情報保護學會論文誌
第16卷 第4號, 2006. 8

호스트 침해 발생 시점에서의 효율적 Forensics 증거 자료 수집 방안

최 윤 호,[1†] 박 종 호,[1] 김 상 곤,[1] 서 승 우,[1‡] 강 유,[2] 최 진 기,[2] 문 호 건,[2] 이 명 수[2]
[1]서울대학교 전기컴퓨터 공학부, [2]KT

An Efficient Method of Forensics Evidence Collection at the Time of Infringement Occurrence

Yoon-Ho Choi,[1†] Jong-Ho Park,[1] Sang-Kon Kim,[1] Seung-Woo Seo,[1‡]
Yu Kang,[2] Jin-Gi Choe,[2] Ho-Gun Moon,[2] Myung-Su Rhee[2]
[1]School of EECS at Seoul National University, [2]KT

요 약

컴퓨터 Forensics는 급증하고 다양화 되어 가는 컴퓨터 관련 범죄가 발생할 시, 침입에 대한 전자 증거자료를 수집하고 분석함으로써 악의적 사용자를 찾아내는 분야로서, 최근 이에 관한 많은 연구가 진행되고 있다. 그러나 지금까지는 사건 발생 접수 후 전자 증거자료를 수집하는 방안에 대한 연구가 이루어져왔다. 본 논문에서는 사이버 범죄에 적절하게 대응하기 위해 악의적 사용자에 의해 고의적으로 시스템이 침해된 경우, 사건 발생 시점에 기초하여 양질의 증거자료를 효과적으로 수집하기 위한 방안에 대해 제안한다. 이를 위해 침입 탐지시스템(IDS)의 로그와 분석(감시 및 보호)대상 호스트에서의 로그 및 환경 설정 정보의 상관관계를 분석하는 기법을 제시한다. 제안한 기법은 이종 시스템 로그 간 상관관계 분석을 통해 범죄 대응을 위한 자료 손실을 최소화하기 위해, 감시 및 보호 대상 호스트들의 공격에 대한 침해 위험도를 계산하고 이를 기초로 호스트의 침해(실제 시스템이 위험에 노출)발생 시점에서 증거자료를 수집한다. 이를 통해, 침해 분석에 사용되는 분석 대상 자료의 양을 줄일 뿐만 아니라 침해 판단에 사용되는 자료의 손상을 최소화하여 판단의 정확성을 보장한다. 또한 정상적인 사용자나 공격자에 의한 전자증거자료의 훼손을 최소화한다.

ABSTRACT

The Computer Forensics is a research area that finds the malicious users by collecting and analyzing the intrusion or infringement evidence of computer crimes such as hacking. Many researches about Computer Forensics have been done so far. But those researches have focussed on how to collect the forensic evidence for both analysis and proofs after receiving the intrusion or infringement reports of hosts from computer users or network administrators. In this paper, we describe how to collect the forensic evidence of good quality from observable and protective hosts at the time of infringement occurrence by malicious users. By correlating the event logs of Intrusion Detection Systems(IDSes) and hosts with the configuration information of hosts periodically, we calculate the value of infringement severity that implies the real infringement possibility of the hosts. Based on this severity value, we

접수일: 2006년 4월 19일 ; 채택일: 2006년 7월 19일
† 주저자, yuno@cnslab.snu.ac.kr
‡ 교신저자, sseo@snu.ac.kr

selectively collect the evidence for proofs at the time of infringement occurrence. As a result, we show that we can minimize the information damage of the evidence for both analysis and proofs, and reduce the amount of data which are used to analyze the degree of infringement severity.

Keywords : *Computer Forensics, Infringement Severity, Evidence Collection*

Ⅰ. 서 론

최근 보고에 따르면[1], 개인정보침해 및 개인정보 누출 등의 피해에 대한 신고 접수가 꾸준히 증가하고 있으며, 인터넷을 통한 거래가 늘면서 심각한 컴퓨터 범죄에 까지 이르는 경우들이 발생하고 있다. 그러나 이러한 범죄 발생 시 수사를 위한 증거 자료의 부족으로 이에 대해 적절히 대처하지 못하고 있다. 따라서 이러한 범죄 발생과 관련된 전자적 증거 자료(이후 '증거 자료'라 칭함)를 수집, 분석, 추론하고 추적하는 컴퓨터 Forensics (이후 'Forensics'라 칭함)가 새로운 관심 분야로 떠오르면서 그 중요성이 강조되고 있다.

기존의 Forensics의 경우에는(그림 1-(a)), 사건 발생 보고 시점을 기준으로 관련 증거 자료를 수집/보관/분석해 왔다. 즉, 관련 시스템의 사용자 혹은 관리자에 의한 침해 사고신고 접수 시점을 기준으로 대상 시스템에 대한 과거 일정 기간 동안 저장된 증거 자료를 수집 분석하였다. 그러나 이 과정에서 분석 자료가 방대해지고 손상되는 경우가 발생했다. 또한 공격자에 의한 증거자료 훼손 및 변조에 적절히 대응할 수 없었으며, 정상적 사용자의 다른 프로그램 동작 등으로 인한 저장된 증거 자료 훼손의 가능성을 최소화 하지 못했다. 이러한 문제점은 공격자에 의한 고의적 시스템 침해가 발생하는 경우 더욱 더 심각해진다. 이에 본 논문에서는 악의적 공격자에 의한 고의적 호스트(리눅스 시스템) 침해가 발생하는 경우, 감시 및 보호 대상(즉, 분석 대상들)으로부터 수집되는 침해 판단을 위한 분석 자료의 양과 손상정도를 최소화하기위한 방안을 제안한다(그림 1-(b)).

이 논문의 구성은 다음과 같이 이루어진다. 단락 2에서 기존 연구와의 비교 분석을 통해 제안된 방안에 대해 개괄적으로 기술한다. 단락 3에서는 이 논문에서 제안하는 SNORT와 호스트의 로그 및 보안 설정에 관한 정보를 이용하여, 단계별 공격에 대한 대상 호스트의 공격 단계별 침해 사실을 판단하고 Forensics정보를 수집하기 위한 방안에 대해 기술한다. 단락 4에서는 제안되어진 방안의 성능검증 결과를 기술한다. 끝으로, 단락 5에서 이 논문을 요약하고 결론을 맺도록 한다.

Ⅱ. 제안된 방안에 대한 기술

기존의 침입 탐지에 대한 연구는, 침입 로그(alert)의 통합(aggregation)과 상관 또는 연계(correlation) 을 통해 침입시도를 탐지하기 위한 상관관계 분석에 초점이 맞춰졌다. EMERALD architecture의 일부분으로 구현되어진 EMERALD[3]가 제안되어진 이래로, Hervé Debar는 새로이 수집 되어진 alert에 대해, 이전에 동일한 alert 이 수집되었는지 확인(duplicate operation)하고 두 수집된 로그간의 인과관계(consequence operation)에 대해 정의한다. 이를 통해 수집되고 통합된 alert들의 주어진 severity를 합산한 weight를 침입 탐지에

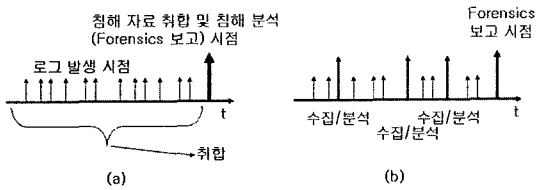


그림 1. 침해사고 보고 비교: (a)기존(취합), (b)제안(수집)

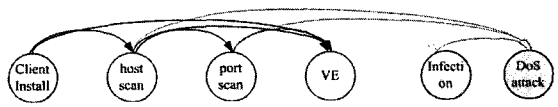


그림 2. 공격의 단계 별 침해 상태 및 가능한 공격 흐름(검은 색: 침해초기, 주황색: 네트워크침해경로설정, 보라색: 최종공격), Client: 공격자, DoS(Denial of Service): 5 단계로 이루어진 다단계 공격의 최종 공격 예

표 1. 기존 상관관계 연구 결과와의 비교

	기존 연구	제안된 방안
분석 목표	• 침입시도탐지 • 사용자 침해 보고 시점에서의 Forensics 증거 자료 취합	• 호스트 침해 탐지 • 호스트의 침해 시점에 기반 한, Forensics 증거 자료 수집
분석 기법 및 수단	• 호스트 로그 간 상관관계 분석 • IDS로그 간 상관관계 분석	• 호스트 로그 및 IDS 로그의 상관관계 분석 • 호스트 침해 위험도 탐지
효능	• 정탐 (오탐율 및 미탐율 감소)	• 호스트의 실제 침해도 산출 • 호스트 침해 시점에서, 침해 판단 및 Forensics 증거 자료 수집
한계점	• 증거 자료 손실 • 침입 시도에 기반 하므로, 실제 침해 사실이 없는 시스템에 대해서도 분석이 필요 • 휘발성 데이터 수집 방안 부재	• 생성 후 소멸시간이 제안된 수집 방안의 응답시간보다 빠른 휘발성 데이터에 대한 수집 방안 부재

이용한다[4]. Benjamin Morin이 제안한 M2D2[5]의 경우, 감시 대상 호스트의 특징과 취약점 정보를 고려함으로써 위의 방안을 확장시킨다. 또한, 이외에도 Frederic Cuppens의 경우[6], '통합'을 alert 융합과정으로 '상관'을 '악의적 의도 인식'으로 정의한 뒤, IDS에 의해 발생 되어진 관찰에 상응하는 가능한 scenario 사례의 집합을 정의하고 이를 침입 탐지에 이용한다.

이러한 논문들은 IDS의 미탐(False Negative) 및 오탐(False Positive)을 최소화하기 위한 방안에 초점을 맞추고 있다. 그로인해, 침입자의 침입 시도 자체를 탐지하는데 목적을 두고 있다. 반면, 호스트에 대한 Forensics증거자료 수집을 위한 침해 시점 판단을 위해서는 호스트의 실제 침해 시점을 기초로 관련 로그 간 상관관계를 분석하고 호스트의 침해 유무를 판단할 필요가 있다.

이에 본 논문에서는, 호스트에 대한 공격이 단계별로 이루어지는 경우(다단계 공격)에 대해(예제: 그림 2), 증거 자료 미 수집상황을 최소화하기 위해 침해 초기단계에서(Host scanning, Port scanning, Vulnerability Exploit(VE))부터 최종 공격 단계(예: DDoS (Distributed Denial of Service)) 까지 각 단계별 관련 Forensics정보를 수집하는 방안에 대해 기술한다. 이를 위해 공격자에 의한 침입을 침입 시도 및 침해(시스템에 대한 실제 침입이 발생) 관점으로 구분하고, 다단계 공격에 대한 분석 대상 호스트의 침해 발생 시점에서의 각 단계별 침해 위험도를 계산하여 단계별 증거자료 수집을 위한 시점을 판단한다. 위험도 계산 시 잘못된 판단을 최소화하기 위해, 침입 탐지 시스템(IDS)인 SNORT[2]의 로그 분석을 통해 침입 시도를 판단하고, 호스트의 로그(이벤트 로그) 및 보안 설정(configuration) 정보간의 상관관계 분석을 통해 침해 사실을 판단한다. 한편 대상 호스트의 침해에 대한 판단 기준 자료 및 증거 자료는 공격 단계별 상태정보로 표현한다. 표 1에서, 이 논문의 기여도를 요약하기 위해, 제안되어진 방안과 기존 방안의 비교 분석 결과를 기술한다.

III. 침해 판단 및 Forensics 정보 수집을 위한 방안

이 장에서는 그림 2와 같은 5단계로 이루어진 일반적인 '다단계 시스템 공격 시나리오'를 가정하고, 분석 대상 호스트의 실제 침해 시점에 근거한 SNORT 로그와 호스트 로그 및 설정 정보의 상관관계 분석을 통한 침해 판단 방안과 Forensics정보 수집 방안에 대해 기술한다.

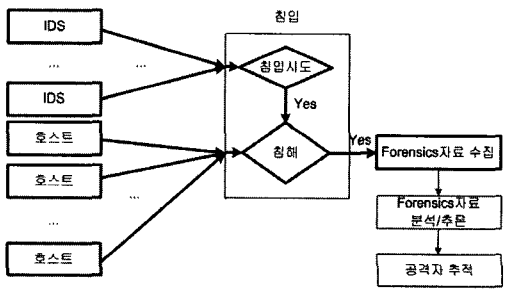


그림 3. 침해 판단과 Forensics 증거 자료 수집 방안에 대한 다이어그램. 굵은 선: 이 논문의 기술 대상

1. 분석 대상 및 절차에 관한 개괄적 기술

제안된 분석 방안은 연속된 공격의 sequence에 대한 IDS 및 호스트의 판단 정보(로그)를 상관관계 분석대상으로 한다. 그림 3과 같이 IDS의 로그와 여러 분석 대상 호스트의 설정 및 관련 로그에 기초하여, 침해 초기 단계를 파악하고 침입을 위한 daemon의 설치(infection)를 탐지한 뒤, 최종 공격으로 인한 시스템 침해 사실을 판단한다. 이 때 분석 대상 호스트의 침해에 대한 위험도(severity)를 계산하여 그림 2의 각 state 정보에 저장한다. 이 값에 기초하여 통합 분석 서버에서는 각 단계별 침해 발생 시점에서 누가/무엇을/어떻게/왜/어디서/언제 했는지에 대한 정보를 모니터하고 각 상태별 Forensics 정보[8]를 수집할 수 있는 기준을 제공한다.

2. 공격 단계별 침해 severity 분석 및 증거 자료 수집 알고리듬

먼저, 분석에 사용되는 parameter를 정의한다.

- (침해) severity: 호스트가 실제 침해되었을 가능성
- state(s): 연속된 공격에 대해, 분석 대상 호스트의 severity값을 기초로 한 대상 호스트의 침해 단계 s. (s:1~5 or HS:1, PS:2, VE(Vulnerability Exploit의 약자):3, D(Dameon의 약자):4, DDoS:5)
- 하위 state(ss): state s에서 severity계산을 위한 분석 자료가 동일한 침해 증거를 가리키는 경우 분석 자료의 중요도에 따라 구분된 하위 state. 중요도는 분석에 사용되는 서로 다른 자료가 동일한 침해 가능성에 대해 지시하는 경우 자료의 신뢰도에 따라 정의됨.
- v(s, IP): state s에서의 침해된 IP의 severity
- sv(s, ss): state s에서의 ss번째 하위 state에서의 sub-severity
- pre-condition: 호스트 침해가 성공하기 위한 조건
- post-condition: 호스트 침해에 관한 보고
- $c_k(s)$: 침해 성공가능성과 관련된 state s에서의 조건, (pre-condition, post-condition), (i=1, 2,...)
- $c_m(s, ss)$: 침해 성공가능성과 관련된 state s의 하위 state ss에서의 m번째 조건,(pre-condition, post-condition), (ss, m=1,2,...). state s의 post-condition을 구성하는 분석 자료가 동일한 침해 증거를 지시하는 경우 중요도에 따라 부분집합(sub-severity계산을 위한 조건)으로 구성됨.
- C(s): state s에서의 침해 판단을 위한 조건 집합, $c_k(s)$ 혹은 $c_m(s, ss)$ 로 구성.
- |C(s)|: C(s)의 element 수
- wt(s): state s 에서의 severity계산에 사용되는 조건 weighting factor
- wt(s, k or ss): state s에서의 $c_k(s)$ 혹은 sub state의 네트워크 관점 혹은 호스트 관점에서의 중요도에 따라 주어지는 조건 weighting factor (k: 1~|C(s)|, ss: 1~하위 state ss의 수)
- wt(s, ss, m): state s에서의 m개의 조건에 의해 주어지는 ss번째 하위 state에서의 sub-severity 계산을 위한 weighting factor
- Lo_N: 상태별 분석 대상 로그 파일의 수
- Li_N: 분석 대상 로그파일의 라인 수
- state s 에서 특정 분석 대상 호스트의 total severity = 분석 시점을 기준으로 계산되어진 특정 분석대상 호스트의 severity값의 합.
- S(s)={분석대상로그, 침해된 호스트 IP, 침해된 호스트 port, attacker IP, attacker port, t(s), v(s, 침해된 호스트 IP), state s에서 침해된 호스트의 severity, state s에서 침해된 호스트의 total severity, collection}: 공격 단계 s에서 침해된 호스트의 state value
- 다음의 절차를 따라, 침해 사실을 판단하고 관련 정보를 수집한다. 이때, time stamp를 가지는 로그의 time 정보에 대해 신뢰한다고 가정한다.

Step1. (전처리 단계)

주기적 시간 간격을 두고 단일 호스트로부터 수집된 로그를 그림 2의 각 상태에서 사용되는 정보로 분류/분석(C(s)의 post-condition) 후, 각각의 상태 정보와 관련된 정보를 저장한다.

Step2. (침해 severity 계산)

수집된 로그에 대해 시간적 우선순위와 시스템 상태 정보에 기초하여 관련 정보를 분석한다. 이때, 그림 4의 알고리듬을 이용하여 그림 2에 기술된 각 공격 상태별(s)로 공격에 대한 분석 대상 호스트(IP)의 severity(v(s, IP))를 계산한다. 이는 침해 시점에

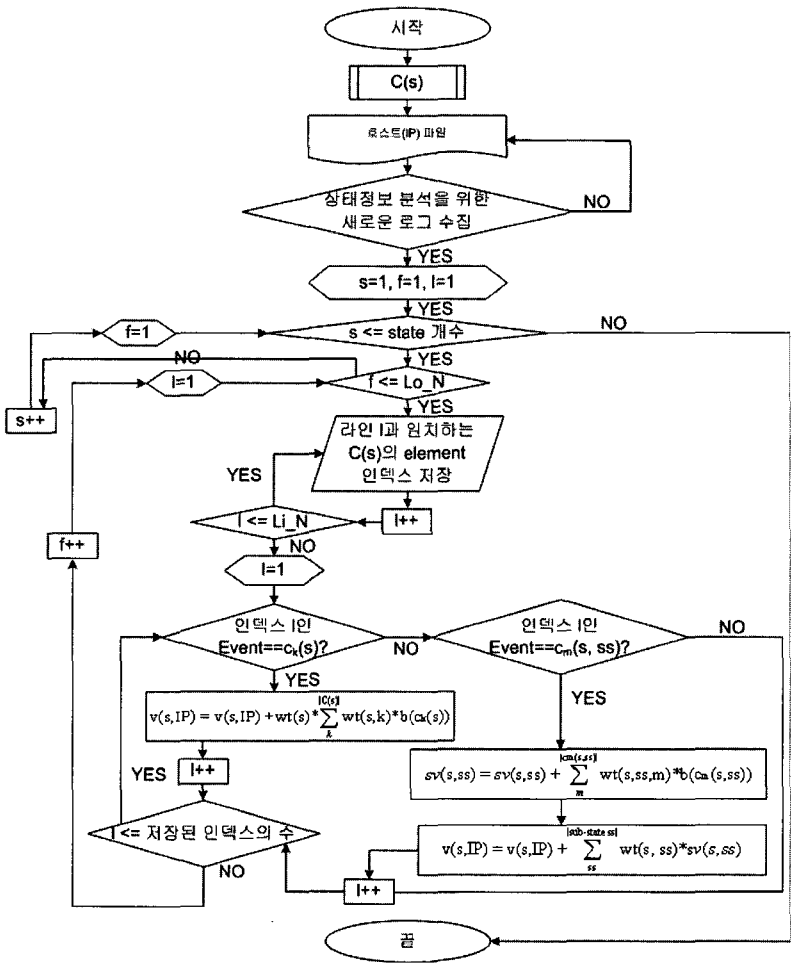


그림 4. step 1과 step 2를 통한 단계별 severity계산을 위한 알고리듬 flow chart, O(Li_N) (Lo_N〈〈Li_N && 인덱스 수 〈〈Li_N)

서 시스템이 침해되기 위한 조건(database에 index로 저장되어짐)과 실제 분석 대상 호스트로부터 발생된 침해 분석 자료를 비교 분석하고 실제 침해 상황을 판단하기 위함이다. 구체적인 적용 방안은 다음과 같다. 이 때 사용되는 조건변수(2진 변수(0 or 1)로 표현됨) b()를 구성하는 조건은 C(s)에 의해서 주어지고, C(s)는 각 state별로 다른 값을 가진다. 또한, $\sum_{k}^{|C(s)|} wt(s, k)=1$, b(조건)=1(조건이 참) or 0(조건이 거짓)의 값을 가진다. wt(s)는 state 중요도에 따라 변동 가능한 값으로 s의 값이 커짐에 따라, 시스템 침해 정도가 마지막 공격 단계에 근접한다고 볼 수 있으므로 상대적으로 큰 값을 가진다. 이 때 최종 공격 단계에서의 wt(최종 공격)는 최종 공격이 성공하기 이전 단계들의 합과 동일한 weighting을 가진다고 생각할 수 있으므로, 이전 state에서의 wt(s) 값의 합과 같은 값을 가진다. 표 2에서 각 단계별 위험도 계산에 사용되어지는 수식을 정리하였다. 이 때, 각 단계별 위험도 계산에 사용되어지는 파라미터 값은 다음과 같이 주어진다.

〈침해 초기 단계 - 분석 대상 호스트의 취약성 정보 수집〉

2-1단계. Host scanning 감지 단계: 수식 (1), where wt(HS)=1, C(1)={$c_1(1)$:(존재하는 IP, 스캔되어진 IP), $c_2(1)$: (존재하는 subnet IP, 스캔되어진 subnet IP),...}

2-2단계. Port scanning 감지 단계: 수식 (2), where wt(PS)=2, C(2)={$c_1(2)$:(사전에 스캔되어진 open port정보, iplog[10]로부터 수집된 scan되어 진 포트

표 2. 다단계 공격 시나리오에 기반 한, Step 2에서 사용되어진 각 공격 단계별 침해 위험도 분석을 위한 계산식.

< 침해 초기 단계 - 분석 대상 호스트의 취약성 정보 수집 >
$v(HS,IP)=wt(HS)*\sum_{k}^{\lvert C(1)\rvert} wt(HS,k)*b(c_k(HS))$ (1)
$v(PS,IP)=wt(PS)*\sum_{k}^{\lvert C(2)\rvert} wt(PS,k)*b(c_k(PS))$ (2)
$v(VE, IP) = wt(VE)*\sum_{k}^{\lvert C(3)\rvert} wt(VE,k)*b(c_k(VE))$ (3)
< 네트워크 침해 경로 설정 탐지 단계 >
$v(D,IP)=wt(D,1)*sv(D,1)+wt(D,2)*sv(D,2)+wt(D,3)*sv(D,3)+wt(D,4)*sv(D,4)$ (4)
where, $sv(D,1)=wt(D,1,1)*!b(c_1(D,1))+wt(D,1,2)*!b(c_2(D,1))+\sum_{m=3}^{\lvert c_m(D,1)\rvert} wt(D,1,m)*b(c_m(D,1))$ (5)
$sv(D,2)=wt(D,2,1)*!b(c_1(D,2))+wt(D,2,2)*!b(c_2(D,2))+\sum_{m=3}^{\lvert c_m(D,2)\rvert} wt(D,2,m)*b(c_m(D,2))$ (6)
$sv(D,3)=wt(D,3,1)*b(c_1(D,3))+\sum_{m=2}^{\lvert c_m(D,3)\rvert} wt(D,3,m)*b(c_m(D,3))$ (7)
$sv(D,4)=wt(D,4,1)*b(c_1(D,4))+\sum_{m=2}^{\lvert c_m(D,4)\rvert} wt(D,4,m)*b(c_m(D,4))$ (8)
< 최종 공격 감지 단계 >
$v(DDoS,IP)=wt(DDoS)*\sum_{k}^{\lvert C(5)\rvert} wt(DDoS,k)*b(c_k(DDoS))$ (9)

정보), $c_2(2)$:(사전에 스캔되어진 open port정보, SNORT로그에 남겨진 scan 포트 정보), ...}

2-3단계. Vulnerability Exploit 감지 단계: 수식 (3), where wt(VE)=3, wt(VE,k)는 VE의 위험도(high, medium, info)에 따라 다르게 주어짐. 3rd (state) pre-condition=대상 호스트의 사전에 수집되어진 취약점 정보, 3rd (state) post-condition=분석 대상 호스트에 대해 SNORT에서 탐지된 로그, C(3)={$c_1(3)$: (3rd pre-condition (high), 3rd post-condition (high)), $c_2(3)$: (3rd pre-condition (medium), 3rd post- condition (medium)), $c_3(3)$: (3rd pre-condition (info), 3rd post- condition (info)), ...}

〈네트워크 침해 경로 설정 탐지 단계〉

2-4단계. Infection 감지 단계: 수식 (4), where !: not, wt(D,1)=3, wt(D,2)=3, wt(D,3)=2, wt(D,4)=2, ss:1~4, C(4)={$c_1(4,1)$: (사전에 scan된 파일 정보, 주기 T마다 scan된 파일 정보), $c_2(4,1)$:(사전에 scan된 파일의 소유권한에 대한 정보, 주기 T마다 scan된 파일의 소유권한에 대한 정보), $c_1(4,2)$:(사전에 scan된 활성화된 포트정보, 주기 T마다 scan된 활성화된 포트정보), $c_2(4,2)$:(사전에 scan된 활성화된 프로세스, 주기 T마다 수집된 scan된 활성화된 프로세스), $c_3(4,2)$:(실행가능 파일의 문자열 정보, 실행중인 daemon에서 발견되는 공격 daemon(DDoS daemon)의 문자열), $c_1(4,3)$:(공격 툴 별 daemon실행 시 주고

받는 특정 문자열, IDS에서 탐지 문자열, 예: *HELLO* message), $c_1(4,4)$:(chkrootkit[7]의 infect file DB, chkrootkit에서 infected file 탐지), ...}

〈최종 공격 감지 단계〉

2-5단계. DDoS 감지 단계: 수식 (9), where $wt(DDoS)=wt(HS)+wt(PS)+wt(VE)+wt(D,1)+wt(D,2)+wt(D,3)+wt(D,4)$, $C(5)=\{c_1(5)$:(nothing, SNORT 탐지 로그 존재), $c_2(5)$:(호스트에 따라 고정된 DDoS daemon의 포트 번호에 대한 rand()함수 값, 호스트의 scan된 포트 번호), ...}

이 때 각 단계별 severity계산에 사용되는 b(조건)의 값을 1로 만드는 관련 로그는 누가(공격자 IP 주소), 어떻게(port/취약점/daemon설치), 왜(취약점, 시스템 설정), 언제(time stamp), 무엇(단계정보), 어디서(피해자 IP주소)의 형식(:S(s))으로 침입 사실을 알리기 위해 통합 분석 서버에 저장된다. 이 때 시간적 차이를 두고 동일한 조건이 만족되는 경우, 시간적 우선순위가 높은 로그를 저장하고 이전 분석 자료는 저장하지 않는다. 또한 b(조건)을 1로 하는 새로운 조건이 존재하는 경우는 이전 severity값에 새로운 severity값을 더해서 저장한다.

Step3. (침해 단계 별 증거자료 수집)

Step2에서 계산되어진 severity값을 기준으로 분석 대상 호스트의 침해 상황별로 각 단계의 증거 수집 대상 호스트와 수집되어야 하는 증거자료를 정의한다.

Case 1. 특정 호스트의 침해 상황: 특정 호스트의 state s에서 계산 되어진 severity 값이 min.th.(s)(=특정 호스트에 대한 state s에서의 C(s)의 element별 weighting factor의 최소 값*wt(s)) 값과 avg.th.(s)(=특정 호스트에 대한 state s에서의 element별 weighting factor의 평균 값 * wt(s))값 사이에 위치하는 경우, 특정 호스트의 state s에서의 증거 자료를 수집한다.

Case 2. 특정 호스트 및 주변 감시 대상 호스트의 침해 상황: 특정 호스트의 state s에서 계산 되어진 severity 값이 avg.th.(s)값보다 크거나(i.e. C(s)의 element가 2개 이상 만족되거나 만족되어진 하나의 element의 weighting factor가 avg.th.(s) 값보다 큰 경우), state s까지 severity값의 합이 sum.min.th.(s)(=특정 호스트의 모든 state s에 대한 min.th.(s)의 합)값과 sum.avg.th.(s)(=특정 호스트의 모든 state s에 대한 avg.th.(s)의 합)값 사이에 위치하는 경우 다른 감시 대상 호스트의 해당 상태에 대한 증거 자료와 보호 대상 호스트의 최종 공격 단계에 대한 증거 자료를 수집한다.

Case 3. 감시 대상 호스트들이 침해당하고 보호 대상 호스트가 공격 받았을 가능성이 높은 상황: 특정 호스트의 state s까지 severity값의 합이 sum.avg.th.(s)값보다 크거나 분석 대상 호스트들의 state s까지의 severity합이 total.avg.th.(=대상 호스트들의 sum.avg.th.(s)의 합)값보다 큰 경우 주변 분석 대상 호스트들의 state s까지의 Forensics정보 및 보호 대상 호스트의 최종 공격 단계에 대한 Forensics정보를 수집한다. 이때, 각 침해 단계별 수집되는 증거자료는 다음과 같다.

- Host scanning 감지 단계: Nothing.
- Port scanning 감지 단계: 수집 시작 시 date와 time, 열린 TCP/UDP ports, 수집 종료 시 date와 time
- Vulnerability Exploit 감지 단계: 수집 시작 시 date와 time, 활성화된 프로세스의 목록, 시스템에 대한 유용한 정보(CPU정보, 운영체제의 버전, 시스템의 가동시간, 호스트 이름/도메인 이름, 모든 스왑 파티션들, 모든 파일시스템들, 마운트 된 파일 시스템들), 운영체제의 커널 메모리에 적재되어 있는 모듈들의 목록, arp와 routing table로부터 수집된 cache tables, 수집 종료 시 date와 time
- Infection 감지 단계: 수집 시작 시 date와 time, 물리적 memory image, 운영체제의 커널 메모리에 적재되어 있는 모듈들의 목록, 활성화된 혹은 미심쩍은 프로세스의 목록, 시스템에 대한 유용한 정보, arp 및 routing table로부터 수집된 Cache tables, 수집 종료 시 date와 time
- 최종 공격(DDoS) 감지 단계: 수집 시작 시 date와 time, arp 및 routing table로부터 수집된 Cache tables, 수집 종료 시 date와 time

Ⅳ. Experiment

이 장에서는 제안되어진 방안의 성능 분석결과를 기술하였다. 이를 위해, 외부 망(Internet)과 내부 망 사이에 그림 5-(a)와 같은 허니팟을 구성하고, Trin00 Daemon을 이용한 DDoS공격 상황을 가

정하였다. 각 호스트에서 발생되어지는 정보의 분석을 위한 서버를, 침해위험도 관리 서버라 칭함, 허니팟 내 동일 스위치를 통해 연결하였다. 각각의 호스트(클라이언트)로부터 수신되어지는 정보를 분석하기 위해, 서버에서 그림 5와 같은 침해위험도 관리 agent인 Severity Management-Server(SMS)를 동작시키고, 다음의 시나리오를 기반으로 성능을 테스트하였다.

1. 시스템 구성 및 공격 시나리오

실험에 사용되어진 서버는 CPU 2.7GHz, DDRRAM 1GB를 사용하여 구성하였으며, 동작상황에 대한 시각적 관찰을 위하여, windows상에서 동작하였다. 각각의 클라이언트로 사용되어진 호스트는 CPU 펜티엄3 700MHz, SDRAM 256MB를 사용하여 구성하였으며, 취약점 노출을 위하여 Linux RedHat 7.0상에서 Apache, Php4, Mysql과 함께 Zeroboard 4.1pl5을 운영하였다. 이를 바탕으로, 정형적인 DDoS공격 시나리오 (공격자→Master→Daemon→DDoS공격)에 기초하여 테스트를 진행하였으며, 상세한 공격 시나리오는 다음과 같다.

시나리오 1단계. 공격자가 감시 대상 호스트를 이용, 서버 팜에 대한 DDoS공격을 하기 위해 감시대상 호스트(A, B, C)를 스캔: Nmap을 이용한 host liveness 및 host OS정보 탐지 및 상 호스트의 open port에 대한 정보를 수집. (host scan & port scan)

시나리오 2단계. 공격자가 스캔한 결과를 바탕으로 감시대상 호스트에 툴(DDOS Trin00 Daemon)을 설치 (A:마스터, B, C:데몬): Nessus[9]를 이용한, 이용가능 취약점에 대한 정보 수집 및 툴 DDoS툴 설치. (취약점 정보 탐지, infection)

시나리오 3단계. 호스트 A ,B ,C를 통해 서버 팜(S)을 DDoS 공격

이러한 공격 시나리오를 통해 발생되어진 관련 로그를 수집하고 다음과 같이 이에 대한 분석을 수행하였다. 이때, 각 state별 기술되어진 C(s)정보는 앞장에서 기술되어진 정보와 일치하며, 각 로그의 time stamp에 대한 신뢰도 검증은 이미 이루어 졌다고 가정한다.

분석 1단계. 수식 (1)과 (2)를 통한, 스캐닝 감지를 위한 severity 계산.

(Host scan 관련 분석 로그)

- SNORT 로그

  (시간정보) Auth.Alert IDS snort: [1:402:7] ICMP Destination Unreachable Port Unreachable [Classification: Misc activity] [Priority: 3]: {ICMP} 호스트 A → 공격자

(Port scan 관련 분석 로그)

- SNORT 로그

  ...

  (시간정보) Auth.Alert IDSsnort: [122:3:0] (portscan) TCP Portsweep {PROTO255} 공격자 → 호스트 A

  ...

- iplog[10] 로그

  ...

  (시간정보) TCP: **ssh** connection attempt from 공격자:43116

  ...

- 사전에 수집된 호스트 A의 open port정보

  PORT STATE SERVICE

  ...

  22/tcp open ssh

  111/tcp open rpcbind

  139/tcp open netbios_

  ...

공격자로부터 호스트 A로의 host scanning 및 port scanning 움직임(침입 시도)이 SNORT를 통해 감지되었으며, 실제 호스트 A에서 관련 포트가 open된 것이 발견(실제 침해 발생 가능성)되었다. wt(HS)=1, wt(PS)=2, wt(HS, 1)=0.4, wt(PS, 1)=0.3로 주어진 경우로, v(HS, 호스트 A)=1*(0.4*1+$\sum_{k=2}^{|C(1)|} wt(HS,k)*0$)=0.4 및 v(PS, 호스트 A)=2*(0.3*1+$\sum_{k=2}^{|C(2)|} wt(PS,k)*0$)=0.6로 주어진다. 이 때, host scan 및 port scan에 대하여 주어진 값은, 각각 min.th(HS or PS)(=0.1 or 0.2)값과 avg.th.(HS or PS)(=0.5 or 1)값의 사이에 위치하게 된다. 호스트 B, C에 대해서도 약간의 시차를 두고, 유사한 분석 결과 값을 얻을 수 있었다.

분석 2단계. 수식 (3)과 (4)를 통한, Vulnera-

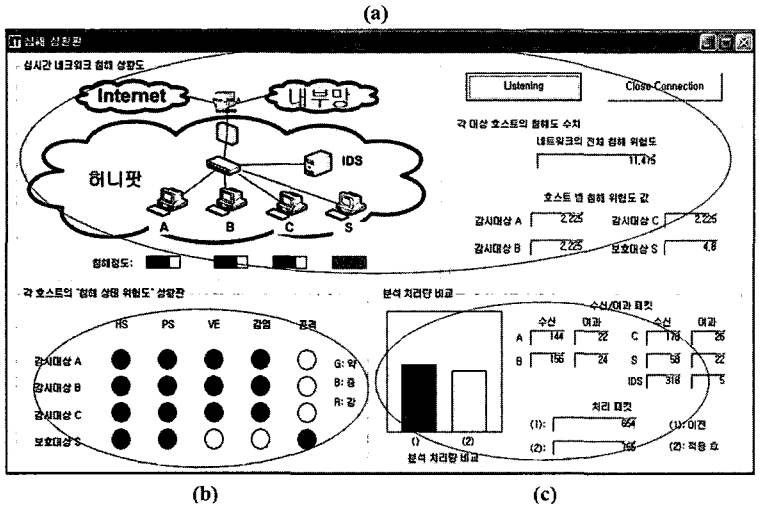


그림 5. Prototype 소프트웨어 구성도 및 동작 화면(서버): (Severity Management(SM)-Server, SMS)

bility Exploit 감지 및 Infection 감지를 위한 severity 계산.

(Vulnerability exploit 관련 분석 로그)

- SNORT 로그

  ...

  (시간정보) Auth.Alert IDS snort: [1:3018:1] NETBIOS SMB NT Trans NT CREATE oversized Security Descriptor attempt [Classification: Generic Protocol Command Decode] [Priority: 3]: {TCP} 공격자:2612 → 호스트 A:139

  ...

(Infection 관련 분석 로그)

- SNORT 로그

  ...

  (시간정보) Auth.Alert IDSsnort:[1:232:5] DDOS Trin00 Daemon to Master *HELLO* message detected [Classification: Attempted Denial of Service] [Priority: 2]: {UDP} 공격자:1995 → 호스트 A:31335

  ...

공격자로부터 호스트 A에 대한 Vulnerability Exploit 시도(침입 시도)가 IDS에서 탐지되었다. 또한, 사전에 취약점 스캐너인 Nessus를 이용하여 호스트 A에 대한 취약점을 탐색한 결과, info:40, medium: 14, high: 18개의 vulnerability가 탐지 되었으며, 이 중 139 포트에 대한 취약점(실제 침해 발생 가능성)은 info에 해당하였으므로, vulnerability exploit에 대한 탐지 결과를 바탕으로 다음과 같은 severity 값을 계산하였다. wt(VE)=3, wt(VE, 2)=18/72로 주어진 경우로, v(VE, 호스트 A) $= 3*(wt(VE,1)*0+(18/72)*1+\sum_{k=3}^{|C(3)|} wt(VE,k)*b(ck))=3*((18/72)*1+0)=3/4$로 주어진다. 이 때, min.th.(VE)(=0.3)과 avg.th.(VE)(=1.5)값의 사이에 위치하게 된다. 또한 실제 Infection과 관련해서, 호스트 A에서 이전에 활성화된 프로세서와 다른 프로세서가 발견되고, 그 프로세스에서 실행중인 daemon의 strings에 DDoS tool에서 사용하는 daemon에서 발견되는 문자열(예: PONG)이 존재(실제 침해 발생 가능성)하였으며, 다음과 같은 공격 툴 별 daemon실행 시 주고받는 특정 문자열이 IDS에서 탐지(침입 시도)되었다. 따라서, wt(D,2,3)=0.3, wt(D,3,1)=0.5인 경우, sv(D,2)=wt(D,2,1)*0+wt(D,2,2)*0+ 0.3*1=0.3와 sv(D,3)=1*1=0.5로부터, v(D,IP)=3*0+3*0.3+2*1+2*0=2.9로 주어진다. 이 때, min.th.(VE)(=0.3)와 avg.th.(VE) (=5)값의 사이에 위치하게 된다. 호스트 B, C에 대해서도 약간의 시

차를 두고, 유사한 분석 결과 값을 얻을 수 있었다.

분석 3단계. 수식 (9)를 통한, DDoS 감지를 위한 severity 계산.

(DDoS 관련 분석 로그)

- SNORT 로그

...

(시간 정보) Auth.Alert IDSsnort: [1:2339:2] TFTP NULL command attempt [Classification: Potentially Bad Traffic] [Priority: 2]: {UDP} 호스트 B:2029 → 호스트 S:69

...

SNORT 로그로부터, 호스트 B에서 호스트 S로의 DDoS공격이 감지되었다(침입 시도 이면서, 동시에 실제 침해 증거). wt(DDoS, 1)=0.7이고 wt(DDoS, 2)=0.3으로 주어진 경우, v(DDoS, 호스트 S)=16*(0.7*1+0)=11.2로 주어지는데, 이는 avg.th.(DDoS)(=8) 값보다 큰 값을 가지게 된다. 또한, 이 때 호스트 A, C로 부터 호스트 S에 대한 호스트 B와 같은 공격이 이루어져서, 결국 state DDoS까지의 severity합이 total. avg.th.(=16)값보다 큰 경우가 발생하게 되고 이는, 실제 여러 단계를 거쳐 최종 공격이 이루어졌음을 나타낸다.

이 실험 과정에서, signature 기반의 IDS인 SNORT를 통해 탐지되어지는 침입 시도 로그는 exact pattern matching에 기반 한 알고리즘을 사용하였다. 실제 오탐 발생 가능성이 높은 짧은 길이의 패턴을 가지는 signature 문자열에 대해서는 낮은 weighting값을 설정함으로써, 오탐을 최소화 하였다. 또한, IDS에서 탐지되지 못한 packet의 경우, 시스템에서의 발생로그 정보를 기반으로 위험도 값을 산출함으로서 실제 시스템에 대한 침해 사실이 있는 경우의 침해 사실 판단 누락을 최소화할 수 있었다. 호스트 침해 분석과정에서 사용자의 정상적 행위를 통해서도 발생가능성이 높은 로그 파일의 정보(예: netstat)에 대해서는 낮은 weighting 값을 설정함으로써, 위험도 분석과정에서의 잘못된 침해 사실 판단을 최소화할 수 있었다.

2. 성능 분석 결과

그림 5는 각 분석(A, B, C) 및 보호(S) 대상 호스트의 실시간 침해 상황 분석 결과를 보여준다.

그림 5-(a)의 좌측 화면의 경우는 대상 호스트의 네트워크 구성도와 더불어 침해 정도를 나타내고 있다, where 적색: 최종 침해 발생, 청색: 침해 발생 조짐에 따른 침해 위험도 높음, 녹색: 초기 침해 발생, 흰색: 침해 가능성 없음. 또한, 그림 5-(a)의 우측 화면의 경우는 각 호스트 별 침해 위험도 값을 나타낸다. 즉, 실시간으로 변화하는 '침해정도' 그래프와 관련 데이터 값을 통해, 각 호스트의 실제 침해 상황 정도를 보여준다. 그림 5-(b)의 경우, 단일 호스트에 대해, 그림 2에서 기술되어진 각 상태별 침해 정도에 따라 적, 청, 녹색으로 위험도를 표시한다. 그림 5-(c)의 경우는, 각 분석 대상 호스트로부터 전송되어지는 분석 정보(Snort로그, 각 호스트의 취약점 정보, process정보, 스캐닝 정보, ...)의 량(패킷 수)을 보여준다(1: 기존, 2: 제안). 그림 5-(a)와 (b)에 보여 진 바와 같이, DDoS공격의 진행과 동시에 감시 대상 호스트(A, B, C)의 침해(흰색→녹색→적색) 사실과 최종 공격이 보호 대상 호스트(S)에 가해진 것(적색)을 확인할 수 있다. 이때, 모의 시나리오 동작과정에서 잘못된 침해 경고 발생가능성이 높은 로그에 대해서는 실제 침해 위험도 값이 낮게 측정됨으로써 잘못된 침해 증거 자료 수집이 되는 경우가 발생하지 않았다. 또한, 실제 특정 포트에 대한 침입 시도가 발생했지만 해당 호스트의 open 포트 정보와 일치하지 않는 경우, 실제 침해 가능성은 0이므로 침해 위험도 값은 0으로 계산되었다. 이를 통해서도, 잘못된 침해 판단으로 인한 증거 자료 수집 노력을 최소화할 수 있었다. 이에 기초하여, 침해 시점에 기초하여 각 호스트의 침해 위험도 값에 따라, 침해 상태별 Forensics을 위한 증거자료를 수집할 수 있었다. 또한, 각 호스트로부터 수집되어지는 침해 분석 자료의 수집 주기(T)와 관련해서, 그림 6, 7과 같은 성능 분석 결과를 얻었다. T=30sec인 경우, 기존 방안 적용 시, 전체 199213byte의 로그가 수집되었으며, 제안된 방안 적용 시 80719byte (평균: 1681byte/T)의 로그가 수집되었다. 또한, T=10sec인 경우, 제안된 방안 적용 시 전체 100624byte(평균: 708byte/T)의 로그가 수집(동일 로그 수집으로 인한 로그 수집 량 증가)되었다. 두 상황 모두에서, 동일한 침해 상황에 대한 분석결과를 얻을 수 있었으며, 각각 60%및 49.5%의 분석에 사용되어지는 로그 량 감소 효과를 가져 올 수 있었다. 즉, 제안된 방안 적용 시, T가 짧아짐에 따라

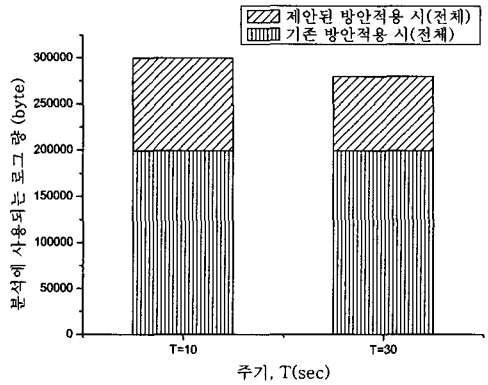


그림 6. Forensics 분석에 사용되어지는 수집 로그 량 비교. 제안된 방안 vs. 기존 방안.

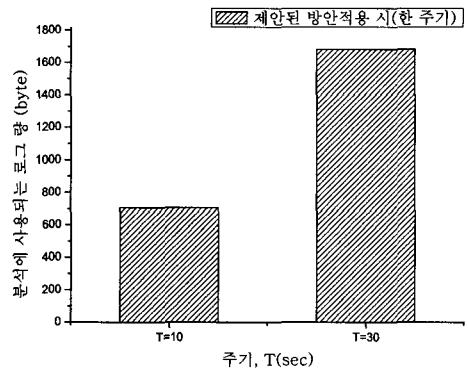


그림 7. 제안된 방안 적용 시 한 주기 당, Forensics 분석에 사용되어지는 평균 수집 로그 량 비교.

분석에 사용되어지는 평균 로그 량을 감소시킬 수 있었다. 그러나 이는, 잦은 시스템 자원 사용으로 인한 각 호스트 및 침해위험도 관리 서버의 평균 부하 증가(최고 부하 감소)를 야기 할 수 있으므로, 반복적 작업을 통해 망 크기 및 시스템 사양 등을 고려하여 최적화시켜야 한다. 모의 시나리오를 통한 검증과정에서, 실제 관리 서버로 로그를 전송하는 호스트의 수가 많지 않아 서버에서의 CPU 및 Memory를 비롯한 자원의 사용량 증가로 인한 시스템 성능변화는 거의 찾아 볼 수 없었다.

V. 결 론

이 논문에서는 다단계 공격에 대해 호스트의 실제 침해당한 시점에 기초하여 침해 상황을 감지하고 관련 증거 자료를 수집하기 위한 방안을 제안하였다. 이는 침해 시점에 기초하여 자료를 분석하고 증거 자료를 수집함으로써 증거자료의 손상을 최소화한다. 즉, 침해 판단을 위해서 호스트의 구성 설정 및 취약점, 시스템 로그, IDS 로그를 이용하여 다단계 공격에 대한 분석 대상 호스트의 침해 severity값을 계산하였다. 이에 기초하여, 분석 대상 호스트의 침해 초기 단계, 침해 경로 설정 단계 및 실제 공격 단계에 이르기 까지 분석 대상 호스트의 각 침해 단계에 따른 관련 증거 자료를 침해 시점에 근거하여 수집한다. 그러나 침해 발생 시점과 침해 판단 후 자료 수집 시간 사이에 공격자에 의한 증거자료 훼손가능성은 존재한다. 현재 이를 보완한 방안에 대해 연구를 진행 중이다.

참 고 문 헌

[1] "월간 정보보호 뉴스", 한국정보보호진흥원 정기간행물, 10, 2005.

[2] Snort *v*2.0, an open source network intrusion detection system, http://www.snort.org

[3] P. A. Porras and P. G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances", National Information Systems Security Conference, 1997.

[4] HervéDebar, Andreas Wespi, "Aggregation and Correlation of Intrusion Detection Alerts", in proceedings of RAID 2001.

[5] Benjamin Morin and al., "M2D2: a formal data model for IDS Alert Correlation", Proceedings of RAID 2002, Zurich, Switzerland, October 2002.

[6] Frederic Cuppens, Alexandre Miege, "Alert Correlation in a Cooperative Intrusion Detection Framework", in proceedings of IEEE S&P, 2002.

[7] Rootkit identifier, http://www.chkrootkit.org

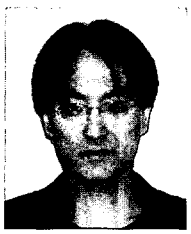
[8] Mariusz Burdach, "Forensic Analysis of a Live Linux System, Pt. 1,2", http://www.securityfocus.com

[9] Nessus 2.2.8, the network vulnerability scanner, http://www.nessus.org

[10] iplog 2.2.3, a TCP/IP traffic logger, http://www.freshports.org/net/iplog

〈著者紹介〉

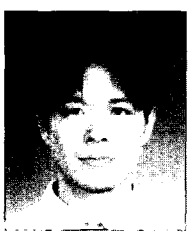
최 윤 호 (Yoon-Ho Choi) 정회원
2002년 8월: 경북대학교 전자전기공학부 졸업
2004년 8월: 서울대학교 전기컴퓨터공학부 석사
2004년 9월 ~ 현재: 서울대학교 전기컴퓨터공학부 박사과정
〈관심분야〉 침입 탐지, 네트워크 포렌식, economics of information security



박 종 호 (Jong-Ho Park) 준회원
2003년 2월: 서울대학교 전기공학부 졸업
2006년 2월: 서울대학교 전기컴퓨터공학부 석사
2006년 3월 ~ 현재: 서울대학교 전기컴퓨터공학부 박사과정
〈관심분야〉 정보보호, 네트워크 포렌식, 무선네트워크

김 상 곤 (Sang-Kon Kim) 정회원
1991년 2월: 홍익대학교 전기공학과 졸업
1993년 2월: 홍익대학교 전기공학과 석사
2003년 9월 ~ 현재: 서울대학교 전기컴퓨터공학부 박사과정
〈관심분야〉 정보보호, 네트워크 포렌식, 통신공학

서 승 우 (Seung-Woo Seo) 정회원
1987년 2월: 서울대학교 전기공학 졸업
1989년 2월: 서울대학교 전기공학 석사
1993년 12월: 미국펜실베니아주립대학 전기공학 박사
1993년 ~ 1994년: 미국펜실베니아 주립대학 전산기공학과 조교수
1994년 ~ 1996년: 미국 프린스톤대학 전기공학 poem연구소
〈관심분야〉 네트워크 보안, 센서 네트워크 보안, economics of information security



강 유 (Yu Kang) 준회원
2003년: 서울대학교 컴퓨터공학과 졸업
2004년 ~ 현재 : KT 정보보호본부
〈관심분야〉 웹 보안, 모의해킹, 위험분석, 컴퓨터 포렌식

최 진 기 (Jin-gi Choe) 준회원
1998년: 숭실대학교 전자공학과 졸업
1999년 ~ 2004년: KT 차세대통신망연구소 보안기술연구실
2005년: 충남대학교 정보통신공학과 석사
2005년 ~ 현재: KT 정보보호본부
〈관심분야〉 위험분석, 네트워크 보안

문 호 건 (Ho-Kun Moon) 정회원
1985년: 숭실대학교 전자공학과 졸업
1987년: 중앙대학교 전자공학과 석사
2005년: 부산대학교 전자공학과 박사
1987년 ~ 2004년 : KT 차세대 통신망연구소 보안기술연구실장
2005년 ~ 현재 : KT 정보보호본부 기술개발 1부장
〈관심분야〉 위험분석, 위험관리, 네트워크 보안

이 명 수 (Myung-Soo Rhee) 준회원
1989년 연세대학교 대학원 전자공학과 박사과정 졸업
1990년 ~ 2004년 : KT 네트워크 보안연구팀장
2004년 ~ 2005년 : 정보보호학회 무임소이사
2005년 ~ 2006년 : KT 정보보호기술담당
관심분야 : e-business, 플랫폼 buiness, 네트워크 컴퓨팅, 프라이버시 보호