Blogs: Security

Analyzing BotNets with Suricata & Machine Learning

Since the official rollout at the year's. conf of the Machine Learning Toolkit(**MLTK**), Splunkers have been pursing some interesting use cases ranging from IT operations, planning, security and business analytics. Those use cases barely scratch the surface of what is possible with machine learning and Splunk. As an example, I will use the machine learning toolkit and data collected from Suricata to analyze botnet populations. This population analysis will be used to create a model for predicting the Mirai botnet based on network features.

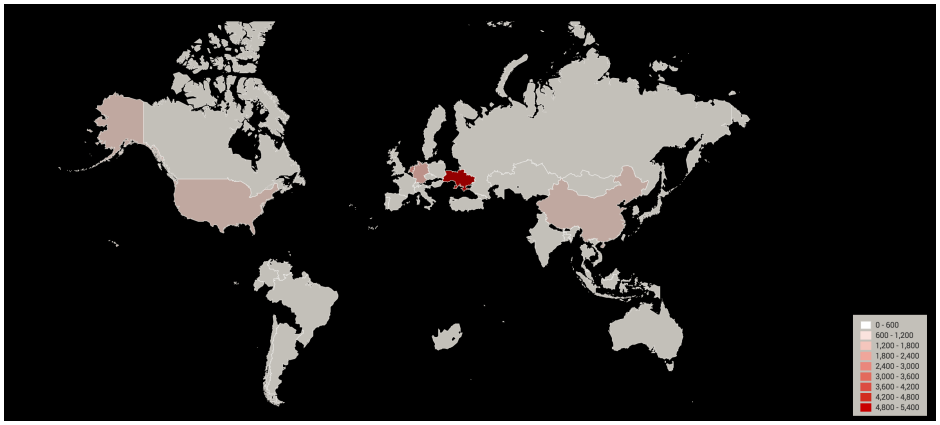
Suricata

Suricata is an open source threat detection engine, which can be run in passive mode for intrusion detection or inline for intrusion prevention. My lab environment is configured for intrusion detection, meaning Suricata will not make any attempt to prevent an intruder from accessing my system. This is a "good" thing because the behavioral signature of Mirai (and variants) use specific usernames for IoT devices found in the scanner.c module in the sshd logs & telnet sessions of the server it attempts to infect.

```
add_auth_entry("\x14\x14\x14\x14\x14\x14", "\x14\x14\x14\x14\x14\x14", 1);              // 666666   666666
add_auth_entry("\x1A\x1A\x1A\x1A\x1A\x1A", "\x1A\x1A\x1A\x1A\x1A\x1A", 1);              // 888888   888888
add_auth_entry("\x57\x40\x4C\x56", "\x57\x40\x4C\x56", 1);                              // ubnt     ubnt
add_auth_entry("\x50\x4D\x4D\x56", "\x49\x4E\x54\x13\x10\x11\x16", 1);                  // root     klv1234
add_auth_entry("\x50\x4D\x4D\x56", "\x78\x56\x47\x17\x10\x13", 1);                      // root     Zte521
add_auth_entry("\x50\x4D\x4D\x56", "\x4A\x4B\x11\x17\x13\x1A", 1);                      // root     hi3518
add_auth_entry("\x50\x4D\x4D\x56", "\x48\x54\x40\x58\x46", 1);                          // root     jvbzd
add_auth_entry("\x50\x4D\x4D\x56", "\x43\x4C\x49\x4D", 4);                              // root     anko
add_auth_entry("\x50\x4D\x4D\x56", "\x58\x4E\x5A\x5A\x0C", 1);                          // root     zlxx.
add_auth_entry("\x50\x4D\x4D\x56", "\x15\x57\x48\x6F\x49\x4D\x12\x54\x4B\x58\x5A\x54", 1); // root     7ujMko0vizxv
add_auth_entry("\x50\x4D\x4D\x56", "\x15\x57\x48\x6F\x49\x4D\x12\x43\x46\x4F\x4B\x4C", 1); // root     7ujMko0admin
add_auth_entry("\x50\x4D\x4D\x56", "\x51\x5B\x51\x56\x47\x4F", 1);                      // root     system
add_auth_entry("\x50\x4D\x4D\x56", "\x4B\x49\x55\x40", 1);                              // root     ikwb
add_auth_entry("\x50\x4D\x4D\x56", "\x46\x50\x47\x43\x4F\x40\x4D\x5A", 1);              // root     dreambox
add_auth_entry("\x50\x4D\x4D\x56", "\x57\x51\x47\x50", 1);                              // root     user
add_auth_entry("\x50\x4D\x4D\x56", "\x50\x47\x43\x4E\x56\x47\x49", 1);                  // root     realtek
add_auth_entry("\x50\x4D\x4D\x56", "\x12\x12\x12\x12\x12\x12\x12\x12", 1);              // root     00000000
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x13\x13\x13\x13\x13\x13\x13", 1);              // admin    1111111
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x13\x10\x11\x16", 1);                          // admin    1234
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x13\x10\x11\x16\x17", 1);                      // admin    12345
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x17\x16\x11\x10\x13", 1);                      // admin    54321
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x13\x10\x11\x16\x17\x14", 1);                  // admin    123456
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x15\x57\x48\x6F\x49\x4D\x12\x43\x46\x4F\x4B\x4C", 1); // admin    7ujMko0admin
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x16\x11\x10\x13", 1);                          // admin    1234
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x52\x43\x51\x51", 1);                          // admin    pass
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x4F\x47\x4B\x4C\x51\x4F", 1);                  // admin    meinsm
add_auth_entry("\x56\x47\x41\x4A", "\x56\x47\x41\x4A", 1);                              // tech     tech
add_auth_entry("\x4F\x4D\x56\x4A\x47\x50", "\x44\x57\x41\x49\x47\x50", 1);              // mother   f[illegible]ker
```

Analysis

The analysis largely builds upon the previous blog post (Analyzing the Mirai Botnet with Splunk), which correlated the failed logins of specific usernames and ip addresses. This threatlist of suspected Mirai ip addresses can be analyzed for various features such as geography, IANA registration, frequency, etc...



CATEGORIES

TIPS & TRICKS	1173
SECURITY	301
LIFE AT SPLUNK	336
DEV	372
UI & DESIGN	62
CUSTOMERS	357
.CONF SPEAKERS	148
SPLUNKNEWS	161
CLOUD	109
WHERE WILL YOUR DATA TAKE YOU?	234
SPLUNK>4GOOD	80
HEALTHCARE	13
PUBLIC SECTOR	74
MOBILE	34
SPLUNKLIVE!	13
EVENTS	109
LEADERSHIP	42
PARTNERS	54

Tags

List | Cloud

splunk (193)
Security (169)
community (100)
microsoft (81)
splunktalk (79)
podcast (75)
big data (74)
Splunk Enterprise (62)
#splunkconf (62)
analytics (50)
operational intelligence (50)
apps (48)
Cloud (47)
.conf2015 (47)
public sector (46)

Splunk Insights

Combining this threatlist with our passive intrusion detection netflow data creates an enriched dataset for building a model. Adding, contextual and detailed information about each access attempt at the packet level provides insights into the activity attempted by that IP or block of IPs during a 24 hour window. As an example, we can determine which tcp flags were present in the packets both client side & server side in each flow transaction and begin grouping similar events together. We can also create a ratio of **packets_in** v. **packets_out** and classify these flows into various producer consumer ratio(**PCR**) categories.

City	Country	Designation	Region	_time	isMirai	lat	lon	packet_pcr_range	packet_ratio	packets_in	packets_out	packets_total	src_ip	src_port	tcp_flag_hex_to_client	tcp_flag_hex_to_server
	Thailand	APNIC		2016-10-13	0	13.75000	100.46670	3:1 Import	-0.111111	5	4	9	1.10.214.109	p_20599	1e	1a
	China	APNIC		2016-08-23	1	35.00000	105.00000	3:1 Import	-0.037736	55	51	106	1.119.7.35	p_9224	19	1b
	China	APNIC		2016-08-23	1	35.00000	105.00000	3:1 Import	-0.070000	107	93	200	1.119.7.35	p_9224	1b	1b
Seoul	Republic of Korea	APNIC	Seoul	2016-09-19	0	37.59850	126.97830	Balanced Exchange	0	18	18	36	1.209.148.34	p_44084	1b	1b
Seoul	Republic of Korea	APNIC	Seoul	2016-09-19	0	37.59850	126.97830	3:1 Import	-0.058824	18	16	34	1.209.148.34	p_45435	1b	1b
Seoul	Republic of Korea	APNIC	Seoul	2016-08-19	0	37.59850	126.97830	70:30 Export	0.083333	11	13	24	1.221.171.138	p_54085	1b	1f
Seoul	Republic of Korea	APNIC	Seoul	2016-08-19	0	37.59850	126.97830	70:30 Export	0.111111	20	25	45	1.221.171.138	p_9224	1b	1f
	Republic of Korea	APNIC		2016-10-20	0	37.57000	126.98000	3:1 Import	-0.028571	18	17	35	1.234.45.104	p_32767	1b	1b
	Republic of Korea	APNIC		2016-10-12	0	37.57000	126.98000	70:30 Export	0.120000	11	14	25	1.234.45.116	p_33535	1b	1f
	Republic of Korea	APNIC		2016-10-11	0	37.57000	126.98000	Balanced Exchange	0	18	18	36	1.234.6.102	p_40703	1b	1b
	Republic of Korea	APNIC		2016-09-21	0	37.57000	126.98000	70:30 Export	0.083333	11	13	24	1.234.6.102	p_49259	1b	1f
	Republic of Korea	APNIC		2016-09-18	0	37.57000	126.98000	Balanced Exchange	0	18	18	36	1.234.6.102	p_55239	1b	1b
	Republic of Korea	APNIC		2016-09-17	0	37.57000	126.98000	3:1 Import	-0.147541	35	26	61	1.234.6.102	p_55551	1b	1b
	Republic of Korea	APNIC		2016-09-20	0	37.57000	126.98000	70:30 Export	0.120000	11	14	25	1.234.6.102	p_56853	1b	1f
	Republic of Korea	APNIC		2016-10-14	0	37.57000	126.98000	3:1 Import	-0.142857	44	33	77	1.234.62.253	p_53503	1b	1b
	Republic of Korea	APNIC		2016-10-13	0	37.57000	126.98000	3:1 Import	-0.028571	18	17	35	1.234.62.253	p_59647	1b	1b
	Republic of Korea	APNIC		2016-10-10	0	37.57000	126.98000	3:1 Import	-0.028571	18	17	35	1.234.69.90	p_44287	1b	1b
	Republic of Korea	APNIC		2016-09-20	0	37.57000	126.98000	70:30 Export	0.120000	11	14	25	1.234.69.90	p_45461	1b	1f
	Republic of Korea	APNIC		2016-09-21	0	37.57000	126.98000	70:30 Export	0.142857	9	12	21	1.234.69.90	p_48857	1b	1f
	Republic of Korea	APNIC		2016-10-16	0	37.57000	126.98000	70:30 Export	0.083333	11	13	24	1.234.69.90	p_56631	1b	1f

MLTK

The MLTK is handy because of the many assistants which ship with the tool, you don't need to know the exact SPL syntax to begin making use of it. Using the clustering assistant, I can attempt to discern different botnet populations based on the features present in my dataset. In the below example, I have selected 50k random Suricata flow events, where the **dest_port** is 22. I have picked features which have some relation to each other, but are enriched by the **PCR** metric. I have created a label of isMirai with possible values of **0** | **1**, depending on the IP address associated with that flow event. I have opted for Kmeans clustering with a value of **k=5**.

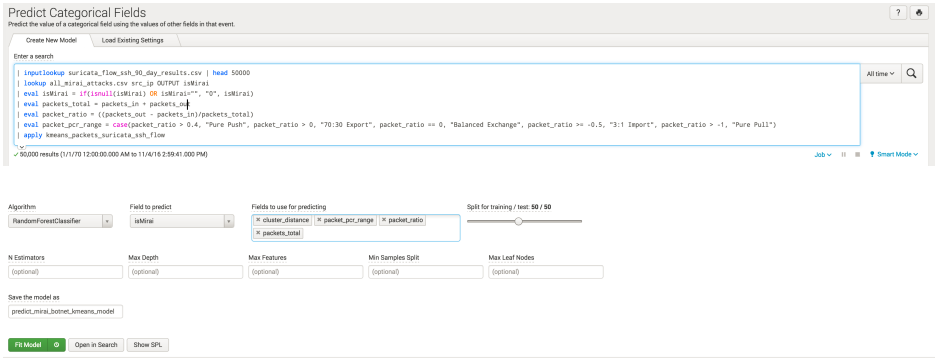
Interestingly, a clear visual pattern emerges with **cluster_4**. It is clearly an outlier compared to the rest of the population, but is there anything special about it? From an isMirai **0** | **1**, perspective there is a mixture of both 1's and 0's. The **packet_pcr_range**, is **3:1 Import**, with varying ratios, which seems to be the only common feature of **cluster_4**.

packet_pcr_range	packet_ratio	packets_in	packets_out	packets_total	isMirai	count
3:1 Import	-0.008696	116	114	230	1	1
3:1 Import	-0.014286	142	138	280	1	1
3:1 Import	-0.014749	172	167	339	1	1
3:1 Import	-0.017143	178	172	350	0	1
3:1 Import	-0.026316	156	148	304	0	1
3:1 Import	-0.030301	238	220	458	0	1
3:1 Import	-0.047319	166	151	317	0	1
3:1 Import	-0.047619	154	140	294	0	1
3:1 Import	-0.04797	142	129	271	1	1
3:1 Import	-0.054545	116	104	220	1	1
3:1 Import	-0.058824	108	96	204	0	1
3:1 Import	-0.064615	173	152	325	1	1
3:1 Import	-0.07	107	93	200	1	1
3:1 Import	-0.081761	172	146	318	0	1
3:1 Import	-0.09466	133	110	243	1	1
3:1 Import	-0.115789	106	84	190	0	1
3:1 Import	-0.122995	105	82	187	0	1
3:1 Import	-0.129032	105	81	186	0	1
3:1 Import	-0.130435	104	80	184	0	1
3:1 Import	-0.132275	107	82	189	0	1

Using a model for prediction

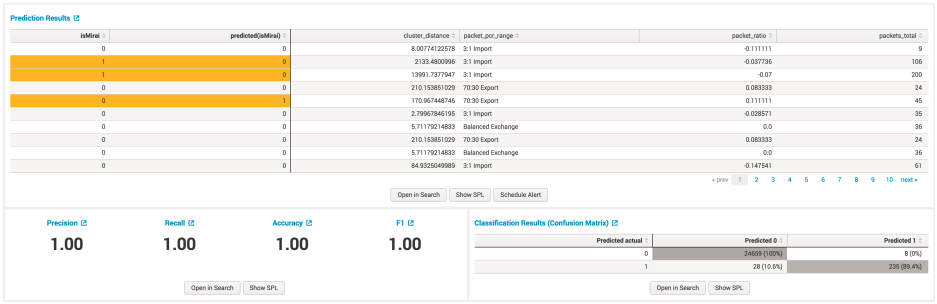
MLTK isn't intended to create models for the sake of creating models, it also allows you to operationalize those models for predicting based on features found in the model, one such feature we get from kmeans is the cluster_distance. This number describes the distance an event is from the centroid.

Using the prediction assistant, the Kmeans model can be loaded in search before selecting features from the dropdown to use for prediction. We can then select the features we wish to use for prediction: **cluster_distance, packet_pcr_range, packet_ratio**, and **packet_total**. The prediction assistant also gives you the ability to adjust the specific algorithm to use for prediction, I have opted for **Random Forest**.



Next Steps

The model appears to be very good at predicting 0 (not Mirai), while it is reasonably good at predicting 1 (89.4%). This is an improvement over Suricata, which did not detect Mirai with the emerging threats ruleset. This may imply that there is an indicator of compromise for the Mirai botnet at the packet level. Proof of this requires further investigation, and independent validation to understand why the model can predict Mirai so effectively to eliminate bias or mistakes. Collaboration with others who have also gathered traffic from botnets is a great way to validate the model against a data set it has not seen before to further validate. If an indicator of compromise can be discerned from this analysis it could be converted into an IDS signature for future detection of Mirai infection attempts.



JANUARY 30, 2017

0

Posted by Anthony Tellez in Security, Tips & Tricks, Where will your Data Take You?

Tags:, IoT, machine learning, Mirai, suricata