



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년05월08일
(11) 등록번호 10-1732972
(24) 등록일자 2017년04월27일

(51) 국제특허분류(Int. Cl.)
G06F 21/56 (2013.01) H04L 29/06 (2006.01)
(52) CPC특허분류
G06F 21/56 (2013.01)
H04L 63/14 (2013.01)
(21) 출원번호 10-2015-0168584
(22) 출원일자 2015년11월30일
심사청구일자 2015년11월30일
(56) 선행기술조사문헌
KR1020130006750 A
KR101421630 B1
KR1020120057059 A
JP2014063349 A

(73) 특허권자
인하대학교 산학협력단
인천광역시 남구 인하로 100, 인하대학교 (용현동)
(72) 발명자
김기창
인천광역시 연수구 먼우금로 126 (동춘동, 현대대림2차아파트) 현대대림2차아파트 207동 404
박철우
인천광역시 동구 송림3.5동 67번지
(뒷면에 계속)
(74) 대리인
이원희

전체 청구항 수 : 총 7 항

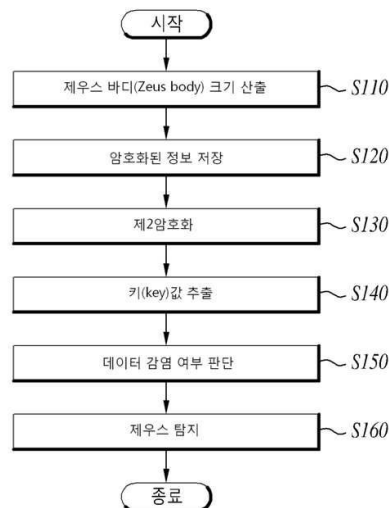
심사관 : 윤혜숙

(54) 발명의 명칭 씨앤씨 제우스의 윈도우 API후킹 및 패킷 길이에 기반한 탐지 방법

(57) 요약

C&C Zeus의 Windows API 후킹 및 패킷 길이에 기반한 탐지 방법이 개시된다. C&C Zeus의 Windows API 후킹 및 패킷 길이에 기반한 탐지 방법은, 패킷에서 제우스 바디(Zeus body) 크기를 산출하는 단계; 상기 제우스 바디 크기를 제1암호화 하여 특정 위치에 암호화된 정보를 저장하는 단계; 상기 제우스 바디 크기 및 암호화된 정보를 제2암호화 하는 단계; 상기 제2암호화를 통해 상기 제1암호화에 사용된 키(key)값을 추출하는 단계; 및 상기 추출된 키 값을 이용하여 암호화된 데이터의 감염여부를 판단하는 단계를 포함한다.

대표도 - 도2



(72) 발명자

손지웅

충청남도 천안시 서북구 봉정로 366 111동 1305호
(두정동, 한성3차필하우스아파트)

황현기

울산광역시 남구 문수로369번길 58 2동 408호 (옥
동, 도성아파트)

이 발명을 지원한 국가연구개발사업

과제고유번호 1345228650

부처명 교육부

연구관리전문기관 한국연구재단

연구사업명 일반연구자지원

연구과제명 대용량 DDoS 공격 진원지의 실시간 역추적 기술 개발

기 여 율 1/1

주관기관 인하대학교

연구기간 2014.05.01 ~ 2015.04.30

공지예외적용 : 있음

명세서

청구범위

청구항 1

사용자 명령어를 입력받는 입력부와 상기 입력부에서 입력받은 명령어에 따라 알고리즘을 수행하는 제어부 및 상기 제어부에서 산출된 데이터를 저장하는 저장부를 구비한 탐지 시스템에서 수행되는 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 방법에 있어서,

상기 제어부가 입력받은 헤더와 바디로 구성된 사용자 명령의 리포트 패킷에서 제우스 바디(Zeus body) 크기를 산출하는 단계;

상기 제어부가 상기 제우스 바디 크기를 제1암호화 하여 상기 저장부에 암호화된 정보를 저장하는 단계;

상기 제어부가 상기 제우스 바디 크기 및 상기 저장부에 저장된 암호화된 정보를 제2암호화 하여 상기 제1암호화에 사용된 키(key)값을 추출하는 단계; 및

상기 제어부가 상기 제2암호화의 과정으로 추출된 키 값과 상기 저장부에 저장된 제1암호화에 따른 암호화 정보에 포함된 키 값을 비교하여, 상기 제1암호화에서 암호화된 정보와 상기 제2암호화에서 암호화된 정보를 포함하는 데이터의 감염여부를 판단하는 단계;

를 포함하는 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 방법.

청구항 2

제 1 항에 있어서,

상기 데이터의 감염여부를 판단하는 단계 이후, 상기 탐지 시스템에서 상기 제우스에 감염되어 주소의 첫 바이트에 오퍼코드(opcode)가 변형된 에이피아이(API)를 추출하여 제우스를 탐지하는 단계를 더 포함하는 것을 특징으로 하는 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 방법.

청구항 3

제 1 항에 있어서,

상기 제1암호화 및 제2암호화는 XOR 암호화인 것을 특징으로 하는 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 방법.

청구항 4

제 1 항에 있어서,

상기 제1암호화 하여 상기 저장부에 암호화된 정보는,

상기 리포트 패킷에 포함된 암호문(Cipher Text)을 비주얼 디크립트(visualDecrypt)한 뒤 상기 리포트 패킷의 헤더에서 21 내지 24번째 바이트의 내용이 상기 제우스 바디 크기에 대응하는 것을 특징으로 하는 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 방법.

청구항 5

제 1 항에 있어서,

상기 제우스 바디 크기는 플레인 텍스트(Plain Text)인 것을 특징으로 하는 씨앤씨(C&C) 제우스(Zeus)의 윈도우

에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 방법.

청구항 6

제 1 항에 있어서,

상기 데이터의 감염여부를 판단하는 단계는,

상기 제1암호화에서 추출된 키와 상기 제2암호화에서 추출된 키가 동일한 경우, 제우스에 감염된 패킷으로 판단하는 것을 특징으로 하는 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 방법.

청구항 7

제 1 항 내지 제 6항 중 어느 한 항에 기재된 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

발명의 설명

기술 분야

[0001] 본 발명은 C&C 제우스(Zeus)의 윈도우 API 후킹 및 패킷 길이에 기반한 탐지 방법에 관한 것이다. 보다 상세하게, 본 발명의 일 실시예는 PC에 대하여 제우스(Zeus)의 감염여부를 판단하고, 외부로 송신되는 패킷에 대해 제우스(Zeus) 패킷인지를 탐지할 수 있는 C&C 제우스(Zeus)의 윈도우 API 후킹 및 패킷 길이에 기반한 탐지 방법에 관한 것이다.

배경 기술

[0002] 멀웨어(Malware: Malicious software)는 악성코드를 의미한다. 이러한 멀웨어 중 하나인 제우스(Zeus)는 전 세계적으로 감염률이 높은 멀웨어 중 하나이다. 일반적으로 인터넷 상에서 특정 바이너리 파일을 다운 받아 실행함으로써 감염이 되며, 감염이 되면 현재 실행 중인 프로세스들의 특정 윈도우 소켓 API(Application Programming Interface)에 후킹을 시도하여, 해당 프로세스가 API를 실행하면 제우스(Zeus)의 특정 코드를 실행하도록 변조가 되어 ID나 비밀번호, 계좌나 카드 관련 내용 등 개인 정보들이 유출되도록 구성되어 있다.

[0003] 한편, 2006년 처음 발견된 제우스(Zeus)는 최초에는 C&C(Command and Control server) 방식으로 통신을 하였으며, 현재는 많은 변종들이 생겨나 P2P(Peer to Peer) 방식으로 통신을 하는 변종도 생겨났으며, 안드로이드 등 모바일을 위한 변종도 생겨나기 시작하였다.

[0004] 또한, 제우스(Zeus)의 감염여부는 제우스(Zeus) 자체가 암호화된 패킷을 사용하고 제우스(Zeus) 자체의 실행파일을 감염 후 지우는 등 여러가지 탐지 회피 기술을 사용하기 때문에 탐지가 용이하지 않다.

[0005] 따라서, 제우스(Zeus)의 기초 버전인 C&C(Command and Control)방식을 중심으로 제우스(Zeus)가 어떤 방식으로 Windows API를 후킹하는지, 그리고 어떤 방법으로 후킹이 되었는지를 살펴보고 동시에 후킹한 정보를 어떻게 암호화하여 송출하는 지를 분석함으로써 제우스(Zeus) 감염여부 및 제우스(Zeus) 패킷을 탐지하는 기술이 요구된다.

발명의 내용

해결하려는 과제

- [0006] 본 발명이 이루고자 하는 기술적 과제는 C&C 제우스의 윈도우 API후킹 및 패킷 길이에 기반한 탐지 방법을 제공하는데 있다.
- [0007] 본 발명이 이루고자 하는 다른 기술적 과제는 암호화된 데이터의 감염여부를 판단하도록 패킷의 사이즈를 산출하고, 상기 사이즈 및 위치를 암호화 하여 저장하고, 제우스(Zeus)에 의한 감염 여부를 판단하는 C&C 제우스의 윈도우 API후킹 및 패킷 길이에 기반한 탐지 방법을 제공하는데 있다.
- [0008] 본 발명이 이루고자 하는 또 다른 기술적 과제는 PC에 대하여 제우스(Zeus)의 감염여부를 판단하고, 외부로 송신되는 패킷에 대해 제우스(Zeus) 패킷인지를 탐지할 수 있는 C&C제우스의 윈도우 API후킹 및 패킷 길이에 기반한 탐지 방법을 제공하는데 있다.

과제의 해결 수단

- [0009] 상기 목적을 달성하기 위한, 본 발명에 따른 C&C제우스의 윈도우 API 후킹 및 패킷 길이에 기반한 탐지 방법은 패킷에서 제우스 바디(Zeus body) 크기를 산출하는 단계; 상기 제우스 바디 크기를 제1암호화 하여 특정 위치에 암호화된 정보를 저장하는 단계; 상기 제우스 바디 크기 및 암호화된 정보를 제2암호화 하여 상기 제1암호화에 사용된 키(key)값을 추출하는 단계; 및 상기 추출된 키 값을 이용하여 암호화된 데이터의 감염여부를 판단하는 단계를 포함할 수 있다.
- [0010] 이 때, 시스템에서 변형된 API를 추출하여 제우스를 탐지하는 단계를 더 포함할 수 있다.
- [0011] 상기 제1암호화 및 제2암호화는 XOR 암호화일 수 있다.
- [0012] 상기 제1암호화 하여 특정 위치에 암호화된 정보는 Cipher Text를visualDecrypt한 뒤의 특정 위치의 내용이 패킷의 크기에 대응하는 암호화된 정보일 수 있다.
- [0013] 상기 제우스 바디 크기는 플레인 텍스트(Plain Text)일 수 있다.
- [0014] 상기 추출된 키 값을 이용하여 암호화된 데이터의 감염여부를 판단하는 단계는, 상기 제1암호화에서 추출된 키와 상기 제2암호화에서 추출된 키가 동일한 경우, 제우스에 감염된 패킷으로 판단할 수 있다.
- [0015] 본 발명에 따른 C&C제우스의 윈도우 API 후킹 및 패킷 길이에 기반한 탐지 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 포함할 수 있다.

발명의 효과

- [0016] 본 발명에 따른 C&C 제우스의 윈도우 API후킹 및 패킷 길이에 기반한 탐지 방법에 의하면, 위협적인 멀웨어인 제우스(Zeus)의 C&C버전에서 사용하는 윈도우 API후킹 기법 및 타겟 PC가 제우스(Zeus)감염이 되었는지를 탐지할 수 있으며, 감염된 PC가 송신하는 제우스(Zeus)패킷의 탐지할 수 있다. 또한, 종래에 비해 제우스를 보다 정

확하게 탐지할 수 있다.

도면의 간단한 설명

- [0017] 도 1은 본 발명의 일 실시예에 따른 C&C 제우스(Zeus)의 윈도우 API후킹 및 패킷 길이에 기반한 탐지 시스템의 구성을 도시한 블록도이다.
- 도 2는 본 발명의 일 실시예에 따른 C&C 제우스(Zeus)의 윈도우 API후킹 및 패킷 길이에 기반한 탐지 방법을 나타내는 흐름도이다.
- 도 3은 프로세서에 제우스가 삽입되는 과정을 나타내는 도면이다.
- 도 4는 C&C제우스 감염과정을 나타내는 흐름도이다.
- 도 5는 후킹 전 인터넷 익스플로러의 LdrLoadDll API을 나타내는 도면이다.
- 도 6은 후킹 후 인터넷 익스플로러의 LdrLoadDll API을 나타내는 도면이다.
- 도 7은 일 실시예에 따른 C&C 제우스 후킹 탐지 방법을 설명하기 위한 도면이다.
- 도 8은 다른 실시예에 따른 C&C 제우스 후킹 탐지 방법을 설명하기 위한 도면이다.
- 도 9는 제우스 리포트(Zeus Report) 패킷 헤더(header) 구조를 나타내는 표이다.
- 도10는 제우스 리포트(Zeus Report) 패킷 아이템(item) 구조를 나타내는 표이다.
- 도 11는 제우스 암호화 과정을 나타내는 흐름도이다.
- 도 12은 제우스 복호화 과정을 나타내는 흐름도이다.
- 도 13은 visualEncrypt, visualDecrypt의 알고리즘을 나타내는 도면이다.
- 도 14은 본 발명의 일 실시예에 따른 제우스(Zeus) 복호화 과정을 나타내는 흐름도이다.
- 도 15는 본 발명의 일 실시예에 따른 ID가 나타내는 데이터에 대한 표이다.
- 도 16은 본 발명의 일 실시예에 따른 실험결과를 나타내는 도면이다.

발명을 실시하기 위한 구체적인 내용

- [0018] 이하 본 발명의 실시예를 첨부된 도면들을 참조하여 상세히 설명할 수 있다. 우선 각 도면의 구성요소들에 참조 부호를 부가함에 있어서, 동일한 구성요소들에 대해서는 비록 다른 도면상에 표시되더라도 가능한 한 동일한 부호를 가지도록 하고 있음에 유의해야 할 수 있다. 또한 본 발명을 설명함에 있어, 관련된 공지 구성 또는 기능에 대한 구체적인 설명이 당업자에게 자명하거나 본 발명의 요지를 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명은 생략할 수 있다.
- [0019] 도 1은 본 발명의 일 실시예에 따른 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 시스템의 구성을 도시한 블록도이다.
- [0020] 도1을 참조하면, 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 시스템 (1)은 입력부(100), 제어부(200), 출력부(300) 및 저장부(400)을 포함한다.
- [0021] 우선, 입력부(100)는 사용자 명령을 입력받을 수 있다. 사용자 명령은 패킷 형식일 수 있다.
- [0022] 제어부(200)는 입력부(100)에서 입력받은 사용자 명령어에 따라 알고리즘을 수행할 수 있다. 제어부(200)는 패킷에서 제우스 바디(Zeus body) 크기를 산출할 수 있다. 제어부(200)는 제우스 바디 크기를 제1암호화 하여 특정 위치에 암호화된 정보를 저장하도록 제어할 수 있다. 제어부(200)는 상기 제우스 바디 크기 및 암호화된 정보를 제2암호화 할 수 있다. 제어부(200)는 제2암호화를 통해 상기 제1암호화에 사용된 키(key)값을 추출할 수 있다. 제어부(200)는 추출된 키 값을 이용하여 암호화된 데이터의 감염여부를 판단할 수 있다. 또한, 제어부(200)는 시스템에서 변형된 에이피아이(API)를 추출하여 제우스를 탐지할 수 있다.

- [0023]
- [0024] 출력부(300)는 제어부(200)에서 감염됨이 탐지된 제우스 정보를 출력할 수 있다.
- [0025] 저장부(400)는 제어부(200)에서 산출된 값들을 저장할 수 있다. 저장부(400)는 사용자 명령어를 저장할 수 있다. 또한, 저장부(400)는 제우스 바디 크기, 암호화된 정보 및 추출된 키(key) 값 중 적어도 하나를 저장할 수 있다.
- [0026] 도 2는 본 발명의 일 실시예에 따른 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 방법을 나타내는 흐름도이다.
- [0027] 도 2를 참조하면, 우선, 패킷에서 제우스 바디(Zeus body) 크기를 산출한다(S110). 상기 제우스 바디 크기는 플레인 텍스트(Plain Text)일 수 있다.
- [0028] 다음, 산출된 제우스 바디 크기를 제1암호화 하여 특정 위치에 암호화된 정보를 저장한다(S120). 보다 상세하게, 산출된 제우스 바디 크기를 XOR하여 특정 위치에 암호화된 정보를 저장할 수 있다. 여기서, 상기 제1암호화 하여 특정 위치에 암호화된 정보는 암호문(Cipher Text)을 비주얼 디크립트(visualDecrypt)한 뒤의 특정 위치의 내용이 패킷의 크기에 대응하는 암호화된 정보일 수 있다.
- [0029] 다음, 상기 제우스 바디 크기 및 암호화된 정보를 제2암호화 한다(S130). 보다 상세하게, 상기 제우스 바디 크기 및 암호화된 정보를 XOR하여 암호화할 수 있다.
- [0030] 다음, 상기 제2암호화를 통해 상기 제1암호화에 사용된 키(key)값을 추출한다(S140).
- [0031]
- [0032] 다음, 상기 추출된 키 값을 이용하여 암호화된 데이터의 감염여부를 판단한다(S150). 보다 상세하게, 상기 제1암호화에서 추출된 키와 상기 제2암호화에서 추출된 키가 동일한 경우, 제우스에 감염된 패킷으로 판단할 수 있다.
- [0033] 마지막으로, 시스템에서 변형된 에이피아이(API)를 추출하여 제우스를 탐지한다(S160).
- [0034] 상기 설명된 씨앤씨(C&C) 제우스(Zeus)의 윈도우 에이피아이(API) 후킹 및 패킷 길이에 기반한 탐지 방법은 도 14 내지 도 16을 통하여 보다 상세하게 설명한다.
- [0035] 도 3은 프로세서에 제우스가 삽입되는 과정을 나타내는 도면이다. 도 4는 씨앤씨(C&C) 제우스(Zeus) 감염과정을 나타내는 흐름도이다.
- [0036] 도 3 및 도 4를 참조하면, 로더(loader) 파일을 실행하면, 우선 자신을 시작프로그램에 등록하여 매 부팅시마다 실행하도록 한 다음에 실행 중인 모든 프로세스에 자신의 이미지를 삽입시켜 제우스(Zeus)에 감염되도록 한다.
- [0037] 제우스(Zeus) 이미지를 모든 프로세스에 삽입시키는 과정을 알아보기 위해 도3을 참고해보면, 우선 제우스(Zeus)는 크리에이트툴헬프32스냅샷 에이피아이(CreateToolhelp32Snapshot API)를 통해 현재 실행 중인 모든 프로세스 목록을 얻어온 뒤에 오픈프로세스 에이피아이(OpenProcess API)를 통해 그 중 하나의 프로세스에 대한 핸들을 추출한다. 추출된 핸들을 이용하여 버추얼얼록 에이피아이(VirtualAlloc API)를 통해 해당 프로세스 내에 가상 메모리 공간을 확보한 후, 라이트프로세스메모리 에이피아이(WriteProcessMemory API)로 제우스(Zeus) 이미지를 가상 메모리 공간에 삽입하게 된다. 이렇게 삽입된 제우스(Zeus) 이미지는 로더(loader)에서 크리에이트리모트스레드 에이피아이(CreateRemoteThread API)를 통하여 실행되어 제우스(Zeus)가 프로세스 내에서 작동하게 된다. 이러한 방법을 반복하여 로더(loader)는 모든 프로세스에 제우스(Zeus)를 삽입 및 실행하도록 한다.
- [0038] 감염된 이후에는 여러 가지 서비스들을 활성화 시키며, 프로세스에 로드된 API들에 대한 후킹을 시도한다. 서비스들은 주로 서버와의 통신에 사용이 되며, 후킹된 API는 사용자의 개인정보를 넘겨받거나, 새로운 프로세스 및

스레드가 생성되었을 때 제우스(Zeus) 이미지를 삽입시키는데 사용된다.

- [0039] 도 5는 후킹 전 인터넷 익스플로러의 엘디알로드디엘엘 에이피아이(LdrLoadDll API)를 나타내는 도면이다. 도 6은 후킹 후 인터넷 익스플로러의 엘디알로드디엘엘 에이피아이(LdrLoadDll API)를 나타내는 도면이다.
- [0040] 도 5 및 도 6을 참조하면, 제우스(Zeus)의 후킹은 에이피아이(API)를 실행하였을 때 제우스(Zeus)의 코드로 이동하도록 하기 위해 제이엠펜 인스트럭션(jmp instruction)을 사용한다.
- [0041] 제우스(Zeus)에 감염이 되기 전에는 인터넷 익스플로러의 엘디알로드디엘엘 에이피아이(LdrLoadDll API)의 코드는 도 5에 도시되어 있다. 하지만 제우스(Zeus)의 감염이 되어 로더(loader)로 실행이 되면, 후킹이 일어나며 에이피아이(API)의 첫 부분이 제이엠펜 인스트럭션(jmp instruction)으로 바뀌게 된다. 도5에서 'PUSH 26C'가 도 6에서 'JMP 00142A70'으로 바뀌었음을 볼 수 있다. 00142A70 위치로 가면, 삽입된 제우스(Zeus) 코드가 있음을 볼 수 있다.
- [0042] 결국, 감염이 된 이후에는 인터넷 익스플로러에서 엘디알로드디엘엘 에이피아이(LdrLoadDll API)를 실행하면 제이엠펜(jmp)을 통해 바로 제우스(Zeus)의 코드가 실행되게 되며, 코드에서 특정 작업이 완료된 후 제우스(Zeus)는 다시 정상적으로 엘디알로드디엘엘 에이피아이(LdrLoadDll API)를 실행하게 된다. 따라서 일반적인 사용자들은 감염이 되었는지 판단할 수 없다.
- [0043] 도 7은 일 실시예에 따른 씨앤씨(C&C) 제우스(Zeus) 후킹 탐지 방법을 설명하기 위한 도면이다. 도 8은 다른 실시예에 따른 씨앤씨(C&C) 제우스(Zeus)후킹 탐지 방법을 설명하기 위한 도면이다.
- [0044] 도 7 및 도 8을 참조하면, 제우스(Zeus)의 후킹 기법은 에이피아이(API)의 첫 부분을 제이엠펜 인스트럭션(jmp instruction)으로 바꾸는 것으로 이루어진다. 따라서, 현재 PC가 제우스(Zeus)에 감염되었는지 탐지하기 위해서 이러한 특징을 이용할 수 있다.
- [0045] 우선 제우스(Zeus)가 감염된 이후에는 모든 프로세스에 바이너리가 삽입되어 각각의 프로세스들에 대한 후킹을 하게 된다. 따라서, 크리에이트툴헬프32스냅샷(CreateToolhelp32Snapshot) 등의 에이피아이(API)를 사용하여 현재 실행 중인 모든 프로세스의 리스트를 얻은 후, 각각의 프로세스의 아이아에티(IAT:Import Address Table)에 접근하여 임포트된 에이피아이(API)들에 대하여 첫 바이트에 대해서만 바이트를 탐색한다. 제이엠펜 인스트럭션(jmp instruction)의 오퍼코드(opcode)는 'E9'로 시작하기 때문에 첫 바이트가 'E9'라면 제우스(Zeus)에 후킹 된 에이피아이(API)라고 판단할 수 있다. 일반적인 윈도우(Windows) 에이피아이(API)는 제이엠펜 인스트럭션(jmp instruction)으로 시작하지 않기 때문이다.
- [0046] 삭제
- [0047] 보다 상세하게, 확인할 프로세스는 cmd.exe를 선택하였으며 에이피아이(API)는 겟파일어트리뷰트엑스(GetFileAttributesEx)를 선택하였다. 에이피아이(API)의 주소는 프로세스가 로드될 때마다 바뀔 수 있으므로 중요하지는 않으며, 실제 중요한 부분은 에이피아이(API)의 주소의 첫 5바이트이다. 감염되지 않은 도 7에서는 첫 5바이트가 '8b ff 55 8b ec'로 스택 프레임을 구성하는 오퍼코드(opcode)가 들어있는 반면, 도 8에서는 첫 5바이트가 'e9 d9 19 94 83'으로 구성되어 오퍼코드(opcode)가 'e9'로 시작함을 확인할 수 있으며 이것은 즉 제이엠펜 인스트럭션(jmp instruction)을 의미한다. 따라서 도 8의 시스템은 제우스(Zeus)가 감염되었다고 탐지할 수 있는 것이다.
- [0048] 도 9는 제우스 리포트(Zeus Report) 패킷 헤더(header) 구조를 나타내는 표이다. 도 10은 제우스 리포트(Zeus Report) 패킷 아이템(item) 구조를 나타내는 표이다.
- [0049] 도 9 및 도 10을 참조하면, 씨앤씨(C&C) 제우스(Zeus)에서는 서버(Server)와 봇(Bot)간의 통신을 위해 주기적으로 여러 종류의 패킷이 전송되는데, 그 중 하나는 리포트(Report) 패킷이라고 부른다. 리포트(Report) 패킷은 봇(Bot)에서 수집한 개인 정보를 서버(Server)에 전달할 때 사용되는 패킷으로써, 구조는 도 9 및 도 10과 같다. 보다 구체적으로, 리포트(Report) 패킷은 크게 헤더(header)와 바디(body)로 구성되어 있으며, 헤더(header)는 도9와 같은 구조를 가지며, 바디(body)는 여러 개의 아이템(item)이 결합되어 있으며, 하나의 아이템(item)은 도 10과 같은 구조를 가질 수 있다.
- [0050]

- [0051] 도 11은 제우스 암호화 과정을 나타내는 흐름도이다. 도 12는 제우스 복호화 과정을 나타내는 흐름도이다. 도 13은 비주얼 인크립트(visualEncrypt), 비주얼 디크립트(visualDecrypt)의 알고리즘을 나타내는 도면이다.
- [0052] 도 11 내지 도 13을 참조하면, C&C (Zeus)에서는 크게 2가지의 암호화 알고리즘을 사용한다. 하나는 RC4 알고리즘이며, 또 하나는 비주얼 인크립트(visualEncrypt), 비주얼 디크립트(visualDecrypt)로 구성되어 있는 제우스 (Zeus) 고유의 알고리즘이다.
- [0053] 이때, 암호화 또는 복호화 되는 과정은 도 11 및 도 12에 도시되었으며, 비주얼 인크립트(visualEncrypt)와 비주얼 디크립트(visualDecrypt)로 구성되는 제우스 고유 알고리즘은 도 13에 도시된다. 상기 제우스 고유 알고리즘은 간단한 XOR 과정 거치는 함수로써, 구현되기에 용이하다.
- [0054] 보다 구체적으로, 전송되는 패킷은 암호문(Cipher Text)으로 전송이 되므로, 패킷을 받아 RC4로 복호화를 하고, 비주얼 디크립트(visualDecrypt) 과정을 거치면 플레인 텍스트(Plain Text)를 얻을 수 있다. 하지만, RC4 복호화를 위해서는 스트림 키(stream key)가 있어야 한다. 제우스(Zeus) 초기 버전에서는 스트림 키(stream key)가 봇(Bot)의 특정한 메모리 주소에 존재하여 가져 올 수 있었지만, 2.0.8.9 버전에서는 임의의 메모리 주소에 존재하여 가져올 수가 없다. 따라서 정상적으로 복호화는 불가능하다. 따라서, 2.0.8.9 버전에서도 정상적으로 복호화할 수 있는 프로그램이 요구된다.
- [0055] 도 14는 본 발명의 일 실시예에 따른 제우스(Zeus) 복호화 과정을 나타내는 흐름도이다.
- [0056] 도 14를 참조하면, 상기 도 12를 통해 설명한 복호화 과정에서 RC4 과정을 생략한 과정을 거쳤을 경우에 나오는 뉴 텍스트(New Text)는 플레인 텍스트(Plain Text)와는 다르지만, 다른 RC4 키를 사용한 것과 같은 결과를 추출하며, 여러 개의 플레인 텍스트(Plain Text)에서 같은 위치에 동일한 바이트가 존재한다면 뉴 텍스트(New Text)에서도 같은 위치에 존재하는 바이트 값이 같은 것이 확인 되었다.
- [0057] 따라서, 각각의 리포트(Report) 패킷에서 동일한 위치에 같은 바이트가 기록되어 있다면, 다른 패킷들과 구별할 수 있는 특징을 뽑아낼 수 있다.
- [0058] 도 15는 본 발명의 일 실시예에 따른 ID가 나타내는 데이터에 대한 표이다. 도 16은 본 발명의 일 실시예에 따른 실험결과를 나타내는 도면이다.
- [0059] 도 15 및 도 16을 참조하면, 도 10에서 볼 때, 각각의 아이템(item)은 뒤에 따라오는 데이터(data)에 어떠한 내용이 들어 있는지 가장 처음에 있는 4바이트의 ID로 그 내용이 나타내며, 도 15는 ID가 가리키는 데이터를 나타낸다.
- [0060] 이 중에 봇(BOT) ID에 해당하는 10001은 모든 리포트(Report) 패킷 바디(body)에 항상 첫 아이템(item)으로 추출된다. 따라서 10001이 해당하는 위치인 49~52 바이트는 C&C 제우스(Zeus)라면 모두 같은 값을 가지고 있으며, 이로써, New Text의 49~52 바이트도 모두 동일한 값을 가진다.
- [0061] 또한, 동일한 봇(Bot)이 동일한 서버(Server)에 리포트(Report) 패킷을 전송한다면 같은 봇(Bot) ID로 전송을 하기 때문에 49~52 바이트뿐만 아니라 그뒤에 나오는 데이터 길이(data length)와 데이터(data) 부분도 같게 되며, 이는 심도 있는 탐지에 도움이 된다.
- [0062] 또한, 제우스(Zeus)의 암호화 알고리즘은 패킷의 크기를 변동시키지 않으므로, 만약 바디(body)의 크기가 같다면 헤더(header)에서 바디 크기(body size)에 해당하는 21~24 번째 바이트의 값도 동일할 것이다. 본 발명에서는 이와 같은 방법을 사용하여 제우스(Zeus) 패킷을 탐지한다.
- [0063] 보다 상세하게, 해당 패킷에서 제우스 바디(Zeus body)의 크기를 알면, 상기 제우스 바디의 크기는 플레인 텍스트(Plain Text)에 해당하고, 도 14에 따라 암호문(Cipher Text)을 비주얼 디크립트(visualDecrypt)한 뒤의 21~24번째의 내용은 바디(body)의 크기에 대응하는 암호화된 내용이 된다.
- [0064] RC4는 곧 XOR 암호화와 내용이 일치하므로 상기 제우스 바디의 크기와 암호화된 내용을 XOR 하면 새로운 키(key)가 나올 것이다. 이전의 RC4 암호화에서 같은 키(key)를 사용했다면 이 새로운 키(key)는 모두 같게 된다. 도 16은 이러한 과정을 통해 새로운 키(key)를 구하는 도면이다.

[0065] 도 16에서 사용된 암호문(Cipher Text)은 같은 서버와 클라이언트에서 클라이언트가 보내는 리포트(Report) 패킷을 캡처한 것들의 일부이며, 같은 RC4 키(key)를 사용한다. 암호문(Cipher Text)을 비주얼 디크립트(visualDecrypt) 한 후, 바디(body)의 크기를 리틀 엔디안(little endian)으로 변환하여 XOR 연산을 하면 모두 0xc409ea0e 라는 같은 값이 나오며, 이것이 도 14에서 언급한 뉴 스트림 키(new stream key)의 일부에 해당한다.

[0066] 따라서, 네트워크 상의 패킷들을 캡처하여 비주얼 디크립트(visualDecrypt) 과정을 거친 후, 21 내지 24번째의 내용과 크기를 XOR 하였을 때, 같은 값들이 나오는 경우가 있다면, 이는 Zeus와 관련된 패킷이라고 탐지할 수 있다.

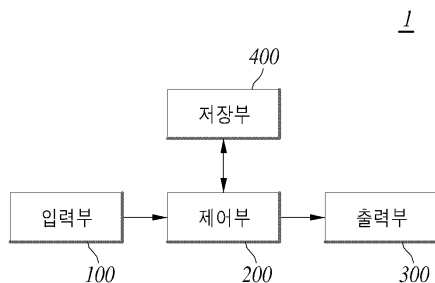
[0067] 이상에서 본 발명의 바람직한 실시예에 대해 도시하고 설명하였으나, 본 발명은 상술한 특징의 바람직한 실시예에 한정되지 아니하며, 청구범위에서 청구하는 본 발명의 요지를 벗어남이 없이 당해 발명이 속하는 기술분야에서 통상의 지식을 가진 자라면 누구든지 다양한 변형 실시가 가능한 것은 물론이고, 그와 같은 변경은 청구범위 기재의 범위 내에 있게 된다.

부호의 설명

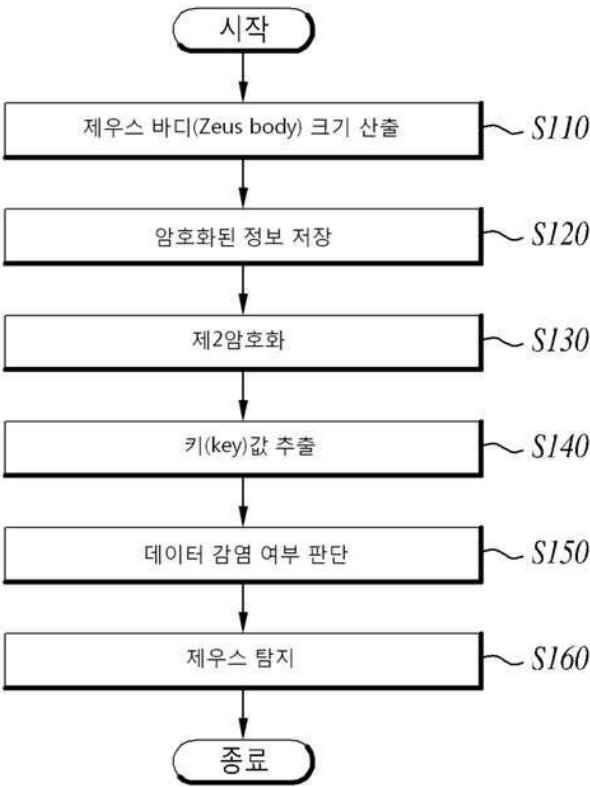
[0068] 100: 입력부
200: 제어부
300: 출력부
400: 저장부

도면

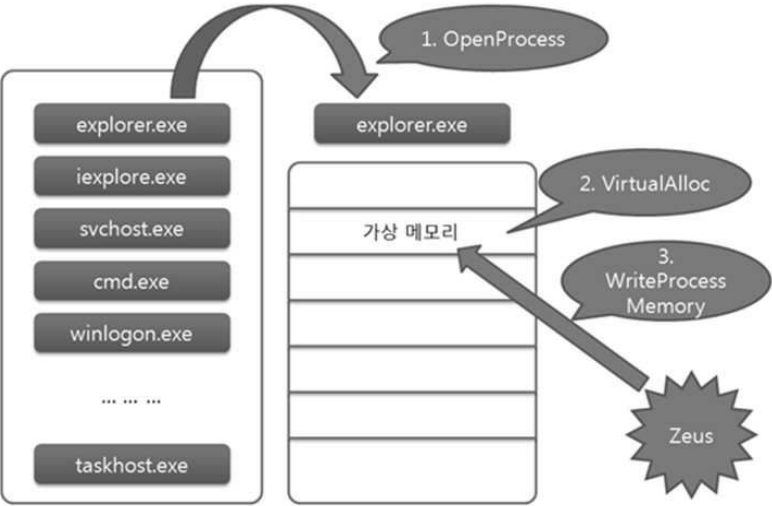
도면1



도면2



도면3



도면4



도면5

7C9463A3	68 6C020000	PUSH 26C
7C9463A8	68 F864947C	PUSH 7C9464F8
7C9463AD	E8 F984FFFF	CALL 7C93E8AB
7C9463B2	A1 C8B09A7C	MOV EAX,DWORD PTR DS:[7C9AB0C8]

도면6

7C9463A3	- E9 C8C67F83	JMP 00142A70
7C9463A8	68 F864947C	PUSH 7C9464F8
7C9463AD	E8 F984FFFF	CALL 7C93E8AB
7C9463B2	A1 C8B09A7C	MOV EAX,DWORD PTR DS:[7C9AB0C8]

도면7

```
cmd.exe [3588]
targetAddress = 0x75f6453c <GetFileAttributesEx>
5Bytes : 8b ff 55 8b ec
```

도면8

```
cmd.exe [1152]
targetAddress = 0x7c811185 <GetFileAttributesEx>
5Bytes : e9 d9 19 94 83 *****This process is suspicious*****
```

도면9

20	4	4	4	16
random padding	body size	flags	number of items	MD5

도면10

4	4	4	4	data length
item ID	zero padding	data length	data length	data

도면11



도면12



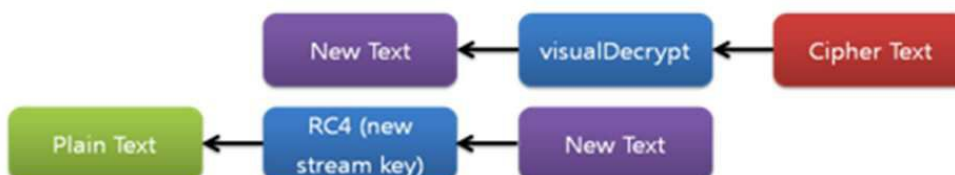
도면13

```

void Crypt::_visualEncrypt(void *buffer, DWORD size)
{
    for(DWORD i = 1; i < size; i++)
        ((LPBYTE)buffer)[i] ^= ((LPBYTE)buffer)[i - 1];
}

void Crypt::_visualDecrypt(void *buffer, DWORD size)
{
    if(size > 0)
        for(DWORD i = size - 1; i > 0; i--)
            ((LPBYTE)buffer)[i] ^= ((LPBYTE)buffer)[i - 1];
}
    
```

도면14



도면15

ID	Data Type
10001	SBCID_BOT_ID
10002	SBCID_BOTNET
10003	SBCID_BOT_VERSION
10005	SBCID_NET_LATENCY
10006	SBCID_TCPPORT_S1
10007	SBCID_PATH_SOURCE
10008	SBCID_PATH_DEST
10009	SBCID_TIME_SYSTEM
10010	SBCID_TIME_TICK

도면16

Cipher Text (Size 0x527)
cc:b6:c0:97:da:6d:90:ac:0e:e0:c0:53:f4:fb:21:f7 7d:3f:7a:cd:2e:22:c8:c6:1f:d7:94:6d:d6:6d:20:bc 3a:6f:3a:49:93:ff:bc:4f:95:5d:ca:51:85:28:99:30
After VisualDecrypt (Size 0x527)
00:7a:76:57:4d:b7:fd:3c:a2:ee:20:93:a7:0f:da:d6 8a:42:45:b7:e3:0c:ea:0e:d9:c8:43:f9:bb:bb:fd:9c 86:55:55:73:da:6c:43:f3:da:c8:979:9b:d4:ad:b1:a9
XOR
0xe30cea0e XOR 0x27050000 = 0xc409ea0e
Cipher Text (Size 0x15a)
3a:fc:1e:a1:7b:00:38:8c:24:d8:81:62:c5:fe:36:96 C4:df:9a:5f:c1:c9:23:2d:f4:3c:7f:86:3a:81:cc:50 fa:f8:65:f5:9c:1e:aa:b8:87:8b:8a:51:3a:f4:87:01
After VisualDecrypt (Size 0x15a)
00:c6:e2:bf:da:7b:38:b4:a8:fc:59:e3:a7:3b:c8:a0 52:1b:45:c5:9e:08:ea:0e:d9:c8:43:f9:bc:bb:4d:9c aa:02:9d:90:69:82:b4:12:3f:0c:01:db:6b:ce:73:86
XOR
0x9e08ea0e XOR 0x5a010000 = 0xc409ea0e
Cipher Text (Size 0x57c)
d9:38:4f:33:f1:7a:52:07:f7:18:ec:e0:a7:e1:75:7a cf:0f:fe:43:fb:f7:1d:13:ca:02:41:b8:03:b8:f5:69 97:1f:a3:6f:4e:e6:76:3c:8d:f1:59:fc:39:b1:13:8a
After VisualDecrypt (Size 0x57c)
00:e1:77:7c:c2:8b:28:55:f0:ef:f4:0c:47:46:94:0f b5:c0:f1:bd:b8:0c:ea:0e:d9:c8:43:f9:bb:bb:4d:9c fe:88:bc:cc:21:a8:90:4a:b1:77:a3:a5:c5:88:a2:99
XOR
0xb80cea0e XOR 0x7c050000 = 0xc409ea0e