`Regin` en famille, une enquête

Guillaume Bonfante Jean-Yves Marion Benjamin Rouxel
Fabrice Sabatier Nicolas Schermann

Université de Lorraine, CNRS and INRIA - LORIA

Abstract

Le 24 novembre 2014 Symantec [Sym] et Kaspersky [Kasa] dévoilent sur leurs sites respectifs l'existence d'un nouveau malware "hyper sophistiqué" dénommé `Regin`. Sa technicité lui donne de bonnes propriétés de furtivité. Et il faut bien avouer que `Regin` a fonctionné de nombreuses années sans alertes divulguées.

Le même jour, le magazine Américain "The Intercept" publie également sur son site [ThI] des échantillons de `Regin` récoltés auprès de différentes sources de la communauté en sécurité. La première question que l'on s'est posée sur ces échantillons a été de savoir si le code de ce nouveau malware faisait partie de la plate forme "tilde" mise en lumière par Kaspersky (ce framework qui a servi à développer `Stuxnet`, `Duqu`, `Flame` et `Gauss`). Pour répondre à cette question, nous avons utilisé la technologie de détection morphologique développée dans notre laboratoire. La réponse est claire et sans ambiguité : il n'y a pas de code commun entre la famille de `Stuxnet` et les échantillons de `Regin`. Essayons autre chose. Voyons s'il y a un lien entre les différents échantillons de loaders de `Regin`. Par exemple, pour le loader `Loader-fd92fd`, on lit

```
DIST: "/The\_intercept-regin/Loader-fd92fd":
100.00% (619/619): "Loader-fd92fd"
97.76% (612/626): "Loader-2c8b9d"
96.43% (594/616): "Loader-4b6b86"
82.88% (484/584): "Loader-26297d", "Loader-bfbe8c"
78.24% (604/772): "Loader-744c07"
52.04% (600/1153): "Loader-225e9596"
39.12% (602/1539): "Loader-47d0e8"
31.05% (154/496): "Loader-b26989", "Loader-b505d6", "Loader-ecd7de33"
```

```
29.88% (599/2005): "Loader-01c2f3"
27.18% (137/504): "Loader-a0e3c52a", "Loader-cca18507"
17.20% (274/1593): "Loader-20831e82", "Loader-7553d4a5"
15.72% (69/439): "Loader-1c024e", "Loader-5c81cf82", "Loader-ba7bb6"
9.44% (37/392): "Loader-d240f0"
9.25% (37/400): "Loader-6662c3"
9.18% (37/403): "Loader-b29ca4"
9.18% (37/403): "Loader-a6603f27"
8.51% (273/3207): "Loader-f89549fc"
8.35% (37/443): "Loader-ffb0b9"
6.44% (25/388): "Loader-187044"
6.31% (25/396): "Loader-06665b"
3.79% (51/1346): "Loader-d42300fe"
```

Comment lire la sortie ? Prenons par exemple la ligne

$$52.04\%(600/1153) : \text{"}\texttt{Loader} - \texttt{225e9596}\text{"}.$$

Elle indique que le loader `Loader-225e9596` contient 1153 sites et que sur ces 1153 sites, 600 sont communs avec ceux de `Loader-fd92fd`. Et pour ceux qui suivent, la première ligne montre que sur les 619 sites de `Loader-fd92fd`, 619 sont dans `Loader-fd92fd` ! Allons plus loin, on distingue des binaires très proches notamment `Loader-fd92fd`, `Loader-2c8b9d` et `Loader-4b6b86` et d'autres qui ont la même partie de code en commun (`Loader-b26989`, `Loader-b505d6`, `Loader-ecd7de33`). Mais l'analyse montre déjà que certains loaders sont bien différents.

1 De `Regin` à `Qwerty`

Le 17 janvier, le lien entre Regin et l'alliance des "five eyes" nous saute aux yeux. Effectivement de nouvelles révélations sur l'affaire de Snowden sont publiées par le journal allemand Der Spiegel [Spi]. On apprend que l'alliance des 5 disposent d'un keylogger dénommé `Qwerty` composé d'un driver et de deux DLLs. Après extraction des code binaires, nous réitérons l'expérience sur `Stuxnet`, `Duqu`, `Flame` et `Gauss`. Une fois de plus la corrélation n'est pas avérée. Pourquoi pas `Regin` ? On se crée une petite base qui contient les sites des deux exemplaires de `Regin` à notre disposition :

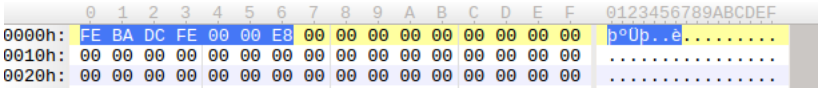
```
./sigtool --learn-sub -f raw -r alo regin.db The_intercept-regin/Orchestrator*
LEARN_OK: "The_intercept-regin/Orchestrator-e420d0cf", 23622 sites/49027 nodes
LEARN_OK: "The_intercept-regin/Orchestrators-41391495", 23390 sites/46489 nodes
```

Apprentissage réussi, les deux versions d'`Orchestrator` ont respectivement 23622 et 23390 sites. C'est beaucoup, on peut s'attendre à des résultats de recherche pertinents. Passons justement à l'étape de recherche, avec les deux exemplaires de `Qwerty` :

```
./sigtool --dist-sub -r alo regin.db QwertyKM/qwerty/
DIST: "QwertyKM/qwerty/20120.dll.bin":
    100.00% (118/118), 0.50% (118/23622): "Orchestrator-e420d0cf"
    100.00% (118/118), 0.50% (118/23390): "Orchestrators-41391495"
DIST: "QwertyKM/qwerty/20121.dll.bin":
    100.00% (98/98), 0.41% (98/23622): "Orchestrator-e420d0cf"
    100.00% (98/98), 0.42% (98/23390): "Orchestrators-41391495"
```

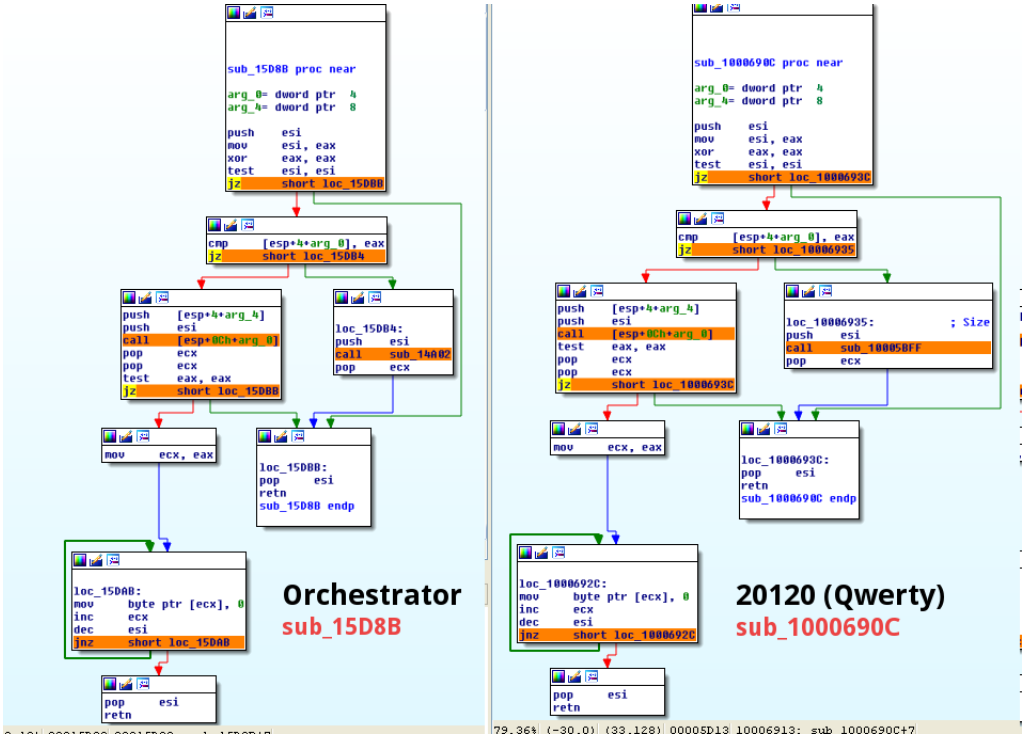
Bingo ! On retrouve bien le code de `Qwerty` dans le dispatcher de `Regin`. Dans un cas, on trouve 118 sites, dans l'autre 98. À la différence des révélations de Kaspersky du 27 janvier [Kasb], ce n'est pas une correspondance avec un plugin de `Regin` (le 50251 Keyboard driver hooking) qui est établi ici mais belle et bien avec le "core" de `Regin` (Stage 4 (32-bit) / 3 (64-bit) – dispatcher module, "disp.dll")

Note pour ceux qui veulent reproduire l'expérience. Les échantillons du "dispatcher" (Ochestrator) ne sont pas des fichiers exécutables mais des dump de RAM, ainsi qu'il est précisé sur le site de "The Intercept": "The malware zeroes out its PE (Portable Executable, the Windows executable format) headers in memory, replacing "MZ" with its own magic marker 0xfedcbafe.".



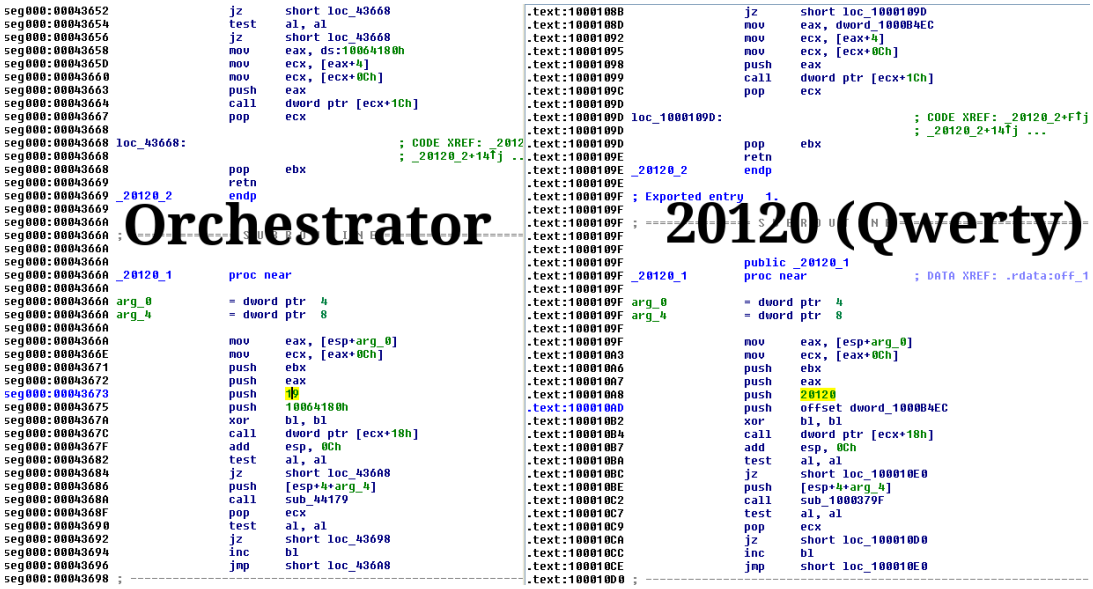
2 Les liens de parenté

A présent que nous savons qu'il y a une correspondance, essayons de retrouver les portions de codes similaires entre `Regin` et `Qwerty`. Pour cela nous ré-utilisons la méthode déjà employée sur `Stuxnet`/`Duqu` dans [BMST12]. Il suffit de calculer le graphe en commun le plus grand possible entre le binaire `20120.dll` et `Orchestrator-41391495` afin qu'il soit unique. Il ne reste plus qu'à faire correspondre chaque noeud aux adresses des deux programmes. Nous avons développé un plugin afin de faciliter la visualisation des deux binaires et adapté la détection morphologique afin de produire un fichier contenant la correspondance entre les adresses.



Chaque instruction surlignée en orange correspond à une adresse identifiée partagée par l'analyse morphologique. Ainsi, nous pouvons juxtaposer facilement les fonctions identiques aux deux programmes et faciliter l'analyse de celles-ci en s'appuyant sur le code de `Qwerty`.

Nous retrouvons également une portion de code similaire faisant référence au numéro de plugin `20120` pour Qwerty et 19 pour `Orchestrator`. Et nous retrouvons ainsi la correspondance telle qu'elle a été établie par Kaspersky.

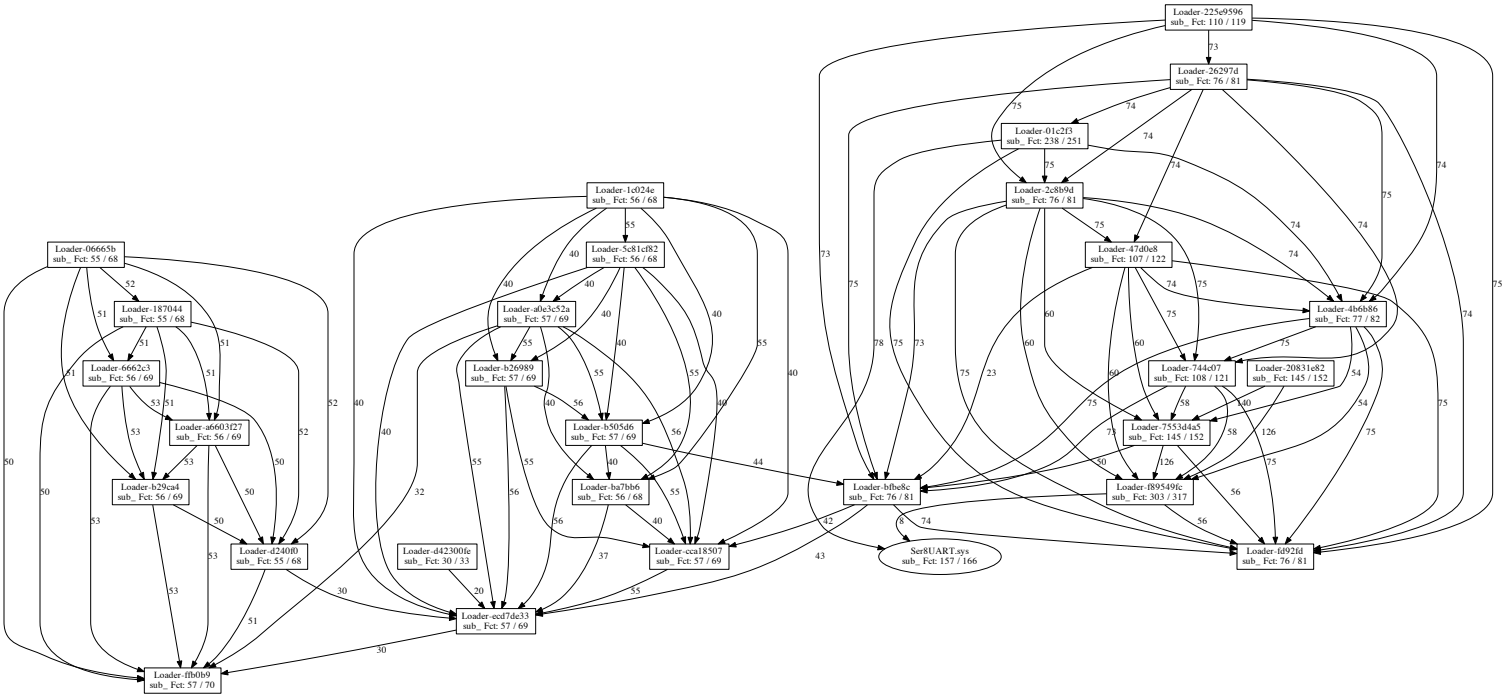


Le code similaire entre les deux binaires fait référence à un numéro d'identification d'un plugin.

3 Une analyse en famille

Revenons sur les loader de `Regin`. Nous avons déjà vu qu'ils étaient tous un peu différents les uns des autres. Cela nous permet de montrer un autre usage de l'analyse morphologique. A partir des similarités, il est possible de définir une distance entre les exemplaires, et ainsi de faire émerger un petit nombre de famille de malwares proches, et par conséquent, lors de la rétro-ingénierie, de ne se concentrer que sur quelques exemplaires.

Pour cela, nous procédons par analyse des fonctions communes de malware. Toujours grâce à l'analyse morphologique, on identifie les fonctions qui ont un code commun, ou proche. Ainsi, nous pouvons pour chaque paire d'échantillon donner le nombre de fonctions communes. Voici le résultat pour la famille des loaders de `Regin`. Attention, pour une telle approche, il faut penser restreindre l'analyse en distance uniquement sur des fonctions propres, pas sur les fonctions systèmes !



Sur ce graphe, il y a de nombreuses informations. Prenons le loader `Loader-06665b` en haut à gauche. Il contient 68 fonctions, et 55 sont propres (pas du système). L'arc qui le lie au `Loader-187044` montre que 52 fonctions sont communes entre les deux loaders. Presque 95% des fonctions partagées, pour la rétro-ingénierie, cela veut dire qu'on pourra oublier un des deux exemplaires.

Pour faire de la classification, on fait comme pour la recherche de communauté sur un réseau social, on applique un seuil (qui va supprimer les liens de parenté les plus faibles) et le résultat est là: on trouve trois belles classes de loader. De quoi restreindre franchement le travail de rétro-ingénierie !

4 Conclusion

Nous rejoignions les conclusions de Kaspersky sur le fait que `Qwerty` est bien un plugin de `Regin` mais en plus nous avons retrouvé des fonctions communes dans ces deux binaires. Et par un effet de bord, comme `Qwerty` est connu [Spi] pour faire partie de la plateforme `Warriorpride`, on peut en déduire que c'est vraisemblablement aussi le cas de `Regin` . . . , une affaire à suivre.

References

[BMST12] Guillaume Bonfante, Jean-Yves Marion, Fabrice Sabatier, and Aurélien Thierry. Recognition of binary patterns by morphological analysis. In *RECON 2012*, 2012.

[Kasa] "`http://25zbkz3k00wn2tp5092n6di7b5k.wpengine.netdna-cdn.com/files/2014/11/Kaspersky_Lab_whitepaper_Regin_platform_eng.pdf`".

[Kasb] http://securelist.com/blog/research/68525/comparing-the-regin-module-50251-and-the-qwerty-keylogger/.

[Spi] http://www.spiegel.de/media/media-35668.pdf.

[Sym] `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/regin-analysis.pdf`.

[ThI] `https://firstlook.org/theintercept/2014/11/24/secret-regin-malware-belgacom-nsa-gchq/`.