Gozi: The Malware with a Thousand Faces

research.checkpoint.com/2020/gozi-the-malware-with-a-thousand-faces/

August 28, 2020

August 28, 2020

Introduction

Most of the time, the relationship between cybercrime campaigns and malware strains is simple. Some malware strains, like the gone-but-not-forgotten GandCrab, are intimately tied to a single actor, who is using the malware directly or distributing it via an affiliate program. Other strains, like the open-source Quasar RAT, are "public domain" malware; they've remained the same for so long and been used as a Lego piece so repeatedly that it'd be a fundamental error to try and attribute them to an actor, a campaign, a victim or a time-frame.

It's comforting to think of malware in the above terms alone. Malware strains are then neatly mapped to actors (or at worst, actor-affiliate pyramids), who we feel comfortable reasoning about; and any malware not neatly mapped to an actor is just a depersonalized tool, which carries no dramatic baggage of its own that researchers need to keep track of.

Alas, reality is more complicated than that. Some strains of malware fall into a gray area. There is no single actor in control of the malicious codebase or the binaries, but there is no universal proliferation of the malware as a standard tool, either. No one is there to call all the shots, but there are many unrelated people each calling some of the shots, putting each branch of the malware in a state of constant divergent evolution. This is the strange and headache-inducing world of malware that has had its source code leaked, and it's not only the concern of researchers and analysts; the confusion and fragmentation that result have serious downstream effects for end users. A researcher flapping his hands in confusion somewhere in China can result, a few weeks later, in a hurricane of ransomware infections in the Caribbean, and a very angry manager being told "yes, we did have the MalBot family covered, but you see, *this* Malbot is not exactly *that* Malbot… look, it's complicated".

In this article we'll make sense out of one of the worst offenders in the category of divergent evolution malware: Gozi.

A Brief Genealogy of Gozi

Today, people know Gozi is a malware heavyweight that boasts an array of complicated features, on which we'll elaborate below, and a very wide reach. One Gozi flavor alone, Dreambot, at one point had 450,000 victims under its sway; when monitoring another flavor, which is currently active, we can see thousands of new unfortunate victims a week registered at the malicious C&C panels. Consider that a single victim can be monetized to the tune of hundreds of dollars, and you'll quickly come to the conclusion that in total Gozi has been *frighteningly* lucrative, even compared to the already lucrative cybercrime market.

Naturally, Gozi wasn't always such a juggernaut. Originally, it was a simple banking Trojan — even more primitive in some aspects than the first version of ZeuS, due to its notable lack of web injection functionality. A fascinating 2014 piece by PhishLabs provides an intimate perspective on the early days of Gozi:

> Nikita Kuzmin, a 25-year-old Russian national [..] worked on coding spyware and Remote Access Trojans (RATs). He borrowed source code from existing families popular at the time [..] UrSnif (developed by Alexey Ivanov, "subbsta"), and botnet C2/management functions and backend code from Nuclear Grabber [..] Kuzmin worked closely with the [malware author] superstars of those days: Corpse, Vladislav Horohorin (BadB), the Vasiliy Gorshkov and Alexey Ivanov team (Suidroot, Eliga, XTZ, Skylack, Kotenok). He had known some of them since the ShadowCrew days. He was younger than most of his peers, at the time posting that he was looking forward to getting his [motorcycle license] and hoping to soon be making enough money for a brand-new, "real" motorcycle [..] Despite his young age, he was trusted, respected for his practical technical skills and coding talent, and also known for his enthusiasm for the idea that Internet fraud, especially against Western targets, was a legitimate profession with better pay and perks than working for local computer and software retail outlets, university labs, and ISPs. [..] Kuzmin had access to the source code for several crimeware kits with overlapping state-of-the-art capabilities, each kit doing something exceptionally clever in one key area compared to the others. [He and the HangUP team] created a repository under version control for a crimeware kit codebase incorporating all of these best features – this is what became known as Gozi. The HangUP team was a nationalistic group with a common following of "cyberfacism," shared Russian and Nazi imagery, and an overarching theme to wage financial warfare on Western interests through the use of the Internet to commit fraud.

For its first year, Gozi operated undetected; It was a 2007 expose by SecureWorks which brought this strain of malware to public attention, complete with a rundown of its internal composition and of the shape of the underlying financial operation. From its humble beginnings, Gozi — Similarly to Emotet — grew into a multi-module, multi-purpose malicious platform, and many of the modern derivatives of Kuzmin's original work are still being actively used in malicious campaigns as of 2020. That's 14 years of activity — several times the average lifespan of a malware brand.

This is not a coincidence. In all probability, the longevity of Gozi can be traced back to a single unfortunate incident.

For three more years past its 2007 public debut, Gozi was a classic malicious campaign — a single codebase kept to a tightly closed group of cybercriminals. Then, in 2010, the sources for this first version of Gozi leaked (this version was called Gozi CRM — that stands for "Customer Relationship Management", namely the management of your banking credentials into a state where cybercriminals possess them). Other actors took the code and ran with it, creating two new versions: Gozi Prinimalka (which went on to merge with Pony and become Neverquest) and Gozi "ISFB" (the meaning of which has apparently been lost to time). These early mutations alone already disrupted the industry's ability to keep track of Gozi. One vendor dubbed ISFB as "Gozi2"; others called it "Ursnif" or "Snifula", after that piece of early-2000s spyware that the original Gozi CRM borrowed code from. Several other vendors started referring to the malware as "Rovnix", after a packer that was commonly used to obfuscate its binaries.

We know that "ISFB" is the proper name for this derivative of Gozi, as the internal binary strings contain references to the "ISFB project". The original act of dubbing the malware "Ursnif" tied back to a long and proud tradition of arbitrary malware-naming that went back at least to the Michelangelo Virus of 1991, thus named by onlookers because it would trigger on a specific date that was coincidentally that artist's birthday. There's nothing inherently wrong with that, but the many monikers and misconceptions surrounding the fledgling Gozi family tree sowed confusion and created fractures in the sphere of knowledge surrounding the malware. "Rovnix" and "Snifula" thankfully fell out of fashion in the nomenclature, but "Ursnif" stuck. In fact, it stuck so well that even in 2020, a full two decades past the publication of Ivanov's original tool, the name "Ursnif" — artifact of a long-bygone era — is still often used as a collective synonym for modern incarnations of Gozi.

Several years later, the source code of ISFB leaked. Sources are conflicted on the exact timing here; most evidence places this second leak in 2015, but some sources claim it happened as early as 2013 (this confusion may be due to the existence of *several* leaks that happened some time apart from each other). One of the resulting branches went on to merge with Nymaim, where it was used as the core code for the two malware families' hybrid child, GozNym. Yet another branch became Dreambot, which heavily relied on code from the original 2010 CRM leak, tweaked ISFB's check-in format and added support for C&C communication over TOR network. Another branch spawned many successful campaigns but did not attain a common name, leaving us no choice but to name it ourselves – "Goziat", after its operators' affinity for the Austrian .at top level domain (the name beat out "Schnitzel Gozi", but just barely).

When second-wave Gozi had been around for long enough, some actors felt that the market was ripe for a new major version — leading to the birth of Goziv3 (RM3 loader), ISFB3, and Gozi2RM3(IAP 2.0). Each of these introduced its own tweaks to the malware's obfuscation mechanism, control flow and C&C communication scheme. In particular, these "third-wave Gozi" campaigns boasted new features such as signed binaries, HTTPS communication and a tiered 2-stage client registration process (which we'll elaborate on below).

By this point it should become apparent to you that these people never bothered to set up a proper regulatory body to handle naming for the various branches of Gozi. Given the considerable migraine that we experienced internalizing all the above, maybe this is less of a bug and more of a feature.

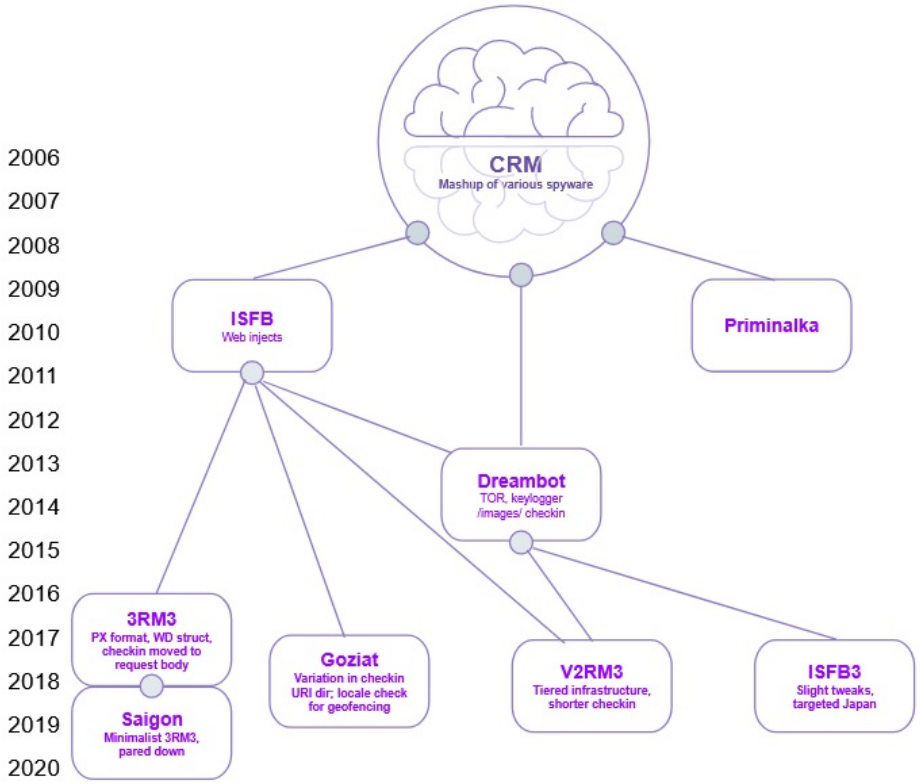


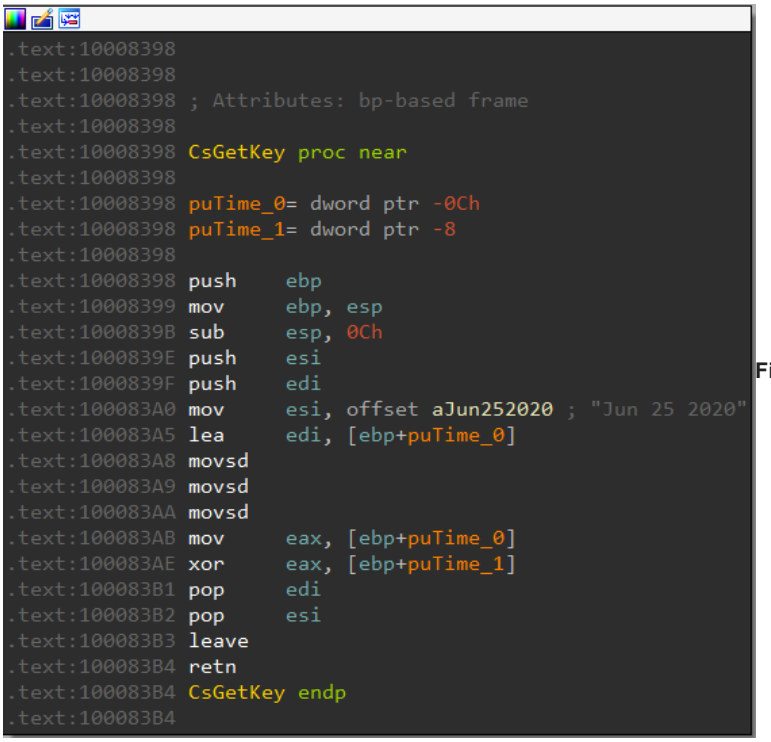
Figure 0: Timeline and influences of Gozi variants.

The Core Gozi Experience

We've emphasized the divergent evolution of Gozi so much that you might have come to expect each variant to do its own thing, and for the malware behavior to be so different from one variant to the next that their common origin is barely noticeable. In fact, the opposite is true: if you're not familiar with Gozi's garden of forking strains, it's pretty easy to mistake one variant for the other, especially when doing black-box behavioral analysis.

Below we list some behaviors which are common to most, if not all, Gozi strains.

> Strings are contained, in encrypted form, in the binary's .bss section. The decryption process makes use of a "key" string which is a string-formatted compilation timestamp (e.g. April 20 2019).



```
.text:10008398
.text:10008398
.text:10008398 ; Attributes: bp-based frame
.text:10008398
.text:10008398 CsGetKey proc near
.text:10008398
.text:10008398 puTime_0= dword ptr -0Ch
.text:10008398 puTime_1= dword ptr -8
.text:10008398
.text:10008398 push    ebp
.text:10008399 mov     ebp, esp
.text:1000839B sub     esp, 0Ch
.text:1000839E push    esi
.text:1000839F push    edi
.text:100083A0 mov     esi, offset aJun252020 ; "Jun 25 2020"
.text:100083A5 lea     edi, [ebp+puTime_0]
.text:100083A8 movsd
.text:100083A9 movsd
.text:100083AA movsd
.text:100083AB mov     eax, [ebp+puTime_0]
.text:100083AE xor     eax, [ebp+puTime_1]
.text:100083B1 pop     edi
.text:100083B2 pop     esi
.text:100083B3 leave
.text:100083B4 retn
.text:100083B4 CsGetKey endp
.text:100083B4
```

Figure 1: String Decryption Process

- A man-in-the-browser attack steals the victim's credentials for a list of pre-configured websites (typically banks, but this is set at the campaign level and is not a feature of the malware per se). In ISFB and its derivatives, web injects were introduced — these modify websites to include input fields that weren't there before, such as bank PIN number, in order to entice the victim into providing that information.
- A specific C&C check-in format, which appears either in the request header or body. A typical example is soft=%u&version=%u&user=%08x%08x%08x%08x&server=%u&id=%u&crc=%x, though there is slight variation in the parameters used and their order.

  An elaborate scheme used to obfuscate the C&C check-in. This in particular is effectively a constant among variants — we've seen no tweaks introduced to it anywhere. The obfuscation works as follows:
  - Encrypt the check-in using a built-in symmetric key (earlier variants used RC6, newer variants use Serpent; both are niche encryption schemes)
  - Encode using base64 encoding
  - Escape non-alphanumeric characters with the homebrew escape character "_". For example, "+" becomes _2B.
  - Randomly intersperse the result with slash characters ("/") .

```
CHAR *__stdcall ObfuscateParamStr(const CHAR *SourceStr, int pKey)
{
  CHAR *DestStr; // edi
  char *fake_param; // eax
  char *_fake_param; // ebx
  int SourceStr_len; // eax
  char *NewStr; // eax
  CHAR *_NewStr; // esi
  LPCSTR _DestStr; // esi
  char *__DestStr; // esi
  const CHAR *v11; // [esp+0h] [ebp-10h]
  int fake_param_len; // [esp+Ch] [ebp-4h]

  DestStr = 0;
  fake_param = GenScriptLine(v11);
  _fake_param = fake_param;
  if ( fake_param )
  {
    fake_param_len = lstrlenA(fake_param);
    SourceStr_len = lstrlenA(SourceStr);
    NewStr = ig_heapAlloc(fake_param_len + SourceStr_len + 1);
    _NewStr = NewStr;
    if ( NewStr )
    {
      strcpy(NewStr, _fake_param);
      lstrcatA(_NewStr, SourceStr);
      DestStr = SerpentEncryptStringToB64(_NewStr, pKey);
      ig_HeapFree(_NewStr);
      StrTrimA(DestStr, "\r\n=");
      _DestStr = UnescapeSpecialBytes(DestStr);
      if ( _DestStr )
      {
        ig_HeapFree(DestStr);
        DestStr = _DestStr;
      }
      __DestStr = PutRandomSlashes(DestStr);
```

Figure 2: Obfuscation Scheme

- A separation at the process level between the malware module that performs the man-in-the-browser attack and the module that makes the actual decisions as to what to inject and where, where the latter is injected into explorer.exe. The two processes communicate using a named pipe, though some of the malware's run-time information is kept globally accessible in the registry (for instance, the CRC value of the last task list received from the C&C server).
- A gamut of standard information stealing features such as key-logging, e-mail, ftp accounts, IM data and certificate grabbing, as well as screen video capture. These are augmented with support for optional DLL-format plugins, which the C&C server can instruct the infected machine to download and execute at runtime.
- Use of a rather cumbersome format, called "joined resources", for various kinds of hardcoded information. The type of hardcoded information isn't written in plain English, either, and is denoted by a CRC32 tag. These have been known to vary.

CRC32 TAG	Readable Name
0x556aed8f	server
0xea9ea760	bootstrap
0xacf9fc81	screenshot
0x602c2c26	keyloglist
0x656b798a	botnet
0xacc79a02	knockertimeout
0x955879a6	sendtimeout
0x31277bd5	tasktimeout
0x18a632bb	configfailtimeout
0xd7a003c9	configtimeout
0x4fa8693e	key
0xd0665bf6	domains
0x75e6145c	domains
0x6de85128	bctimeout
0xefc574ae	dga_seed
0xcd850e68	dga_crc
0x73177345	dga_base_url
0x11271c7f	timer
0x584e5925	timer
0x48295783	timer
0xdf351e24	tor32_dll
0x4b214f54	tor64_dll
0x510f22d2	tor_domains
0xdf2e7488	dga_season
0xc61efa7a	dga_tld
0xec99df2e	ip_service

Figure 3: Sample map of CRC to resource name (credit: Maciej Kotowicz)

Variants use the same elements and the same format for storing web injects

- **@ID@** -> bot id (victim host identity)
- **@GROUP@** -> group id (group id of the bot)
- **@RANDSTR@** -> random string
- @ -> targeted financial institutions
- @ -> configuration
- @ -> video to record once the victim visit the page of interest
- @ -> connect SOCKS server
- @ -> connect VNC

Spotlight on Choice Gozi Strains

Goziat

This flavor of Gozi appears to have debuted several years following the ISFB leak. Its most distinguishing feature compared to other variants is a quirk in its C&C check-in; most other Gozi versions emulate an HTTP get request for an image, and therefore end with some variation of /images/ followed by a long and unwieldy BASE64 blob that's topped off by a .gif or .jpeg file extension. This obfuscation is somewhat tricky for security solutions to deal with, but not impossible — which is probably why goziat instead uses a different resource directory instead of "images", which can be configured during the malware's build process. Since goziat does not bother with the ruse of its check-in being a legitimate request for an image, it also abandons the "image file extensions as encoded requests" trick, as popularized by Dreambot and replicated by many other variants. Instead, it uses a straightforward action=<action> request format.

A small number of researchers dubbed this variant LOLSnif, after its "Living Off the Land" (making use of pre-installed Windows utilities such as mshta.exe and powershell.exe). Instead of the more state-of-the-art geofencing on the server side, this variant has a built-in client-side check for the System default UI language.

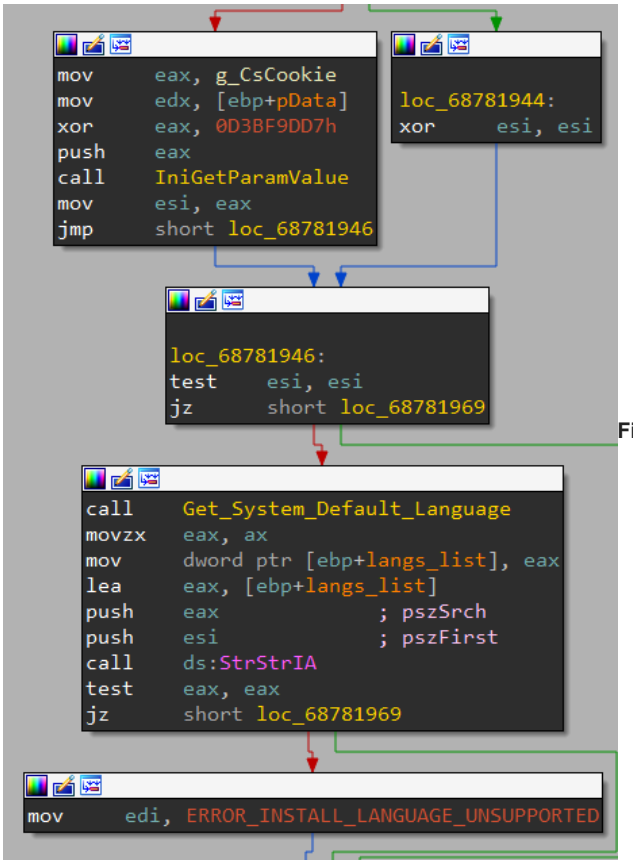


Figure 4: System UI language check used for geofencing

While there appear to be several different campaigns making use of Goziat, these tend to use domains with an .at top-level domain (hence the name we chose for this variant). This, along with some overlap in IP addresses and even the subdomains being used, collectively hint at some commonality at the campaign operation level. These campaigns tend to hang on to the same domains and IP addresses for a relatively long time, which may not be the best choice opsec-wise. The following table lists some of the campaigns we and other vendors have been able to track that have made use of this specific version of Gozi.

Group ID	Serpent Key(s)	Resource Directory	Target	Domain examples /notes	General Notes

100020003000	W7fx3j0lFvOxT2kFU7yKaYwFde7YtppY	api1	Italy	pipen.at Laurela.atcalag.at	Spam campaign that disseminated password-protected archives hosted on Google drive and firefox send. The archives (usually with the password 7777) contained a heavily obfuscated JavaScript file named presentation_????.js, which in turn dropped a DLL launched via a regsvr32.exe -s command. First-stage domains live for a long time, but lately the serpent key changes frequently.
1000	F1cl1tAcbPsStUtM	rpc	USAEurope India Russia	dicin.at kartop.at	Apparently defunct since 2018, possibly before. Distributed by malspam, as well as the GrandSoft exploit kit. A certain common domain suggests that the actor behind this campaign may have also experimented with deploying DreamBot to attack targets in Japan.
1000	K2u7G0lE4u1VoS0V Nf6lU8d5X0i1Wr7V	wpapi	USA Canada Italy	evama.at mobipot.at	Seems to have taken off around March 2018. Crafted spam apparently sent by DHL, used to lure victims to Grandsoft Exploit Kit landing page.
20001500	Gwe9HMygngWe8kPKTEJopj7WLDojJKx4	webstore		sorna.at Rivier.atexplik.at	Started in the end of 2018. Dropped by Hancitor, which switched to distribution also via COVID-19 themed emails in early 2020. Based on shared domains, this actor seems to have also run some campaigns using Hancitor loader, EvilPony and CobaltStrike.
40005000	Ni8wR0zp1Ak5FoOW Xio4U7r3MlO7FwcQ	wpx	Italy	deepmoler.cn eromov.at	Seems to have started in 2018; distributed via QuantLoader, GrandSoft exploit kit and Hancitor malspam.

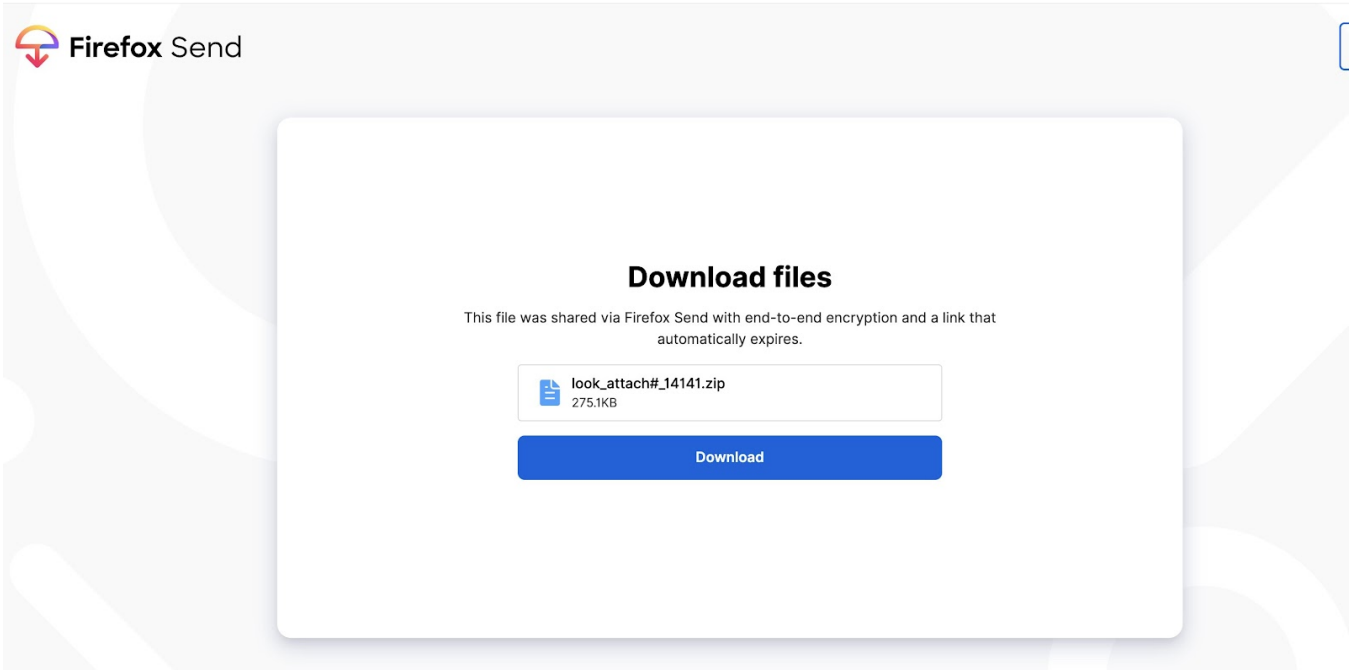


Figure 5: malicious file hosted at Firefox Send
Apart from the domains listed above, each campaign also supported TOR domains.

- **Api1**: 6buzj3jmnvrak4lh.onion, g4xp7aanksu6qgci.onion, l35sr5h5jl7xrh2q.onion
- **Rpc**: v6ekxns6ldq5uai3.onion, uaoyiluezum43ect.onion, tjiqtwzewnkbqxmh.onion
- **Wpapi**: 4fsq3wnmms6xqybt.onion, em2eddryi6ptkcnh.onion, nap7zb4gtnzwmxsv.onion, t7yz3cihrrzalznq.onion
- **Webstore**: vo5vuw5tdkqetax4.onion , zq4aggr2i6hmklgd.onion
- **Wpx**: pzgxy5elkuywloqc.onion, q7nxkpgras35dwxk.onion, rbhqdxwdwrlp67g6.onion, jesteoq7glp3cpkf.onion

Gozi2RM3 / Gozi IAP2.0

This version of Gozi doesn't introduce any C&C communication tweaks (as opposed to other third-wave Gozi, which we will discuss later); based on the communication and the binary metadata, one might easily mistake it for other version 2 variants. The most salient difference between Gozi2RM3 and earlier-generation Gozi is at the campaign infrastructure level, which implements a thorough vetting process.

The C&C infrastructure of a Gozi2RM3 campaign is subdivided into 2 stages, where the address of a stage-1 C&C is hardcoded into the initial binary that infects the victim. The stage-1 C&C is pre-configured with an ISP/geolocation deny list which is used to pre-filter connections suspected to be curious researchers rather than true victims. In some cases, even when these checks are passed, the victim must prove themselves a good victim by continually sending data; only then the second-stage server will consider pushing the campaign's main payload and sending the true configuration.

This devious design choice is a deserved entry on our List of Reasons We Thank The Lord Every Day that Malware Authors Never Learn from Each Other's Achievements. Thankfully, there is at least a default serpent encryption key for C2 communications (10291029JSJUYNHG) that many campaigns never bothered to modify.

While these differences in infrastructure are the main ones, there are also some differences in functionality. There's the C&C panel, which makes us believe that this variant is probably up for sale somewhere.

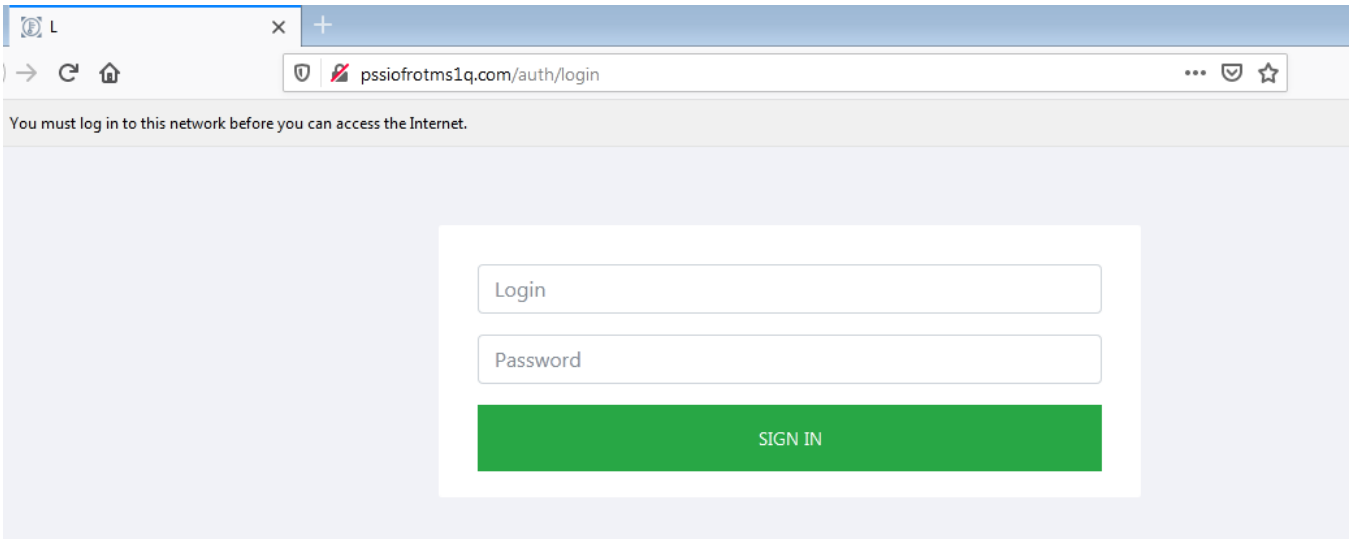


Figure 6: Gozi2RM3 C&C web panel

There's also a more terse URI format string (no *os*, *size*, *hash*). The protocol for specifying what resource is requested from the C&C is the "feign request for image" protocol, taken directly from dreambot.

Control Mask	Request Type	Comment
/t*php	get new task	Used until Sep 2015
/c*php	get new config	Used until Sep 2015
/d*php	send stolen data	Used until Sep 2015
/images/*.gif	get new task	current format
/images/*.jpeg	get new config	current format
/images/*.bmp	send stolen data	current format
/images/*/.avi	download 2nd stage dll	not every c&c

Figure 7: Bogus image requests and their meaning; used in Dreambot and some other variants (credit: Maciej Kotowicz)

There appears to be a connection between Goziat actors and Gozi2RM3 actors. Both variants changed the way they decrypt payloads by adding another step of encryption — the malicious server's public RSA key is further encrypted with a serpent key, which itself is only assembled at runtime. The fact both these variants changed their flow in the same way, and in such a short time-frame, seems to imply that they either voluntarily share code or one of the groups is really vigilant about imitating the other.

The following table lists some of the campaigns we have been able to track that have made use of this specific version of Gozi. This is just a shortlist of recently-alive campaigns among a very long list of long-dead ones. Some commonalities between the campaigns suggest that the number of active actors using Gozi2RM3 might be smaller than appears at first sight.

Group ID	Serpent Key(s)	Target	Domain examples /notes	General Notes
3xxx (varies)	10291029JSJUYNHG	USA Canada Germany, Australia, Italy	TLD of drop site is .cab	Campaign consistently used first-stage droppers named info_date.doc, distributed using reply-chain hijacking. Tenuous connection to Valak malware via use of the same dropper.
44444343 205x	10291029JSJUYUON 21291029JSJUXMPP	Italy	bizzznez.com Consaltin.orgredflash.org	Distributed via standard malspam (also seen by Cutwail), at some times COVID-19 themed or pretending to be from DHL. Focused specifically on stealing credentials of Italian bank customers.
89897979989893938182	10291029JSJUYNHG	Italy	Sub domains gstat[.] and line[.]	Distributed via Excel files with hidden sheets. connected to 3xxx campaign. Probably the same actor.
5555400040105600	78347829JSDUKLHG	USA Japan	TLD is .today, .website and .space	Distributed through exploit kits (RIGEK, GrandSoftEK, FallOutEK) Seems to have taken off around November 2019
416x	10291029JSJUYNHG	Czech, USA, Italy	*lokoloppo4.com, 38.132.124.193*	Seen distributed through spam and docs with macros. For example, doc impersonating to be Hubata Cernoska (big shop in Czech). The web injection configuration is very similar to 21291029JSJUXMPP.

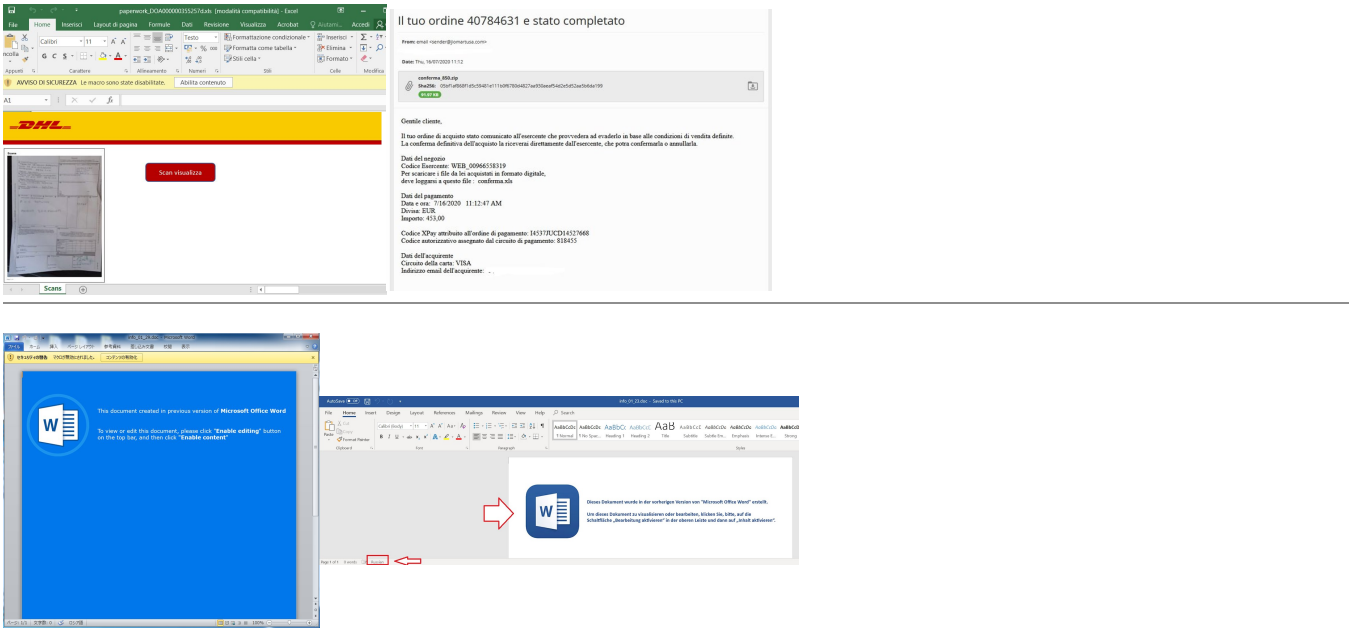


Figure 8: Assortment of malicious documents used to spread Gozi2RM3

For educational purposes, we include the configuration for intercepted and hidden URLs used by one of the campaigns (group id 4444, serpent key 21291029JSJUXMPP) in **Appendix B** .

Goziv3 RM3

This variant has been in the wild since at least the summer of 2017. A lot of the code from ISFB is still there, but a lot is also different. The group behind this variant is pretty sophisticated and makes a decent effort to stay under the radar while attacking mostly Australia, Italy and the USA.

There are some salient technical differences between this variant and second-wave Gozi:

1. RM3 loader uses a unique file format that is called “PX format”. Every dll is loaded using a homebrew loader for this format. A tool for processing PX format files can be found here.

1. The Joined Resources (“JJ”) struct has been discarded in favor of a “WD” struct which has a different format. Also, the struct is not kept right after the PE headers as previously, but instead in the security directory.

```
00000000 WD_struct       struc ; (sizeof=0x18, ma
00000000 size            dd ?
00000004 magic           dw ?
00000006 flags           dw ?
00000008 inner_WD_struct In_WD_Struct ?
00000018 WD_struct       ends
00000018
00000000 ; ------------------------------------
00000000
00000000 In_WD_Struct    struc ; (sizeof=0x10, ma
00000000
00000000 xor_key         dd ?
00000004 crc32_name      dd ?
00000008 size            dd ?
0000000C addr            dd ?
00000010 In_WD_Struct    ends
```

Figure 9: Goziv3 RM3 WD struct

1. This variant makes use of a word list that is kept in one of the WD structs for pseudorandom generation of registry key names. This makes detection of this variant by registry key IOCs more difficult than that of previous variants.

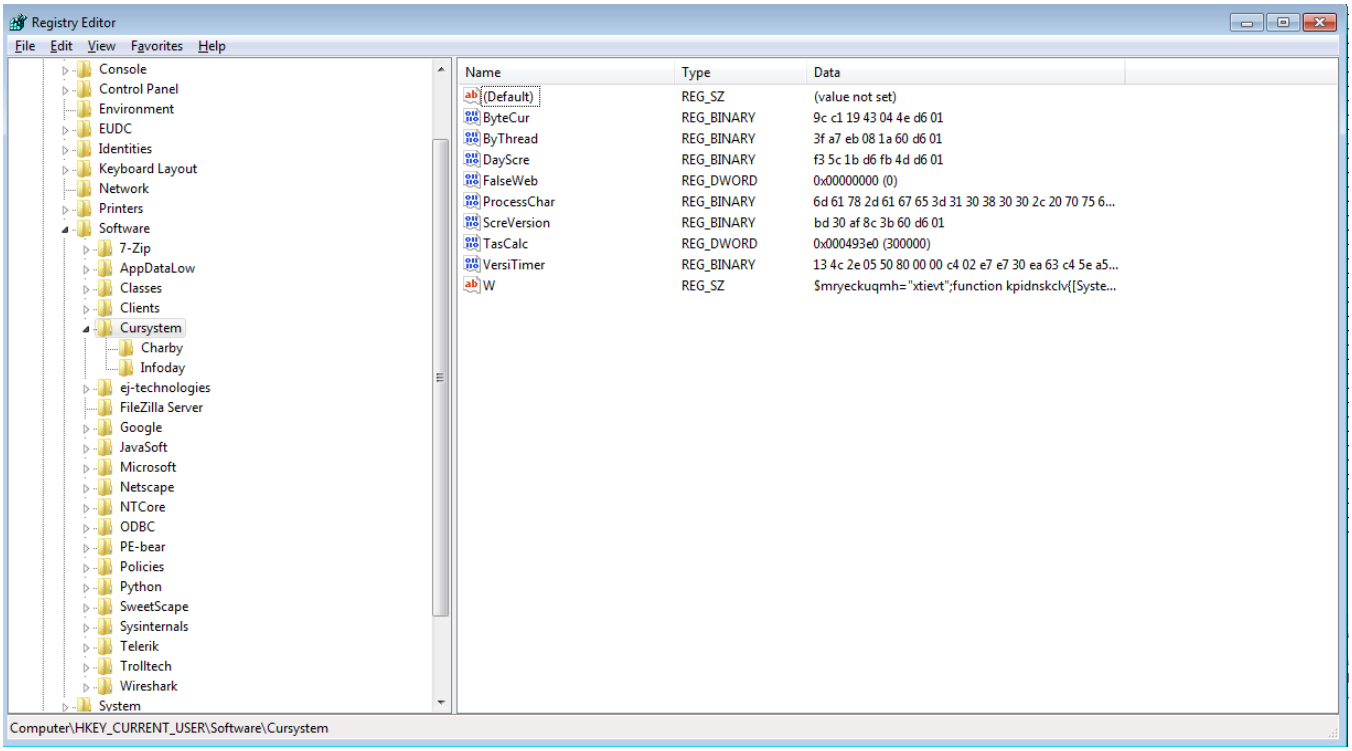


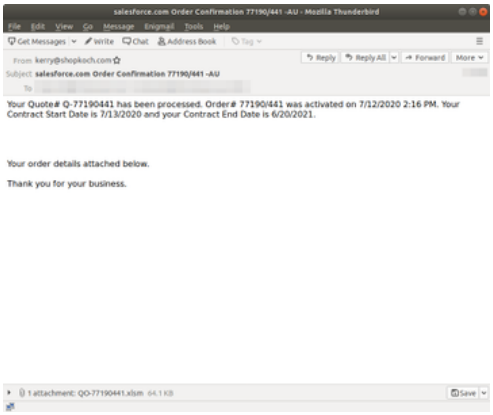
Figure 10: Goziv3 RM3 data kept in randomly-named registry entries

1. This variant uses forfiles.exe in order to execute a powershell script that loads shellcode into memory and then executes that shellcode using APC injection. By adding the forfiles executable during the infection chain, this variant can evade detection mechanisms which search for persistence of more known script engines such as PowerShell and mshta.

1. Probably the difference that will jump out at an analyst first is the tweak to the communication method. While the obfuscation scheme for the C&C check-in remains the same one globally used in Gozi variants, in this variant it is put in the request body instead of the URI (which becomes just “index.htm”). Also, the plain check-in has a slightly different format:

```
{rand1}={rand2}&type={type}&soft={soft}&version={version}&user={user}&group={group}&id={id}&arc={arc}&crc={crc}&uptime={uptime}
```

While these technical differences do exist, as with Gozi2 RM3, the group behind this variant seem to have focused their innovation on ways to prevent researchers from interacting with the C2 and obtaining payloads. They do so by restricting the payload delivery on the server side – when clients try to connect to a C2 server, they are geolocated and rejected if the location does not match the targeted region of the current campaign. More importantly, the first-stage C2 stays online for a very short period of time — just long enough to serve a large number of victims and disappear before being subjected to unwelcome analysis. Gozi3 RM3 is usually distributed by spam emails that carry an obfuscated VGS file, or an xls4.0 macro, as an attachment. At least since 2017, This variant of Gozi has been known to be bundled with a popular loader also used by some Emotet and Dridex campaigns.



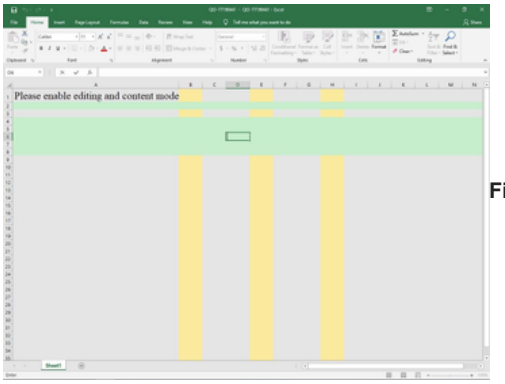


Figure 11: Goziv3 RM3 malspam and attached maldoc

The top-level domain of the C&C servers is typically .xyz, and the samples are typically signed with Verisign (in practice, this means that when a victim runs these samples they have to click 'OK' on one less dialogue box that warns them that maybe they shouldn't execute random files handed to them by strangers on the internet).

The people behind this operation seem to value their work-life balance — the campaigns are somewhat far-between, and are inactive on weekends. You can read here about a possible connection to the "Evil Corp" group.The lack of variance in all the above characteristics seem to suggest that this Gozi variant is mainly in use by a single actor.

Legacy Gozi

These are Gozi variants that seem to have fallen out of use. The groups pushing them have either moved on to more modern malware, or moved on from the malware business entirely.

Dreambot

This branch of the leaked ISFB sources was first spotted shortly after the ISFB leak (many sources say 2014, some say 2015). It remained in active development for quite a while and added many new features; most important among those were Tor-hosted command and control servers, a keylogging capability, the ability to steal browser cookies and data from email clients, a screenshotting feature, the ability to record a victim's screen and a VNC remote access feature. Dreambot had a Cybercrime-as-a-Service (CaaS) monetary model, and was available to any aspiring cybercriminal for the right price.

Many of these features were unique even among newer variants, but ultimately Dreambot was a product of the second-wave Gozi era. To evade prying eyes, it relied on C2 check-ins imitating a GET request for an image — a trick that drastically lost its value once it became common knowledge, and could not compete with the sophisticated stealth operation best practices of third-wave Gozi such as the signed binaries and tiered C2 model. After a long and fruitful run, in March of 2020 Dreambot seems to have finally gone silent.

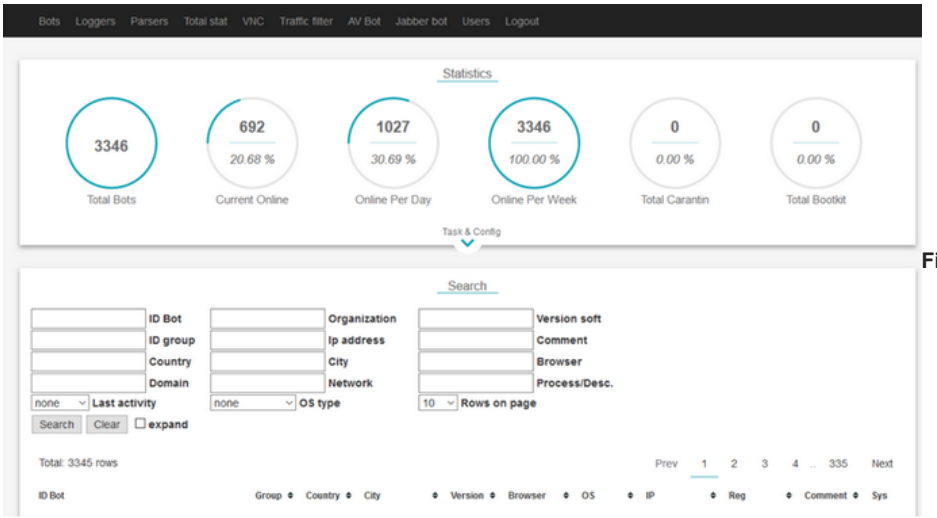


Figure 12: Dreambot C&C web panel

Saigon

This fork of Goziv3 RM3 was identified by FireEye in September 2019. It introduced various changes to base Goziv3 RM3, among them:

- New arguments to the C2 check-in. Noticeable among those was knock, which encoded a number of seconds that the client would wait between making subsequent requests.

- The SERPENT encryption is used in the ECB mode of operation, instead of CBC. In actual cryptography this would be considered a downgrade, but the general rule is that no one ever really runs cryptanalysis on malware communications anyway, and this is just one glorified obfuscation scheme replaced with another.
- No interspersed random slashes when obfuscating the encrypted C2 check-in.
- No use of the PX format.

As can be seen, most of the changes seem to be focused on removing features, making the malware simpler and reverting to something more closely resembling second-wave Gozi.

ISFB3 / Ursnif-A

This variant saw a very limited and precise use, attacking Japanese victims during 2018-2019. Due to a commonality in its distribution method, it has been speculated to be tied to threat actor TA544. As typical for Gozi in general, it was distributed by malspam (specifically through Cutwail) carrying attached office documents, and used geolocation to keep out any requests outside its targets (typically customers of domestic Japanese banks). The spam would often carry the Bebloh downloader, which would only then fetch the actual Gozi.

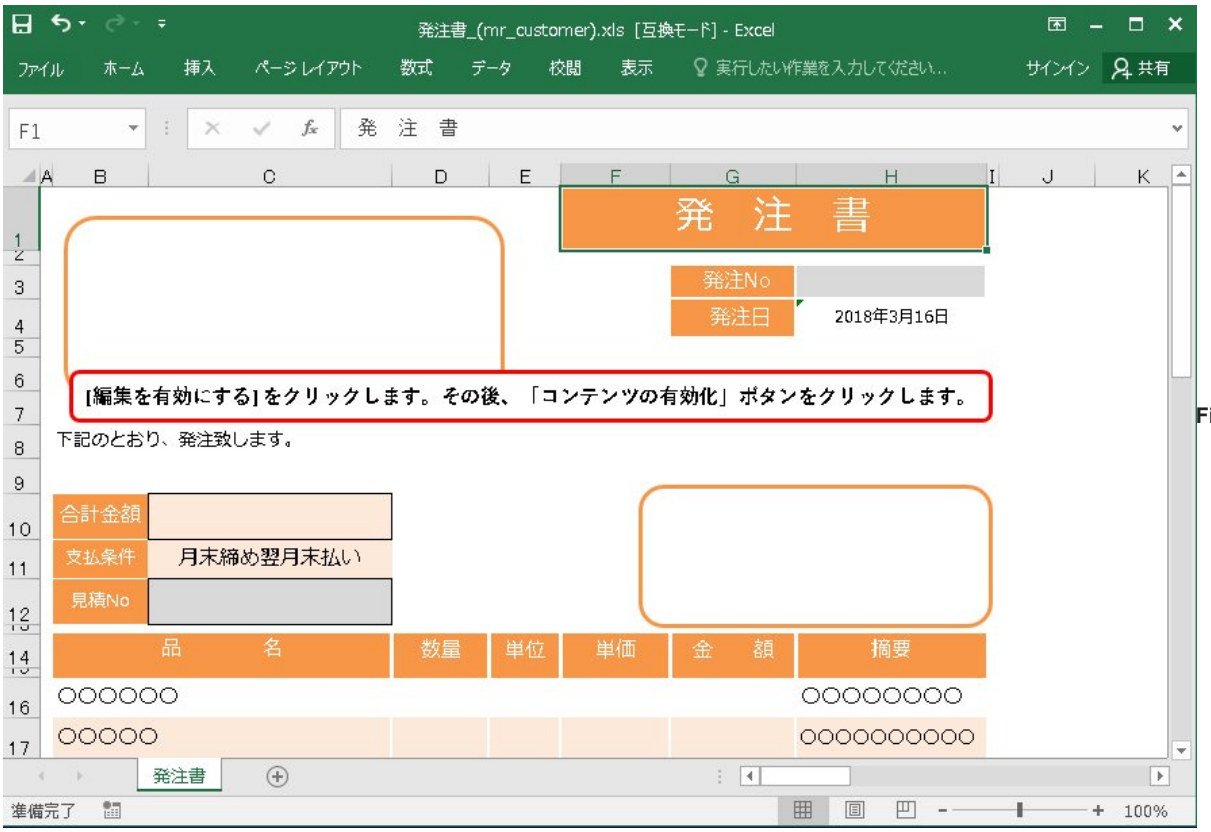


Figure 13: ISFB3 maldoc, targeted at Japanese bank customers

The name “ISFB3” is derived from an explicit pdb path in the malicious binaries — c:\isfb3\x64\Release\client.pdb — but while the digit “3” is in the title, this isn’t third-wave Gozi. The mock image-retrieving GET request is still there, and no ground-breaking features that prevent research of the campaign are present. An array of minor technical differences set this variant apart from other strains: “J1” used as a magic prefix in the joined resources (this is a throwback to early versions of Dreambot), some changes to the operation of its information stealing modules, as well as a few tweaks in its methods for persistence and evasion (see here and here).

Conclusion

When we deal with malware, we often stand on the shoulders of giants. Vendor consensus will quickly tell us the name of the malware we are dealing with, and if it doesn’t, some Google searching for unique-seeming indicators will quickly show us the hard-won insights of researchers who have gone before us and had to deal with the exact same malware. Once we know the malware’s true name, we then have power over it. We can write signatures, hunt for more samples, and use our exacting knowledge to protect potential victims.

Malware with a very fragmented ecosystem, such as Gozi, throws a monkey wrench into all of that. Grant that you’ve gotten hold of a malicious sample or a campaign, and after some quick sleuthing you’ve determined that you are dealing with “Gozi”, or “ISFB” or “Ursnif”, which are all the same but somehow also different. You then have to contend with a list of variants, all referred to interchangeably by those names, all with subtly different communication methods, behavioral patterns and campaign infrastructures. What are you going to do now? For basic triage work, it’s fine to say “this is Gozi, a banker that evolved into a malicious content delivery platform”. But for anything that gets more technical, these subtle differences are a recipe for a headache. If your scope is *all* the Gozi versions, you are going to have a very bad time.

There appears to be no easy fix for this issue. The sun rises in the east, the sea rushes to shore, cybercriminals create new malware and researchers confabulate new names for it. All we can do is produce very delayed reviews, such as this one, that try to inject some order into the overarching chaos. Other than that, we can take solace in the fact that these incidents of multiple-branch malware are few and far between, since cybercriminals have no financial motive to share malicious code, and they usually know better than to make it public. When Gozi-style fragmentation occurs, it's usually because of a leak. So we would like to conclude with a plea to cybercriminals: secure your codebases, for all our sakes!

6. Special Thanks

We would like to deeply thank Benoît Ancel of CSIS Security Group. This article would have not been possible without his detailed technical input on the subject of Gozi's long, branching and confusing history.

IOCs

Goziat

First loader:

cb4f92bf9fef3708e7aeba5d8994a0502952d06374c8a83ff2c1ee0b7e603d35

Main loader:

21a03d9c845e446cb96eba7c93aa6403b8a9aaa744801e77468bf73c0507d028

Main worker:

c486d8579308999b7d9f8cbb6de33b7a3976b9db5b98c06b7744adf5d5d11caf

Gozi2RM3

First loader:

644482a0851ff6f801e1e9afea1eee19e4f4de72b28895de9e3a4e43dd48ff1e

C8456d507da3484d2151f051f439b2de13174be9f72f4f1f03abf52885adca1a

b1ce263fb1945ae153e4c111fdfe0bd3d26cd09381ee21cf82ff5b115ad8ca77

Main loader:

c2ee9cf24f0bddb07914503dbae35c4497d66f9ca01ea65108ef40ff13cbec02

6fa79786607c16fbf5f86d5978ad5c60d32e055946dc6d9d6d78310b1f7cc918

Main worker:

81734690442c224cf104fba0db8bacabdf3dc347bba3da3415a92de587df6d82

b78390654db57946d3d551f9a34a488a726a0e7890746019b7313eddb74a9372

Gozi v3/ RM3

First loader:

41e52cec2091e4451beadad93c5f693d5a008cf56eaf160f9fa4d577b1d707f6

5548507963e58e7c89e406452a4c00823db72b26c96cfcf7ed82799dacaf48d7

Main loader(as PE):

a353dfb1b5eb69808244356cf9a784181c53eea2cb3f254749fa19c307c30cfc

Second stage Dlls (as PEs):

8a2dace2a6fbd650b3e079f20f3886dd595044dbd4cb6bbab16023182f8d17af

495351fb739029a95a2bad9e80da3e6207b9bbd218c5c1d35f7e8eb0a090c04f

979048ab6e71c9589bb204e3017e72dfb8fe8d0b088fbccfa7e0b5791be89c68

87058836bd1c2c7a428ae4a3b4729035dab25795fe4da55b3f5793cc115c611a

Dreambot

52cb2bd9724270b3efe575894112d0a866734856a3257ddcfb24308e42861f6a

ee8a404264b4d3144bc37ef7118da24c77dd15b20d38250badbf53140f7c1d2a

2d5c9af9419bfb08e1066b41f2d2daba93024fb6201ac079e0ec474c424d21ff

0ed9d5dcdc2ea4b90f056071feaec0463c3563eb89b67533741f079a22c7a1e7

f2556099982b2b1ebea7443c863b8b205b6683112624f46393454e430e69aed5

Saigon

8ded07a67e779b3d67f362a9591cce225a7198d2b86ec28bbc3e4ee9249da8a5

431f83b1af8ab7754615adaef11f1d10201edfef4fc525811c2fcda7605b5f2e

628cad1433ba2573f5d9fdc6d6ac2c7bd49a8def34e077dbbbffe31fb6b81dc9

lsfb3

First loader:

cacc1c3af8ad58b992c707bdf36ec1bd5f039dd80780ad2978cb142ccfe714d6

8d7ffebb0774e0dfe9d85f175cd5e1800dfd757bb5fbc4565a8f8a173e739ea5

Client DLL:

f3182febadb7a9cc0d43621bd9909688f9b0f7720d87673970be1c81249e208a

8e707c3afeaff26026153079886 68ff8d2a3be20e5e41f34c6eb33c8deb6ceea

Appendix A

Yara rules

```
rule goziv3: trojan {
    meta:
        module = "goziv3"
    strings:
        $dec_bss = {D3 C0 83 F3 01 89 02 83 C2 04 FF 4C 24 0C}
        $gen_serpent = {33 44 24 04 33 44 24 08 C2 08 00}
    condition:
        ($dec_bss and $gen_serpent) and (uint16(0) == 0x5A4D  or uint16(0) == 0x5850  )
}

rule Gozi_JJ_struct: trojan {
    meta:
        module = "Gozi_JJ_struct"
    strings:
        $jj = "JJ" ascii
        $pe_file = "This program cannot be run in DOS mode" ascii
        $bss = ".bss" ascii
    condition:
         #jj >= 2 and (for all i in (1,2) : (@jj[i] < 0x400 and @jj[i] > 0x200)) and (@jj[2] - @jj[1] == 0x14) and ($pe_file in
(0..1000)) and ($bss in (0..1000))
}
```

Appendix B

Targeted (intercepted) websites

amazon.

paypal.

sella.it

clienti.chebanca.it

ibk.nexi.it/ibk/web

nowbanking.credit-agricole.it

banklinknet2.cariparma.it

banklinknet2.carispezia.it

corporate.friuladria.it

nowbankingcorporate.cariparma.it

nowbankingcorporate.friuladria.it

banking4you.it

fideuramonline.it/script/ServiceLogin/ib/login

paco.cabel.it

icbp.seceti.it/contrattobpercbi

cbi.bpergroup.net/ibk/

scrigno.popso.it/

idp-bpiol.poste.it

banking.bnl.it

unicredit.it/it/privati.html

unicredit.it/wps/myportal/retail

online-smallbusiness.unicredit.it

digital.mps.it/pri/login/

digital.mps.it/pri/pr/

icb.mps.it

icb.mps.it

aziendaonline.mps.it

intesasanpaolo.com

homebanking.bpergroup.net

inbiz.intesasanpaolo.com

carigeonline.gruppocarige.it/wps

gruppocarige.it/wps

ibbweb.tecmarket.it/

ib.mps.it

ib.cbibanking.it

bancagenerali.it/pib

qweb.quercia.com/deutschebank

dbonline.italy.db.com/portalserver/dbiPortal/

poste.it

ubibanca.com

finecobank.com

ihb.cedacri.it*

bancopostaimpresaonline.poste.it

business.bnl.it

creval.it/bancaperta

csebo.it/webcontoc

inbank.it

core.cedacri.it*

banking-imprese.credem.it

mybanking.credem.it

credem.it

relaxbanking.it

Blocked websites

sucmetrics.unicredit.it/

dc.services.visualstudio.com/

.gdatasecurity.de/

.kaspersky.

*lo.v.liveperson.net/api/js/*IwZTMyMDUwZGJkNzk3NThl*

*lo.v.liveperson.net/api/js/*VhMjk2MGQxMjlhM2FjYjc4*

nowbanking.credit-agricole.it/API/Core/erro

.banking4you.it/mobile

.bancagenerali.it/mobile

my.unipolbanca.it/hb/RUEI/

cdn.chebanca.net/js/afp_obf.j

webchat.credem.it/

secure.credem.it/

*.credem.it/*supporto-e-sicurezz*

*.credem.it/*contatti*

m.credem.it/

cache.inbank.it/

cdn.inbank.it/

col.eum-appdynamics.com/eumcollector/beacons/browser/

m.intesasanpaolo.com/

.relaxbanking.it/relaxbanking/sso.LoginMobil

m.unicredit.it/

m.mail.tim.it/

fideuramonline.it/script/ServiceLogin/ib/login