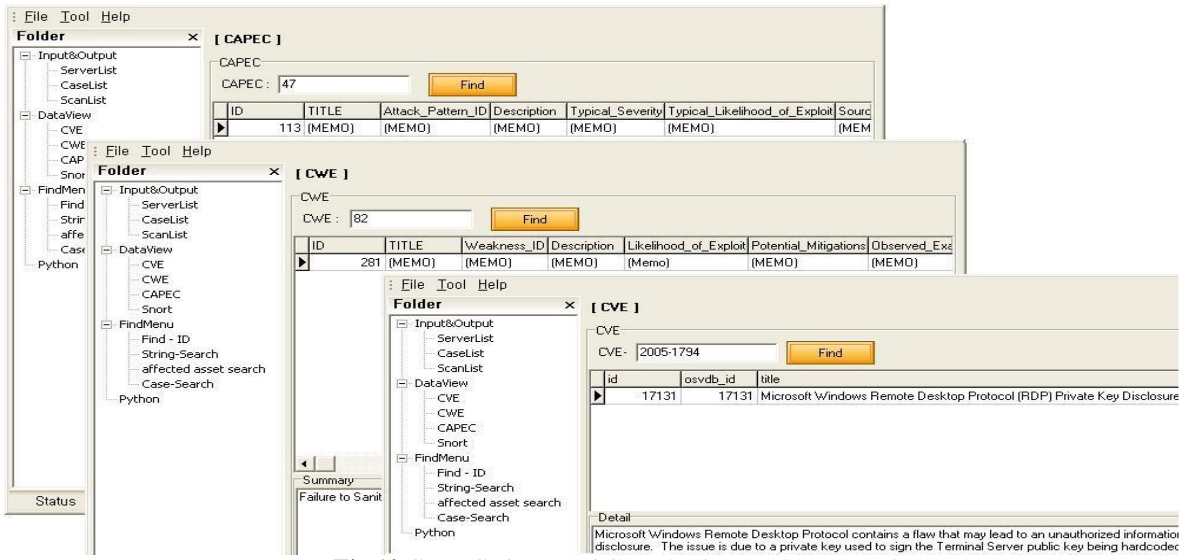
CVE, CWE, and CAPEC database are shown in the Fig.10. CAPEC, CWE database and query tool is built with parsing CAPEC and CWE dictionary files. CVE database is built with processing OSVDB.

Fig.11 shows the main query interface for searching the most similar attack signature. It supports query function which can resolve various search conditions (year, protocol, attack category and strings similarity function). With this tool, we can simulate that what function shows better result in the specific conditions.



msg	content	referenc_cve	allhex_content
EXPLOIT Oracle Web Cache COPY overflow attempt	COPY	2004-0385	434F5059
EXPLOIT Oracle Web Cache MOVE overflow attempt	MOVE	2004-0385	4D4F5645
EXPLOIT kerberos principal name overflow UDP	j|01 A1|	2003-0072	6A 01A1
EXPLOIT kerberos principal name overflow TCP	j|01 A1|	2003-0072	6A 01A1
EXPLOIT eMule buffer overflow attempt	PRIVMSG	2004-1892	505249564D5347
EXPLOIT Volition Freespace 2 buffer overflow attempt	|00 E1|..|B4 00 00 00|	2002-0727	00E12E2EB4000000
EXPLOIT WINS overflow attempt		2004-1080	
EXPLOIT Veritas backup overflow attempt	|02 00|)|00|)!|00|	2004-1172,2003-0944	0200 00 2100
EXPLOIT AIM goaway message buffer overflow attempt	goaway?message=	2004-0636,2001-0677	676F617761793F6D6573736167653D
EXPLOIT MSN Messenger png overflow	application/x-msnmsgrp2p)|89|PNG|0D 0A 1A 0	2004-0957,2003-0629	6170706C69636174696F6E2F782D6D736E6D73677270327
EXPLOIT WINS name query overflow attempt UDP		2003-0825	20
EXPLOIT Arkeia backup client type 84 overflow attempt	|00|T)!|00|	2005-0491,2002-0375	0054 2100
EXPLOIT Bontago Game Server Nickname buffer overflow	|FF 01 00 00 00 00 01|	2005-0501,2004-0054	FF010000000001
EXPLOIT WINS name query overflow attempt TCP		2003-0825	20
EXPLOIT Arkeia backup client type 77 overflow attempt	|00|M	2005-0491,2003-0903	004D
EXPLOIT Computer Associates license GCR CHECKSUMS overflow	GCR CHECKSUMS<	2005-0581	474352204348454348B53554D533C
EXPLOIT Computer Associates license GCR NETWORK overflow at	GCR NETWORK<	2005-0581	474352204E4554574F524B3C
EXPLOIT Computer Associates license PUTOLF overflow attempt	PUTOLF	2005-0581	5055544F4C46
EXPLOIT Computer Associates license GETCONFIG server overflow	GETCONFIG SELF)!<EOM>	2005-0581,2003-1441	474554434F4E4649472053454C4620 213C454F4D3E
EXPLOIT Solaris LPD overflow attempt	|02|//////////		022F2F2F2F2F2F2F2F2F2F
EXPLOIT Computer Associates license GETCONFIG client overflow	GETCONFIG SELF)!<EOM>	2005-0581	474554434F4E4649472053454C4620 213C454F4D3E
EXPLOIT Computer Associates license invalid GCR NETWORK atte	GCR NETWORK<	2005-0581	474352204E4554574F524B3C
EXPLOIT Computer Associates license invalid GCR CHECKSUMS a	GCR CHECKSUMS<	2005-0581	474352204348454348B53554D533C
EXPLOIT RADIUS registration vendor ATTR_TYPE_STR overflow a	|01|)|01 01 1A|)|00 00 15 9F|	2005-0699	01 01011A 0000159F
EXPLOIT RADIUS ATTR_TYPE_STR overflow attempt	|01 01 1A|)|00 00 15 9F|	2005-0699	01011A 0000159F
EXPLOIT RADIUS MSID overflow attempt	|01 01 1F|	2005-0699	01011F
EXPLOIT RADIUS registration MSID overflow attempt	|01|)|01 01 1F|	2005-0699	01 01011F
EXPLOIT Computer Associates license PUTOLF directory traversal a	PUTOLF	2005-0581	5055544F4C46
EXPLOIT ARCserve backup universal agent option 00 little endian b	|00 00|	2005-1018	0000
EXPLOIT ARCserve backup universal agent option 1000 little endiar	|E8 03|	2005-1018	E803
EXPLOIT CVS rsh annotate revision overflow attempt	|0A|annotate|0A|	2005-0753	0A616E6E6F746174650A

Fig.9 Attack signature database



CAPEC : 47

ID	TITLE	Attack_Pattern_ID	Description	Typical_Severity	Typical_Likelihood_of_Exploit	Sourc
113	(MEMO)	(MEMO)	(MEMO)	(MEMO)	(MEMO)	(MEM

CWE : 82

ID	TITLE	Weakness_ID	Description	Likelihood_of_Exploit	Potential_Mitigations	Observed_Exa
281	(MEMO)	(MEMO)	(MEMO)	(Memo)	(MEMO)	(MEMO)

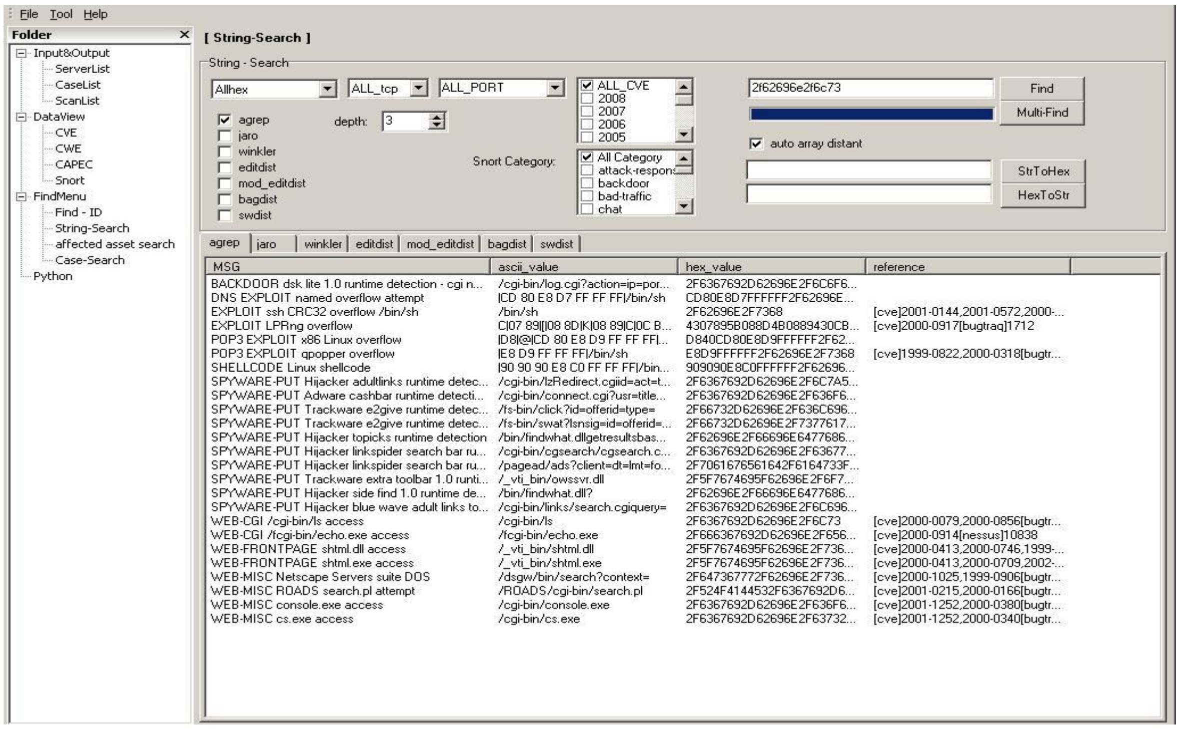
Failure to Sanit

CVE- 2005-1794

id	osvdb_id	title
17131	17131	Microsoft Windows Remote Desktop Protocol (RDP) Private Key Disclosure

Microsoft Windows Remote Desktop Protocol contains a flaw that may lead to an unauthorized informatio
disclosure. The issue is due to a private key used to sign the Terminal Server public key being hardcoded

Fig.10 CAPEC, CWE and CVE database and query tool



[String-Search]

String - Search

Allhex | ALL_tcp | ALL_PORT

[✓ agrep] jaro winkler editdist mod_editdist bagdist swdist

depth: 3

[✓ ALL_CVE] 2008 2007 2006 2005

Snort Category: [✓ All Category] attack-respons backdoor bad-traffic chat

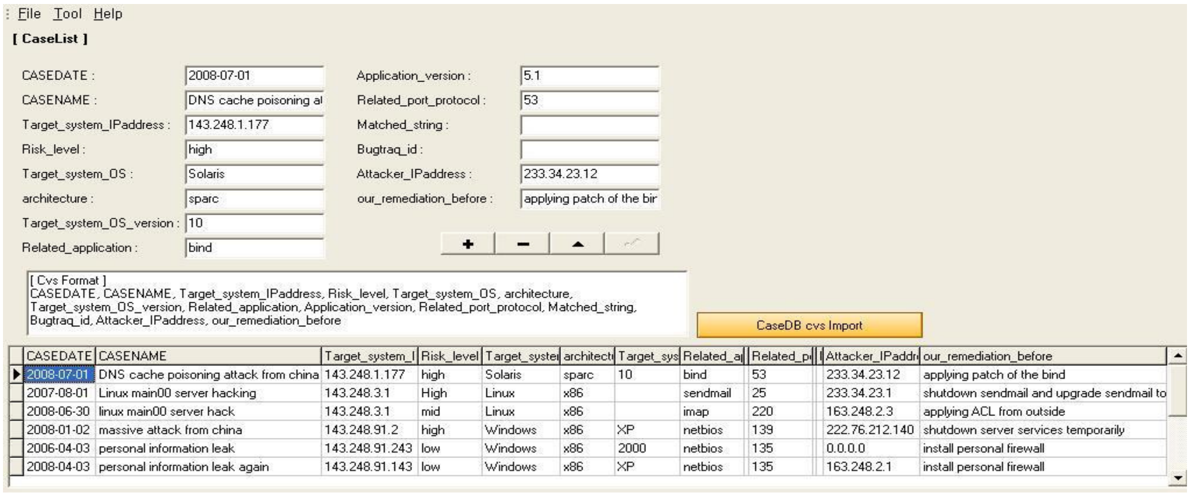
2f62696e2f6c73

[✓ auto array distant]

MSG	ascii_value	hex_value	reference
BACKDOOR dsk lite 1.0 runtime detection - cgi n...	/cgi-bin/log.cgi?action=ip=por...	2F6367692D62696E2F6C6F6...	
DNS EXPLOIT named overflow attempt	|CD 80 E8 D7 FF FF FF|/bin/sh	CD80E8D7FFFFFF2F62696E...	
EXPLOIT ssh CRC32 overflow /bin/sh	/bin/sh	2F62696E2F7368	[cve]2001-0144,2001-0572,2000-...
EXPLOIT LPRng overflow	C|07 89|[|08 8D|K|08 89|C|0C B...	4307895B088D4B0889430CB...	[cve]2000-0917[bugtraq]1712
POP3 EXPLOIT x86 Linux overflow	|D8|@|CD 80 E8 D9 FF FF FF|...	D840CD80E8D9FFFFFF2F62...	
POP3 EXPLOIT qpopper overflow	|E8 D9 FF FF FF|/bin/sh	E8D9FFFFFF2F62696E2F7368	[cve]1999-0822,2000-0318[bugtr...
SHELLCODE Linux shellcode	|90 90 90 E8 C0 FF FF FF|/bin...	909090E8C0FFFFFF2F62696...	
SPYWARE-PUT Hijacker adultlinks runtime detec...	/cgi-bin/lzRedirect.cgiid=act=t...	2F6367692D62696E2F6C7A5...	
SPYWARE-PUT Adware cashbar runtime detecti...	/cgi-bin/connect.cgi?usr=title...	2F6367692D62696E2F636F6...	
SPYWARE-PUT Trackware e2give runtime detec...	/fs-bin/click?id=offerid=type=	2F66732D62696E2F636C696...	
SPYWARE-PUT Trackware e2give runtime detec...	/fs-bin/swat?lsnsig=id=offerid=...	2F66732D62696E2F7377617...	
SPYWARE-PUT Hijacker topicks runtime detection	/bin/findwhat.dllgetresultsbas...	2F62696E2F66696E6477686...	
SPYWARE-PUT Hijacker linkspider search bar ru...	/cgi-bin/cgsearch/cgsearch.c...	2F6367692D62696E2F63677...	
SPYWARE-PUT Hijacker linkspider search bar ru...	/pagead/ads?client=dt=lmt=fo...	2F7061676561642F6164733F...	
SPYWARE-PUT Trackware extra toolbar 1.0 runti...	/_vti_bin/owssvr.dll	2F5F7674695F62696E2F6F7...	
SPYWARE-PUT Hijacker side find 1.0 runtime de...	/bin/findwhat.dll?	2F62696E2F66696E6477686...	
SPYWARE-PUT Hijacker blue wave adult links to...	/cgi-bin/links/search.cgiquery=	2F6367692D62696E2F6C696...	
WEB-CGI /cgi-bin/ls access	/cgi-bin/ls	2F6367692D62696E2F6C73	[cve]2000-0079,2000-0856[bugtr...
WEB-CGI /fcgi-bin/echo.exe access	/fcgi-bin/echo.exe	2F666367692D62696E2F656...	[cve]2000-0914[nessus]10838
WEB-FRONTPAGE shtml.dll access	/_vti_bin/shtml.dll	2F5F7674695F62696E2F736...	[cve]2000-0413,2000-0746,1999-...
WEB-FRONTPAGE shtml.exe access	/_vti_bin/shtml.exe	2F5F7674695F62696E2F736...	[cve]2000-0413,2000-0709,2002-...
WEB-MISC Netscape Servers suite DOS	/dsgw/bin/search?context=	2F647367772F62696E2F736...	[cve]2000-1025,1999-0906[bugtr...
WEB-MISC ROADS search.pl attempt	/ROADS/cgi-bin/search.pl	2F524F4144532F6367692D6...	[cve]2001-0215,2000-0166[bugtr...
WEB-MISC console.exe access	/cgi-bin/console.exe	2F6367692D62696E2F636F6...	[cve]2001-1252,2000-0380[bugtr...
WEB-MISC cs.exe access	/cgi-bin/cs.exe	2F6367692D62696E2F63732...	[cve]2001-1252,2000-0340[bugtr...

Fig.11 String similarity search result

The Fig.12 is the main GUI of case-database menu. It is supported to input, update and search cases. The Fig.13 is the query interface for searching the most similar case. The distance is calculated with the method described in the section 3. The example result of similar case search is in the Fig.14. With CWE, CAPEC and CVE database, our DSS can simulate the relationship and find the related CWE, CAPEC and CVE where specific CWE or CAPEC id is given.



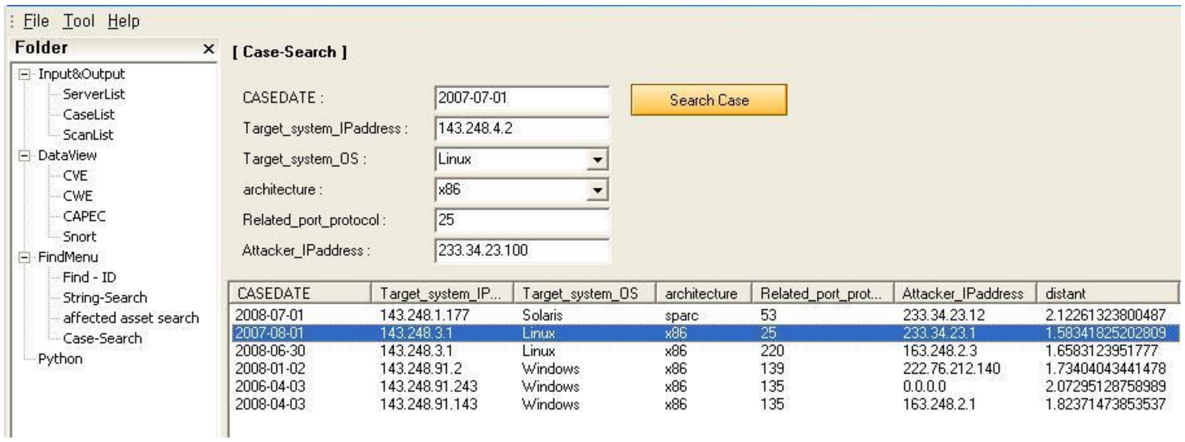
[CaseList]

CASEDATE :	2008-07-01	Application_version :	5.1
CASENAME :	DNS cache poisoning at	Related_port_protocol :	53
Target_system_IPaddress :	143.248.1.177	Matched_string :	
Risk_level :	high	Bugtraq_id :	
Target_system_OS :	Solaris	Attacker_IPaddress :	233.34.23.12
architecture :	sparc	our_remediation_before :	applying patch of the bir
Target_system_OS_version :	10		
Related_application :	bind		

[Cvs Format]
CASEDATE, CASENAME, Target_system_IPaddress, Risk_level, Target_system_OS, architecture, Target_system_OS_version, Related_application, Application_version, Related_port_protocol, Matched_string, Bugtraq_id, Attacker_IPaddress, our_remediation_before

CASEDATE	CASENAME	Target_system_I	Risk_level	Target_syste	architectu	Target_sys	Related_a	Related_p	I	Attacker_IPaddr	our_remediation_before
2008-07-01	DNS cache poisoning attack from china	143.248.1.177	high	Solaris	sparc	10	bind	53		233.34.23.12	applying patch of the bind
2007-08-01	Linux main00 server hacking	143.248.3.1	High	Linux	x86		sendmail	25		233.34.23.1	shutdown sendmail and upgrade sendmail to
2008-06-30	linux main00 server hack	143.248.3.1	mid	Linux	x86		imap	220		163.248.2.3	applying ACL from outside
2008-01-02	massive attack from china	143.248.91.2	high	Windows	x86	XP	netbios	139		222.76.212.140	shutdown server services temporarily
2006-04-03	personal information leak	143.248.91.243	low	Windows	x86	2000	netbios	135		0.0.0.0	install personal firewall
2008-04-03	personal information leak again	143.248.91.143	low	Windows	x86	XP	netbios	135		163.248.2.1	install personal firewall

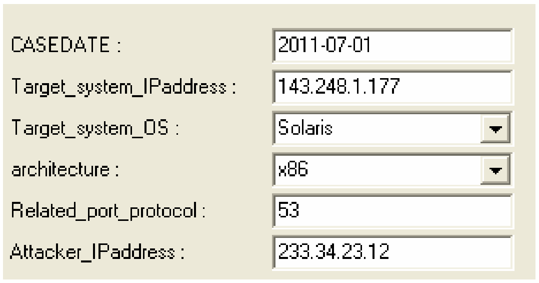
Fig.12 Case-base which includes the attack history



CASEDATE : 2007-07-01
Target_system_IPaddress : 143.248.4.2
Target_system_OS : Linux
architecture : x86
Related_port_protocol : 25
Attacker_IPaddress : 233.34.23.100

CASEDATE	Target_system_IP...	Target_system_OS	architecture	Related_port_prot...	Attacker_IPaddress	distant
2008-07-01	143.248.1.177	Solaris	sparc	53	233.34.23.12	2.12261323800487
2007-08-01	143.248.3.1	Linux	x86	25	233.34.23.1	1.58341825202809
2008-06-30	143.248.3.1	Linux	x86	220	163.248.2.3	1.6583123951777
2008-01-02	143.248.91.2	Windows	x86	139	222.76.212.140	1.73404043441478
2006-04-03	143.248.91.243	Windows	x86	135	0.0.0.0	2.07295128758989
2008-04-03	143.248.91.143	Windows	x86	135	163.248.2.1	1.82371473853537

Fig.13 The query input for the case search



CASEDATE : 2011-07-01
Target_system_IPaddress : 143.248.1.177
Target_system_OS : Solaris
architecture : x86
Related_port_protocol : 53
Attacker_IPaddress : 233.34.23.12

Fig.14 Example screenshot of the case search

5 Experiment Result

5.1 Experiment design

We design experiment to estimate the proposed DSS's accuracy based on the following propositions.

Proposition 1. New attacks (zero-day attacks) can be estimated with the known old attacks.

The number of vulnerability found is enough size to ensure the significance. The known attack data for each year from 1999 are used. The number of vulnerability data is more than 800 for every selected year. We gain the attacks' disclosure date from the OSVDB, this disclosure date is imported into our attack signature database.

The attacks occurred in the year Y is set as zero-day attack test data. The base data are the attacks occurred in 1999 to Y-1.

That is attacks occurred in Y year is the zero-day attacks in the past years (Y-1, Y-2, …), hence the estimation power can be gained with comparing attack data of specific year into the past years. In case of this, if the similarity search metrics find the attacks in a same attack category and same application, then we regard the attack could be estimated in the past years.

Proposition 2. Each string similarity function has its own strength to comparing string data. We will find which function is suitable for some selected protocols through this experiment.

When the similarity metrics find out the known attacks for given unknown attack, the scoring is given as following rules.

Result type	Assessment condition for success/failure
Success	The similarity metrics should find the most similar pattern within a 'same attack category' and 'same target application' , where the similarity value > 0.7 Or The similar patterns in top 3 similarity value by the similarity metrics should be within a 'same attack category' and 'same target application' , when the best similarity value < 0.7
Failure	There are no matches which satisfy the success condition

Table.8 Assessment condition for performance estimation

In case of success, mark the similarity value at that time as a measure. In case of failure, mark the similarity value as zero.

We have experimented some representative applications and protocols – – HTTP, ORACLE, HTTP, VOIP, MSSQL, SMTP, FTP, TELNET and SNMP.

5.2 Experimental results

Every result is summarized as the following tables. The mark ★ means that these similarity measures estimate the exact protocol and related application. The score without ★ means it only estimate the protocol or attack category only. The number (between 0 to 1) means similarity.

This result shown in the Table.9 is gained as follows.

① Set zero-day attack as 'Oracle database' related attack signatures happened in 2007.

② Input attack string in the snort vulnerability ID patterns of the past. (in this case, 1999 to 2006)

③ Score the result after calculation is done by similarity metrics.

Here is the estimated result by various similarity metrics.

Jaro-style metrics (Jaro and Jaro-Winkler) show the better performance rather than Levenshtein-style metrics. In this case, Jaro-Winkler is the winner.

Snort vul ID	Jaro	JaroWinkler	Editdist	Mod editdist	Bagdist	swdist
7867	★1	★1	★1	★1	★1	★1
8528	0.714	0.714	0.219	0.219	0.286	0.25
8555	★1	★1	★1	★1	★1	★1
8749	★0.781	★0.869	★0.344	★0.344	★0.344	★0.512
9110	★1	★1	★1	★1	★1	★1
8545	0.658	0.658	0.448	0.448	0.655	0.279
9111	0.596	0.633	0.217	0.217	0.304	0.12
8502	★1	★1	★1	★1	★1	★1
8544	★1	★1	★1	★1	★1	★1

Table.9 estimation result: putting Oracle attack patterns of 2007 into IDS with 1999~2006 signatures

Table.10 shows the estimated result when the zero-day attacks are mimicked with attacks on HTTP protocol which is exposed in 2008.

Jaro-style metrics (Jaro and Jaro-Winkler) show the better performance rather than Levenshtein-style metrics. In this case, Jaro-Winkler is the winner. But other metrics show the better performance than ORACLE related attack.

Snort vul ID	Jaro	JaroWinkler	Editdist	Mod editdist	Bagdist	swdist
8563-1	0.75	0.775	0.625	0.625	0.625	0.375
8564-1	0.75	0.85	0.5	0.5	0.75	0.5
7864	★0.718	★0.831	0.5	0.5	0.812	★0.306
9114	★0.636	★0.709	★0.352	★0.352	★0.574	★0.122
8562	★0.974	★0.985	★0.923	★0.923	★0.923	★0.864
8573	★0.819	★0.892	★0.625	★0.625	★0.625	★0.714
8563,8564	0.952	0.971	0.857	0.857	0.857	0.923
9112	0.579	0.621	0.313	0.313	0.313	0.167
8565	0.704	0.822	0.132	0.132	0.170	0.203
8569	★0.556	★0.556	★0.241	★0.241	★0.414	★0.089
8539	0.722	0.806	0.533	0.533	0.767	0.367
9113	★1	★1	★1	★1	★1	★1

Table.10 estimation result: putting HTTP attack patterns of 2008 into IDS with 1999~2007 signatures

Snort vul ID	Jaro	JaroWinkler	Editdist	Mod editdist	Bagdist	Swdist
6337	★1	★1	★1	★1	★1	★1
6338	★1	★1	★1	★1	★1	★1
6339	★1	★1	★1	★1	★1	★1
6340	★1	★1	★1	★1	★1	★1
6341	★1	★1	★1	★1	★1	★1
6342	★1	★1	★1	★1	★1	★1
6343	★0.775	★0.865	★0.649	★0.649	★0.649	★0.787
6344	★0.853	★0.912	★0.6	★0.6	★0.6	★0.619
6345	★1	★1	★1	★1	★1	★1
6346	0.761	0.761	0.517	0.517	0.7	0.349
6348	★1	★1	★1	★1	★1	★1
6350	★1	★1	★1	★1	★1	★1
6351	★1	★1	★1	★1	★1	★1
6352	★1	★1	★1	★1	★1	★1
6353	★1	★1	★1	★1	★1	★1
6354	★1	★1	★1	★1	★1	★1

Table.11 estimation result: putting SMTP attack patterns of 2007 into IDS with 1999~2006 signatures

Table.11 shows the result of estimation SMTP zero-day attacks in annual base.

For comparing result conventionally, we give additional 1 more score to the result with ★. (in case of ★0.85 we change the score as 1.85.). We illustrate these results as follows.

In case of SMTP, Jaro-Winkler shows the better performance rather than others (See Table.12). Likewise, the experiment for the other protocols is performed as shown in Table .13.

year	Jaro	JaroWinkler	Editdist	Mod editdist	Bagdist	Swdist
2007	1.8993125	1.908625	1.860375	1.860375	1.8718125	1.8596875
2006	2	2	2	2	2	2
2005	0.9724	1.0092	0.7254	0.7254	0.8704	0.688
2004	1.6604286	1.6960476	1.4152857	1.4152857	1.469619	1.475381
mean	1.6330353	1.6534682	1.5002652	1.5002652	1.5529579	1.5057671

Table.12 overall performance in case of SMTP

year	Jaro	JaroWinkler	Editdist	Mod editdist	Bagdist	swdist
2007	2	2	2	2	2	2
2006	1.9723333	1.9833333	1.9583333	1.9583333	1.9583333	1.9166667
2005	2	2	2	2	2	2
2004	1.8746667	1.8911667	1.8020833	1.8020833	1.8333333	1.4854167
mean	1.96175	1.968625	1.9401042	1.9401042	1.9479167	1.8505208

Table.13 overall performance in case of FTP

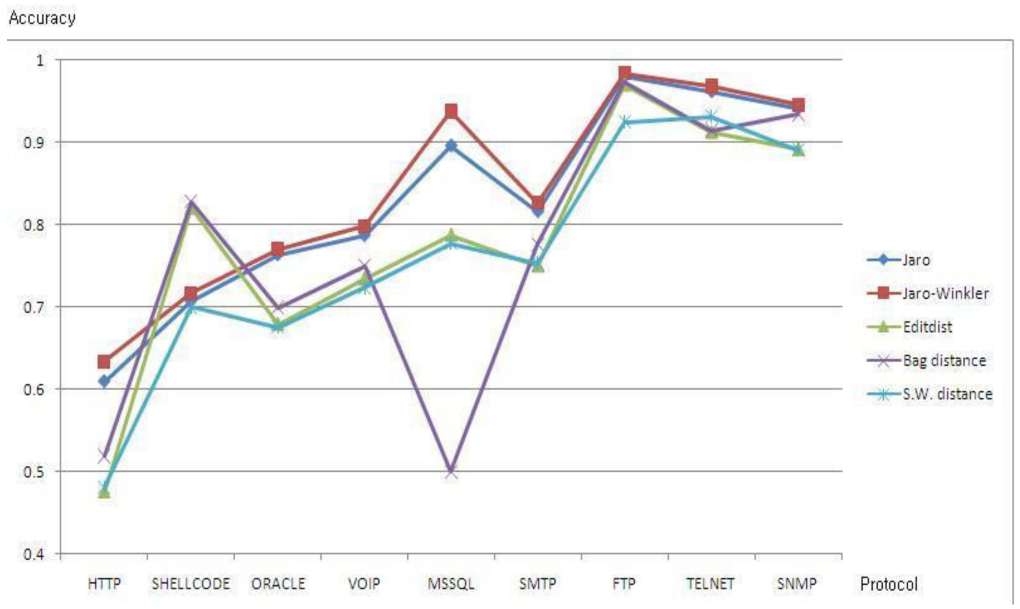


Fig.15 Overall performance of similarity algorithm for the selected protocols and applications

After rescaling the mean value as 0 to 1, we summarized the all experiment result in the Fig.15.

Each function shows different performance in terms of the characteristics of strings. Every attack string pattern is dependent on the protocol and target application. Fig.15 shows the performance of similarity algorithms for the each protocol and application. Jaro and Jaro-Winkler metrics show better performance rather than other metrics except the case of 'SHELLCODE' detection. Also, Edit-distance and modified Edit-distance show better performance rather in case of SHELLCODE detection. Especially, Jaro and Jaro-Winkler have strong estimation power in comparison with other metrics, in case of the estimation for the attacks targeted on the database systems (e.g. ORACLE and MSSQL).

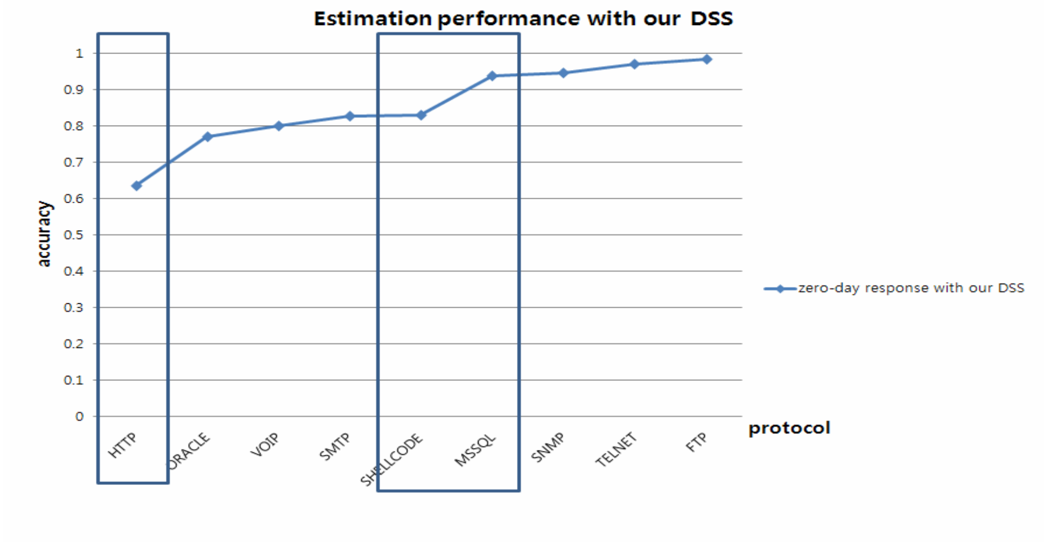


Fig.16 Summarized Overall performance of zero-day attack detection

Fig.16 illustrates the selective best-algorithm's results. The detection power for zero-day attack for HTTP is 63%, where the detection power for the other protocols is more than 80% at least, and up to 99%. However, when considering the variety of web-based attack, we can tell this result shows high level of accuracy. Also, zero-day attacks targeted for MS-SQL, SNMP, TELNET and FTP are easily detected by the proposed system with a high accuracy more than 94%.

6 Conclusion

6.1 Contribution of this study

Currently existing IDS and security risk analysis methods applied could not show the impact of attack exposure on information assets of companies. In the proposed DSS of this research, we adopt the concept of atomic vulnerability and atomic attack. From this, we suggest a way to measure the direct or indirect impact of the attacks on information assets with applying CVE, CAPEC, and CWE. This is the one of contributions of this research.

We also propose two kinds of methods that can be applied in the verification of the attacks and countermeasure of it when the new unknown attacks are detected. For the first method, we suggest a new way to find the most similar patterns from comparing the new attack's text strings in packet payload with that of the known attacks. For this, we adopt text mining and string search algorithms of information retrieval and text mining areas. With experimental analysis, we could show that certain functions perform well for certain protocols.

Second, we propose framework that applies CBR in finding countermeasure for the unknown attacks.

In summary, the DSS proposed in this study is especially effective in making zero-day attack response.

6.2 Future work

To find the most similar patterns from comparing the new attack's text strings in packet payload with that of the known attacks, we adopt text mining and string search algorithms of information retrieval and text mining areas. In our experiment, we focus on which known distance algorithm fits well with the selected well-known protocols. In the future experiment, it is possible to develop more accurate algorithm for distance measure.

Acknowledgement

This research was supported by the Ministry of Knowledge Economy, Korea, under the "ITRC" support program supervised by the National IT Industry Promotion Agency (NIPA-2011-C1090-1001-0004).

References

[1] Noel, S., Wijesekera, D. and Youman, C., Applications of data mining in computer security, *Kluwer Academic Publishers*. (2002)3-25.

[2] Liu, X., Fang, C. and Xiao, D., Intrusion diagnosis and prediction with expert system. *Security and Communication Networks* (2011).

[3] Soewito, B., Vespa, L., Weng, N. and Wang, H., Hybrid pattern matching for trusted intrusion detection, *Security and Communication Network*. 4(2011)33-43.

[4] Barbara, D., and Jajodia, S., Applications of Data Mining in Computer Security, *Kluwer Academic Publishers*. 2002 6-7.

[5] Eskin, E., Anomaly detection over noisy data using learned probability distributions, *Proceedings of 17th International Conference on Machine Learning (ICML).* (2000).

[6] Warrender, C., Forrest, S. and Pearlmutter, B., Detecting Intrusions using System calls: Alternative Data Models. *IEEE Symposium on Security and Privacy*. (1999) 133-145.

[7] Kruegel, C., Mutz, D., Robertson, W. and Valeur, F., Bayesian event classification for intrusion detection, *Proceedings of 19th Annual Computer Security Applications Conference* (ACSAC). (2003) 14-23.

[8] HyungJong K., Huy K. K. and Hae Y. L., Security Requirement representation method for confidence of systems and networks, *International Journal of Software Engineering and Knowledge Engineering*. 20 (2010) 49-71.

[9] Bishop, M., Vulnerabilities Analysis, *Proceedings of the Recent Advances in Intrusion Detection.* (1999) 125-136.

[10] r0ut3r, GoodTech SSH Server SFTP Multiple Command, Handling Overflow, http://www.milw0rm.com/exploits/6804, 2008.

[11] Cohen, F., 50 ways to defeat your intrusion detection system. 1997. http://all.net/journal/netsec/1997-12.html.

[12] CAPEC(Common Attack Pattern Enumeration and Classification). MITRE cooperation. http://capec.mitre.org.

[13] CWE (Common Weakness Enumeration). http://cwe.mitre.org. MITRE cooperation.

[14] CVE (Common Vulnerability Exposure). http://cve.mitre.org. MITRE cooperation.

[15] Wang, L., Noel, S. and Jajodia, S., Minimum-cost network hardening using attack graphs, *Computer Communications*. 29 (2006) 3812-3824.

[16] Jajodia, S., Noel, S. and O'Berry, B., Topological analysis of network attack vulnerability, *Massive Computing*. 5 (2005) 247-266.

[17] Noel, S. and Jajodia, S., Managing attack graph complexity through visual hierarchical aggregation, *Conference on Computer and Communications Security, Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security (VizSEC/DMSEC)*.(2004) 109-118.

[18] Rieke, R., Tool based formal Modeling, Analysis and Visualization of Enterprise Network Vulnerabilities utilising Attack Graph Exploration, *European Institute for Computer Antivirus Research (EICAR)*. 2004.

[19] Musman, S. and Flesher, P., System or Security Managers Adaptive Response Tool, *DARPA Information Survivability Conference & Exposition (DISCEX)*. 02 (2000) 56-68.

[20] Joo, D., Hong, T., Lee, S. and Han, I., A comparison of Case-based Reasoning and Neural Network Model for Intrusion Detection, *Computer and Security*. (2003).

[21] Huy K. K., Kwang H. I. and Sang C. P., DSS for computer security incident response applying CBR and collaborative response, *Expert Systems with Applications*. 37 (2010) 852-870.

[22] Jaro, M. E., Advances in record linking methodology as applied to the 1985 census of Tampa Florida, *Journal of the American Statistical Society*. 84 (1989) 414-420.

[23] Winkler, W. E., The state of record linkage and current research problems, Statistics of Income Division, *Internal Revenue Service Publication*. R99/04 (1999).

[24] Smith, T. F. and Waterman, M. S., Identification of Common Molecular Subsequences, *Journal of Molecular Biology*. 147 (1981) 195-197.

[25] Cohen, W., Ravikumar, P. and Fienberg, S. E., A Comparison of String Distance Metrics for Name-Matching Tasks, *KDD Workshop on Data Cleaning and Object Consolidation*. (2003) 73-78.

[26] Changsok Y., Huy K. K., Eunnyeong H., The Economic Value of Online Game Developers in Early Stages, *Journal of Future Game Technology*. 1, 1(2011) 21-34.

[27] Dong-bum Lee, Smartwork Security Framework with Secure Access Control, *Journal of The Korea Knowledge Information Technology Society*, 6, 4(2011)17-26.

[28] Dae-Sik Park, Smartcard-based Remote User Authentication Protocol with Improved Security, *Journal of The Korea Knowledge Information Technology Society*, 6, 4(2011)27-34.

[29] Ki-Seok Bang, Classification Criteria and Application Methodology for Evaluating IT Security Products, *Journal of The Korea Knowledge Information Technology Society*, 6, 5(2011)105-112.

Huy Kang Kim received his Ph.D. in Industrial and Systems Engineering from Korea Advanced Institute of Science and Technology (KAIST) in 2009. Currently he is an assistant professor in Graduate School of Information Security, Center for Information Security Technologies (CIST) in Korea University. His research interests include Botnet Detection, Intrusion Detection System, Network Forensics and Online Game Security. Contact him at cenda@korea.ac.kr

Soo-Kyun Kim received Ph.D. in Department of Computer Science & Engineering from Korea University in 2006. He joined Telecommunication R&D center at Samsung Electronics Co., Ltd., from 2006 and 2008. He is now a professor at Department of Game Engineering at Pai Chai University, Korea. His research interests include multimedia, pattern recognition, image processing, mobile graphics, geometric modeling, and interactive computer graphics. Contact him at kimsk@pcu.ac.kr

Seok-Hun Kim received Ph.D. in Department of Computer Engineering from Hannam University, Daejeon, Korea in 2006. He joined ParagonBase Co., Ltd., from 2007 and 2010. Since then he has been an adjunct professor in Computer Engineering at Mokwon University, and also employed by TIME SYSTEM Co., Ltd., Korea. His research interests include mobile computing, information security, convergence technology. Contact him at shkim@mokwon.ac.kr