2020
한국정보보호학회 하계학술대회

CISC-S'20

Conference on Information Security and Cryptography - Summer 2020

07.15 WED
온라인 컨퍼런스
http://www.cisc.or.kr/

주최 주관 한국정보보호학회 Korea Institute of Information Security & Cryptology

후원 국가정보원 National Intelligence Service Korea · 과학기술정보통신부 · 행정안전부 · KISA 한국인터넷진흥원 KOREA INTERNET & SECURITY AGENCY · ETRI 한국전자통신연구원 Electronics and Telecommunications Research Institute · NSR 국가보안기술연구소 National Security Research Institute · cen 아이티센 ITcen

클라우드 오케스트레이션 기반 침입인지형 Active-Decoy 시스템 디자인

주재경*, 박기웅†

* 세종대학교 시스템 보안 연구실, † 세종대학교 정보보호학과

Design of Intrusion-Aware Active-Decoy System based on Cloud -Orchestration

Jaegyeong Ju*, Ki-Woong Park†

* System Security Laboratory, Sejong University

† Department of Computer and Information Security, Sejong University

요 약

타겟을 보호하기 위해 외부에 노출된 접점(Attack Surface, 이하 공격 표면)을 시간에 따라 능동적으로 변이(Mutation)시키는 MTD(Moving Target Defense)라는 기술이 존재한다. 하지만 공격 표면의 능동적인 변이는 공격자뿐만 아니라 정상적인 사용자까지 영향을 미치며 실제 구현에 따른 비용, 안전성 등의 문제점이 존재한다.

본 논문에서는 자원의 유연한 확장을 지원하는 클라우드 환경에서 디코이와 컨테이너 오케스트레이션을 활용하여 타겟의 공격 표면을 확장시키는 한편 공격자가 타겟의 정보를 수집하는 행위를 인지하여 능동적으로 디코이의 수를 변이시켜 자원을 효율적으로 사용하며 대상 시스템을 보호하는 *Active-Decoy* 기술을 제안한다. 이 기술을 통해 타겟의 영향과 자원의 낭비를 최소화함으로써 안전성 및 보안성을 확보할 수 있다.

I. 서론

최근 IDG의 클라우드 컴퓨팅 연구 보고서에 따르면 기업의 IT 환경은 기존 온프레미스에서 클라우드 환경으로 많은 마이그레이션이 진행되고 있다[1]. 한편 공격자는 타겟에 침입하기 위해 시간 대부분을 정보수집과 그에 따른 공격 시나리오 작성에 투자한다[2]. 그로 인해 보안 시스템은 타겟으로 침입 가능한 모든 공격 표면을 방어해야 하는 반면 공격자는 침입 가능한 공격 표면 중 어느 곳이든 침입에 성공하면 되므로 공격자 우위의 비대칭적 공방관계가 형성되어 있다[3].

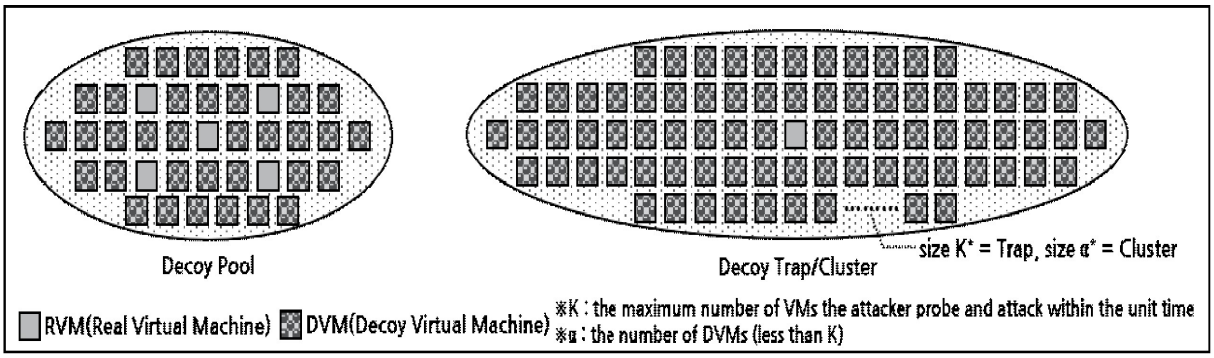
† 교신저자: 박기웅 (세종대학교 정보보호학과 교수)

본 연구는 과학기술정보통신부의 재원으로 정보통신기획평가원(IITP)의 지원(No.2018-0-00420, No.2019-0-00426) 및 한국연구재단 연구과제(NRF-2020R1A2C4002737)의 지원을 받아 수행된 연구임.

이러한 비대칭적 공방관계를 해결하기 위해 다양한 보안 패러다임이 제시되고 있는 가운데 외부에 노출된 공격 표면을 능동적으로 변이시키는 MTD라는 기술이 등장했다. MTD는 타겟의 공격 표면을 시간에 따라 능동적으로 변이시켜 공격자가 대상 시스템의 취약점 분석에 시간적 제약사항을 부여해 타겟에 침입하기 어렵게 하는 기술이지만 실제 적용에 따른 비용, 안전성 등의 문제점이 존재한다[4].

이러한 문제를 해결하기 위해 본 논문에서는 자원의 유연한 확장을 지원하는 클라우드 환경에서 '디코이'와 '컨테이너 오케스트레이션'을 활용하여 타겟의 공격 표면을 확장[5]시키며 공격자가 타겟의 정보를 수집하는 행위를 인지하여 능동적으로 디코이의 수를 변이시켜 자원을 효율적으로 사용하며 대상 시스템을 보호하는 *Active-Decoy* 기술을 제안한다.

본 논문은 2장에서 '디코이'와 '컨테이너 오케스트레이션'에 대한 소개와 제안한 기술이 갖



[그림1] 디코이 모델 구조 [5]

는 의의를 제시한다. 3장에서는 *Active-Decoy* 시스템의 핵심 기능과 개략적인 실행과정을 디자인하며 4장에서는 결론을 통해 전체 장의 내용을 요약하며 추후연구 방향을 제시한다.

II. 관련 연구

본 장에서는 본 논문에서 제안한 모델의 배경이 되는 기술을 소개하고 제안한 모델이 갖는 의의를 제시한다.

2.1. 디코이

일반적으로 디코이라 하면 공격자로부터 타겟을 보호하기 위해 공격자를 교란시키기 위한 더미를 지칭한다. 본 절에서는 다양한 곳에서 활용되고 있는 디코이 중 본 연구주제의 타겟(클라우드)을 고려하여 가상화된 인프라를 보호하기 위해 제안된 디코이 모델을 살펴본다.

■ 디코이 풀[5]은 [그림1]과 같이 DVM이 RVM과 동일한 형태로 배포되며 해당 방법은 DVM 사이에 RVM을 숨김으로써, 공격자의 공격 성공률을 낮추는 것을 목표로 한다.

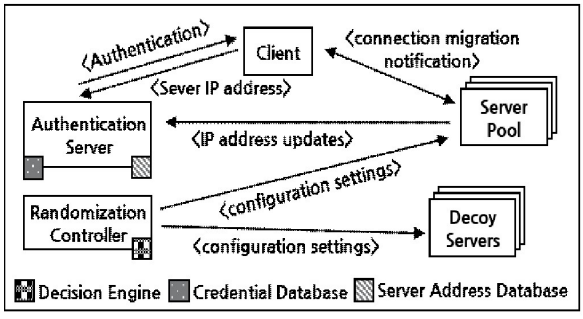
■ 디코이 트랩[5]은 [그림1]과 같이 디코이의 그룹이 트랩으로 설정되며, 트랩 하나의 크기는 공격자가 단위 시간 내에 탐색하고 공격할 수 있는 최대 횟수(이하 K)로 사전에 파악할 수 있다는 가정에 따라 설계를 진행하며, 전체 트랩은 하나의 RVM을 보호하도록 설정한다.

■ 디코이 클러스터[5]는 디코이 트랩에서 K값을 파악할 수 있다는 가정이 현실적이지 못하기 때문에 제안된 방법으로 [그림1]과 같이 RVM에 α개의 DVM을 배치하여 RVM을 보호한다. α가 K값과 같거나 커지면 디코이 클러스터는 디코이 트랩이 된다.

앞서 살펴본 디코이 모델과 본 논문에서 제안한 *Active-Decoy*의 목적은 다르지 않다. 하지만 해당 디코이 모델들은 컴퓨팅 자원을 소모함에 유연한 대처가 어려워 효율적인 자원 관리에 이슈가 발생할 수 있으며 이는 곧 비용 문제와 직결된다. 따라서 디코이 설계 시 보안과 비용의 균형을 고려하여 효율적인 자원 관리가 요구된다.

다음은 공격 표면을 확장시키는 디코이와 공격 표면을 능동적으로 변이시키는 MTD를 융합한 네트워크 기반 MTD 모델을 살펴본다.

■ DESIR(Decoy-Enhanced Seamless IP Randomization)[6]는 [그림2]와 같이 클라이언트, 인증 서버, 서버 풀, 디코이 베드, RC(randomization controller)로 구성된다. RC는 사전에 설정한 정보를 바탕으로 서버 풀과 디코이 베드의 네트워크 속성을 주기적으로 변경시킴으로써 MTD를 수행한다. 하지만 네트워크 속성 업데이트에 따라 필연적으로 세션의 일시적인 중단이 발생하며, 인증 서버는 MTD의 대상이 아니므로 추가적인 보안이 필요하다.



[그림2] DESIR 구조 [6]

디코이와 MTD가 융합한 해당 모델은 공격 표면의 지속적인 변이에 따라 클라이언트와 서버의 연결이 일시적으로 끊김으로 안전성 측면에서 한계점이 명확하다. 더욱이 이 문제는 클라이언트와 많은 세션을 가진 서버일수록 재연결에 따른 오버헤드에 따른 대응책 또한 필요할 것으로 보여진다. 따라서 본 논문에서는 디코이를 배치해 공격 표면을 확장하여 대상 시스템을 보호함으로써, 공격 표면 변이에 따른 이슈를 회피해 안전성 측면을 해결하고자 한다.

2.2 오케스트레이션

이번 절에서는 오케스트레이션 기술을 살펴본다. 오케스트레이션은 컴퓨터 시스템 및 소프트웨어의 자동화된 구성, 조정, 관리하는 것을 말한다. 특히 컨테이너 오케스트레이션은 클라우드 환경에서 컨테이너의 배포 프로세스를 최적화하는 기능을 수행하며 호스트의 수가 많고 복잡한 시스템일수록 필수적으로 요구되는 기능이라 할 수 있다[7]. 아래는 컨테이너 오케스트레이션 도구를 비교한 내용으로 향후 연구에서 테스트베드 구축에 따른 방향을 제시하고자 한다.

■ 컨테이너 오케스트레이션 도구 특징

• Google Kubernetes[8] : 컨테이너화된 응용 프로그램의 배포, 확장 및 관리의 자동화를 해주는 시스템으로 유연한 서비스 검색 및 pod 내 IP 주소 간 로드 밸런싱 수행하며 클러스터 토폴로지를 기반으로 서비스 트래픽을 라우팅하며, 원하는 스토리지 시스템을 자동으로 마운트할 수 있고, 컨테이너에 문제가 발생하면 재기동, 재조정을 통해 자가 치료 기능을 하며, 유연한 이미지의 배포 및 업데이트가 가능하며, 롤아웃 롤백 기능 지원, 가용성에 영향을 주지 않고 설정에 따라 컨테이너를 자동으로 배치하며, 장애가 발생한 컨테이너의 유연한 교체도 가능하며, 간단한 조작으로 확장과 축소 또한 자유롭다.

• Docker Swarm[9] : 도커 컨테이너를 위한 클러스터링, 스케줄링 도구로 도커 엔진과 통합되어 오케스트레이션에 따른 별도의 소프트웨어가 필요하지 않고, 도커 엔진을 통해 노드, 관리자, 작업자 모두를 배포할 수 있으며, 애플리케이션 스택에서 다양한 서비스의 원하는 상태를 정의할 수 있고, 서비스의 상태 유지에 따른 유연한 스케일링 및 다중 호스트 네트워킹 등의 기능을 지원한다. 또한, TLS 상호 인증 및 암호화를 적용한 통신을 지원하며, 배포 및 업데이트에 따른 롤백을 제공한다.

• Apache Mesos[10] : 분산 시스템 커널이며 다양한 애플리케이션(hadoop, spark, kafka 등) 및 클라우드 환경에서 자원 관리 및 스케줄링이 가능한 도구로 1만개의 노드로 쉽게 확장할 수 있으며, 사용자 맞춤 퍼스트 클래스(CPU, 메모리, 디스크, 포트, GPU 등) 분리와 도커 컨테이너를 지원하며, 동일한 클러스터에서 클라우드 네이티브 및 레거시 애플리케이션의 실행이 가능하며, 클러스터 운영 및 모니터링을 위한 API와 컨테이너 샌드박스를 탐색하기 위한 웹 UI와 다양한 OS(Linux, MacOS, Windows)에서 실행할 수 있다.

• Coreos Fleet[11] : 로우 레벨의 클러스터 엔진으로 클러스터 내 임의 호스트에 도커 컨테이너를 배포할 수 있고, 안티 어피니티 기능을 통해 클러스터 간 서비스 배포, 인스턴스 관리 및 시스템 장애에 대한 스케줄링을 지원한다. 한편 2018년 이후 지원이 중단되어 더는 개발이나 관리가 되고 있지 않다.

• AWS ECS[12] : 아마존에서 제공하는 컨테이너 서비스로 도커 컨테이너와 자체 컨테이너 오케스트레이션을 지원하며, 수천 개의 컨테이너로 확장이 쉽고, 배포와 장애에 따른 롤백은 물론 비정상 컨테이너의 자동 복구와 스케줄링, 로드 밸런싱 기능 등을 지원한다. 무엇보다 AWS의 다른 서비스(EC2, Fargate 등)와 조합하여 유연한 사용이 가능하다.

• Redhat Cockpit[13][14] : 리눅스 서버를 관리하기 위한 관리자 중심의 웹 기반 GUI 서비스로 도커 모듈을 추가하여 컨테이너를 관리할 수 있다. 특히 도커가 포함된 리눅스 서버를 관리하고자 할 때 유용하며 대시보드를 통해

쉽게 컨테이너를 관리할 수 있다. 하지만 추가 모듈인 만큼 타 도구 수준의 기능을 지원하지는 못한다.

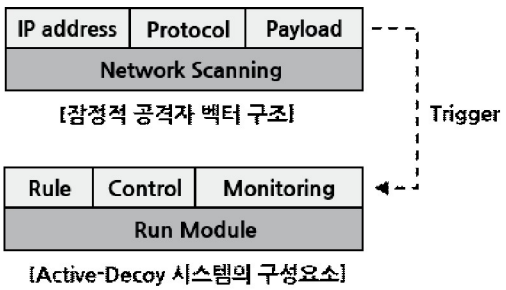
• Hashicorp Nomad[15] : 컨테이너 및 레거시 애플리케이션을 쉽게 배포하고 관리할 수 있도록 지원하는 도구로 자원 관리 및 예약을 단일 시스템으로 결합하고, 애플리케이션, 노드, 드라이버의 장애 발생을 자동으로 처리하며, GPU와 다양한 플러그인을 지원하며, 클러스터 간 연결을 통해 작업을 배포할 수 있고, 1만개 이상의 노드로 확장할 수 있다.

III. Active-Decoy 시스템

본 장에서는 제안한 기술의 요구사항과 *Active-Decoy* 시스템의 개략적인 실행과정을 설명한다.

3.1 Active-Decoy 시스템의 요구사항

공격자는 타겟에 침입하기 위해서 정보수집을 통해 공격 표면을 획득하고 취약점 분석에 따른 대상 시스템에 침입을 시도하는 일련의 과정을 수행한다. 이와 같은 수행 단계 중 본 논문에서는 정보수집 행위를 타겟 침입의 트리거로 정의한다. 보다 구체화하면 타겟 네트워크의 스캐닝 시도를 잠정적인 공격자의 정보수집 행위로 판단하여 *Active-Decoy* 시스템을 활성화하고자 한다. 그에 따라 타겟의 네트워크로 접근하는 패킷을 모니터링하고 스캐닝을 인지할 수 있는 규칙이 요구되며 시스템의 이슈 발생에 따른 유연한 대처를 위한 제어기능이 필요하다. 위의 내용을 도식화하면 [그림3]과 같다.

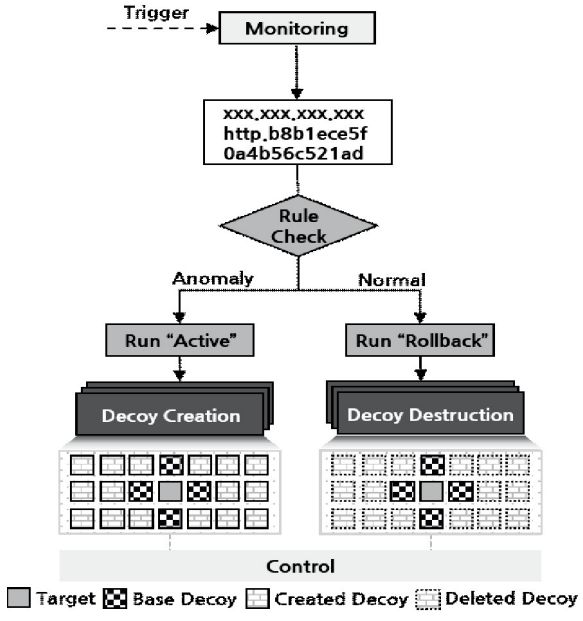


[그림3] Active-Decoy 시스템 요구사항

3.2 Active-Decoy 시스템 설계

타겟을 보호하는데 있어서 비용은 안전성만큼 중요한 요소이며 앞 장에서 살펴본 디코이의 모델은 타겟의 보호를 위해 고정적으로 컴퓨팅 자원의 할당이 필요하다. 특히 클라우드 환경의 특징 중 하나는 컴퓨팅 자원의 유연한 확장이며, 자원은 곧 비용이므로 모델 설계 시 핵심적으로 고려해야 할 부분이다.

이에 따라 본 논문에서는 효율적인 자원 관리를 위해 침입인지에 따른 디코이의 숫자를 조절하여 마치 세균의 침입에 따라 체내 백혈구의 개체 수가 능동적으로 변하는 것처럼 대처하고자 한다. 정리하자면 평상시 타겟의 규모와 트래픽을 고려하여 일정량의 디코이를 배치해두고, 타겟으로 접근하는 패킷을 모니터링하며 잠정적 공격자 벡터 패킷을 인지하는 순간 *Active-Decoy*가 활성화되어 디코이가 증식했다가 정보수집 행위가 일어나지 않다고 판단될 경우 다시 기존의 디코이 숫자로 돌아간다. 평상시에도 디코이를 일부 배치하여 침입인지에 따른 대응까지의 시간차를 이용, 공격자가 타겟을 스캐닝할 때 최소한의 교란(ex. 건물 내 소화기를 배치해 화재 발생 시 소방차가 오기 전까지 초동조치 실시)을 부여한다. 침입인지에 따른 *Active-Decoy*의 실행과정은 [그림4]와 같다.



[그림4] Active-Decoy 시스템 실행과정

IV. 결론

기존의 보안 전략은 타겟의 공격 표면을 공격자의 침입으로부터 수동적으로 대응함으로써 공격자 우위의 비대칭적 공방관계를 형성하였고, 공격 표면을 능동적으로 변이시키는 MTD라는 기술이 등장했지만, 구현에 따른 비용, 안전성 문제로 타겟에 적용하는데 어려움이 따른다. 본 논문은 이런 문제점을 해결하기 위해 *Active-Decoy*를 제안했다. 이 기술은 클라우드 환경에서 타겟 네트워크의 스캐닝 시도를 잠정적인 공격자의 정보수집 행위로 판단하여, 컨테이너 오케스트레이션을 활용해 타겟 주변에 디코이를 생성하여 실제 공격자가 정보를 수집할 경우 다수의 디코이와 타겟의 정보를 수집하게 함으로써, 수집된 정보를 토대로 타겟의 공격 표면을 특정하기 어렵게 만들어 타겟에 미치는 영향을 최소화(안전성 확보)할 수 있으며 유연성과 효율성을 겸비한 클라우드 환경에 적용함으로써 비용 문제에 대한 유연한 대응 또한 가능하다. 추후 연구로써는 본 논문에서 제시한 *Active-Decoy*의 테스트베드 구축을 위한 프레임워크를 개발할 예정이다.

[참고문헌]

[1] IDG. https://www.idg.com/tools-for-marketers/2020-cloud-computing-study/

[2] Kewley, D., Fink, R., Lowry, J., & Dean, M. “Dynamic approaches to thwart adversary intelligence gathering.” In Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01 (Vol. 1, pp. 176-185). IEEE, 2001.

[3] Cai, G. L., Wang, B. S., Hu, W., & Wang, T. Z. “Moving target defense: state of the art and characteristics.” Frontiers of Information Technology & Electronic Engineering, 17(11), 1122-1153, 2016.

[4] Okhravi, H., Hobson, T., Bigelow, D., & Streilein, W. “Finding focus in the blur of moving-target techniques.” IEEE Security & Privacy, 12(2), 16-26, 2013.

[5] Al-Salah, T., Hong, L., & Shetty, S.. “Attack surface expansion using decoys to protect virtualized infrastructure.” In 2017 IEEE International Conference on Edge Computing (EDGE) (pp. 216-219). IEEE, 2017.

[6] Sun, J., & Sun, K. “DESIR: Decoy-enhanced seamless IP randomization.” In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications (pp. 1-9). IEEE, 2016.

[7] 안성원. “클라우드 가상화 기술의 변화(이슈리포트: 제2018-008호).” 소프트웨어정책연구소, 2018.

[8] Google Kubernetes. https://kubernetes.io/

[9] Docker Swarm. https://docs.docker.com/engine/swarm/

[10] Apache Mesos. http://mesos.apache.org/

[11] Coreos Fleet. https://coreos.com/fleet/docs/latest/

[12] AWS ECS. https://aws.amazon.com/ecs/features/

[13] Redhat Cockpit. https://www.redhat.com/sysadmin/intro-cockpit

[14] Container Management with Cockpit. https://www.linux.com/topic/desktop/make-container-management-easy-cockpit/

[15] Hashicorp Nomad. https://www.nomadproject.io/