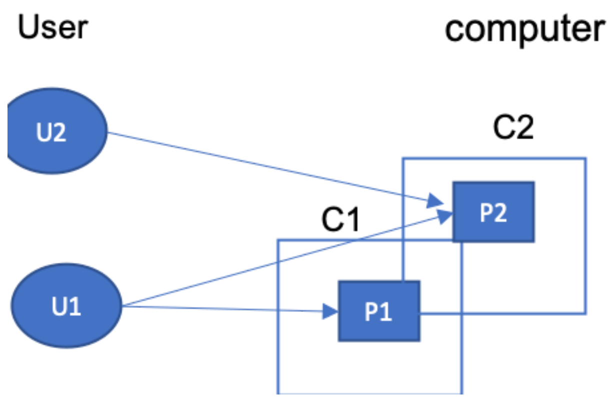
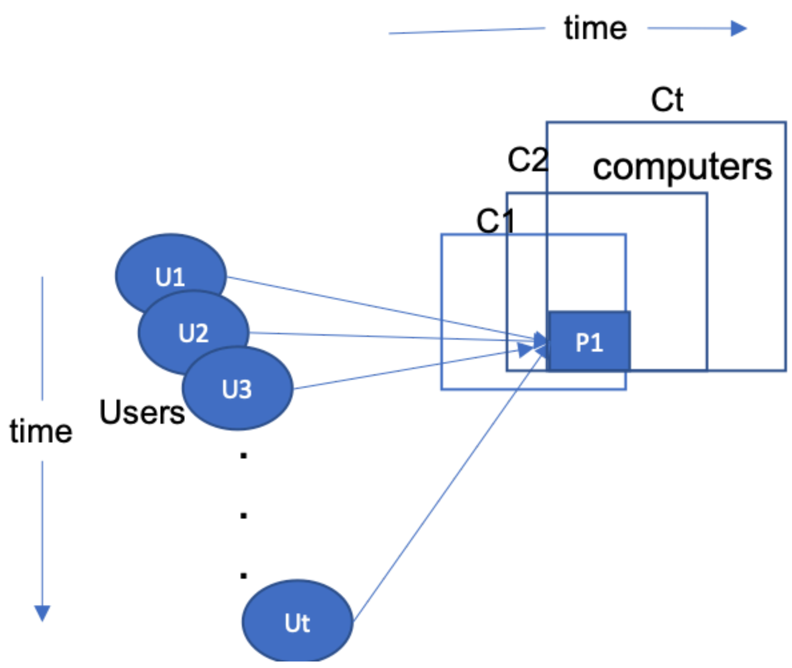
Behavioral Modeling of Computer Processes and Anomaly Detection

Process interaction ratio

The figure below shows a schematic of users working on their machines and the subsequent processes generated. An enterprise network of users will generate an intricate network graph since many users share computers and start processes in an intertwined manner.



In order to make sense of the change in behavior of processes, it is necessary to look at user – computer interactions from the point of view of processes. The figure below shows an abstraction centered on processes and the user-computer interaction based on a duration of time *t*.



Based on this abstraction, we develop an important ratio that represents process behavior in time by dividing the number of computers that host an active process *p,* by the number of users who have initiated that process in a period of time t.

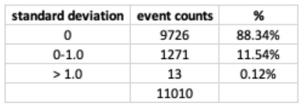
$$process\ interaction\ ratio\ (PIR)\ =\ \frac{no.\ of\ unique\ computers}{no.\ of\ unique\ users}$$

Interpreting process interaction ratio

For each process, the $process\ interaction\ ratio$ is very consistent from one day to the other. Table 2 below shows the distribution of this ratio based on LANL events tracked over 6-day period. When this ratio is above 1.0 means that a user most likely initiates the same process across more than one machine. If the ratio falls below 1.0, this indicates that users are triggering processes on shared machines. Both scenarios could be indicators of fishy behavior especially if there is a rapid accumulation of machines or users over short time periods. High or low PIR values are not inherently bad, but sudden unprecedented down or upswings could well be indicators of attacks and at the very least need to be detected and presented to personnel who can analyze them further. The table below shows the distribution of 6 days of process events from LANL dataset. From the table it is clear that an average of ~86.0% of the events have a process interaction ratio of 1.0 which means the same users are activating the same process on the same machines on a daily basis. These will most likely be normal process events and can be ignored which reduces the scope of events under investigation by close to 90%. There is only about 3.0% of events with process interaction ratio greater than 1.0 on any given day. This is actually the most critical cohort in terms of process abuse likelihood since it represents processes which are being activated on new hosts and therefore likely to be anomalous. There is about 10% of events that have a PIR less than 1.0 on any given day.

	Day 1		Day 2		Day 3		Day 4		Day 5		Day 6	
process Interaction ratio (PIR)	PIR events	% distribution	PIR events	% distribution	PIR events	% distribution	PIR events	% distribution	PIR events	% distribution	PIR events	% distribution
1	5239	88.021%	5061	85.548%	3747	86.019%	1370	86.984%	1177	84.433%	5610	86.521%
< 1.0	569	9.560%	661	11.173%	455	10.445%	132	8.381%	140	10.043%	683	10.534%
> 1.0	144	2.419%	194	3.279%	154	3.535%	73	4.635%	77	5.524%	191	2.946%
Total	5952		5916		4356		1575		1394		6484	

What is even more important, is tracking the PIR from one given time period to another as these changes would signify important shifts in process behavior. The table below shows the standard deviation of PIR over a period of 8 days. From the table it seems that 88.34% of processes do not change at all in terms of their PIR going by their 0 magnitude in standard deviation. 11.54% witnessed at maximum a doubling or halving of PIR. The most significant changes in PIR are processes that had more than a doubling in standard deviation which occurred in a very small cohort of processes (about 0.12%). These results tell us that there are only a few instances of process behavior that we need to actively track in order to catch sensible anomalies in an enterprise network.



standard deviation	event counts	%
0	9726	88.34%
0-1.0	1271	11.54%
> 1.0	13	0.12%
	11010	

PIR process use cases

There are several important use cases that can be based on modeling process PIR:

1. Users starting processes on machines they are not usually associated with or processes getting executed on machines where the processes are not usually executed. This captures process footprint propagating to new machines.
2. Users starting processes they normally don't execute. This represents users acquiring new behavior.
3. A combination of 1 and 2.

We can expound on these use cases using an example:
We have two Users -[U1,U2], who frequently start one process P1 on two machines [M1,M2].
Case 1: [U1,U2] now start process P1 on [M1,M2,M3]
This causes PIR to increase. Due to introduction of machine M3. This could be a spoofed P1 (malicious software under the guise of P1) being executed in M3.
Case 2: [U1,U2,U3] now start process P1 on [M1,M2]
PIR goes down due to introduction of new user (U3) who was not previously associated with P1. U3 could be running a malicious software under the guise of P1 on M1 and M2 (machines that he is not traditionally associated with).
Case 3: [U1,U2,U3] now start process P1 on [M1,M2,M3,M4,M5]
PIR shoots up caused by a proliferation of process starts on new machines [M3,M4,M5] by a new user [U3] not previously associated with machines [M1,M2] and process P1. This represents change is both user and process behavior which is rare and therefore interesting.

Monitoring PIR algorithmically to detect anomalies

In order to monitor PIR, one needs to determine the duration of time that is reasonable to track. Essentially, it boils down to computational resources one is able to dedicate for this. Based on the analysis we did, you do not expect to see major changes in PIR at levels of time granularity less than one hour. One day time spans also look reasonable and would make the production cycles easier to handle.
Steps:

1) For each process in the network, compute PIR by dividing number of machines that hosted the process by the number of users who started them for each hour during a 24-hour span.
2) Use the PIR data points collected in the first day as history from which we can calculate the *mean* PIR and expected *standard deviation* of PIR.
3) Obtain PIR from a new time period T_1 for process. T_1 could an hour or a day. Use history to calculate the PIR *z-score* from the historical *mean* and *standard deviation*. PIR *z-score* is an indicator of anomalousness of PIR value.
4) Identify processes whose PIR *z-score* is very huge (usually > 6.0) and also are backed by a significant number of historical values (usually > 24). If a process has a high *z-score* but

very few historical data points then it is prudent to avoid it and wait for its history to accumulate.

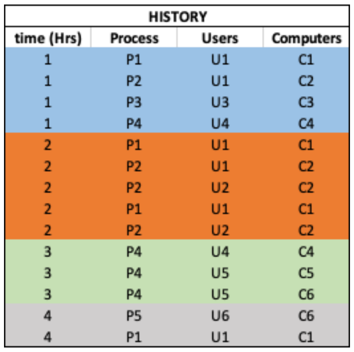
5) Take the events with high PIR *z-score* and ascertain that even at smaller time windows (an hour or less) that the PIR values are exceptionally high for certain times.
6) Combine T_1 events with history to create a new history which will be used to compare with T_2 and follow step 4 to identify anomalous cases. This process is repeated till you get to the end. Obviously, the length of history one needs to hold has to be capped to minimize the computational burden. About two weeks of most recent history should suffice and the rest of the events should be aged out.

Resolving problems of PIR complexity

The greatest challenge of this algorithm is to be able to track PIR of processes over time and be able to compute significant deviations from history fast enough. In order to catch these anomalies in real time, one needs to compute PIR for all the process in small intervals of time. Using LANL data set, as an example of an enterprise network traffic, this means tracking 11,000 processes PIR. Various time series algorithms such as CUSUM [1], exponentially weighted moving averages EWMA [2] or Bayesian Online detection [2] can be used comfortably for this task but it would require tracking each process separately which can be computationally very intensive since 11000 re-computations have to be done in real time and the history carried forward. Based on analysis of LANL data, this is an overkill since most processes' PIR (close to 90%) do not change from one day to the other. We therefore propose only computing PIR every hour for the first day (in order to build history) and at the end of the day for each subsequent day. When computing deviations of each process's PIR from history we will only compute deviations from just about 10% of process since 90% will not have changes at all. This reduction in computation will allow us to use more complicated time series techniques comfortably if need be.

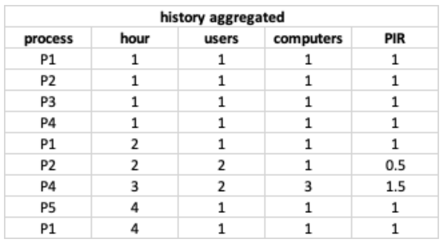
PIR anomaly detection algorithm procedure

- We begin by collecting process events at an hourly rate to build initial history. At the end of day 1 we will have 24 hours of all process 'start' events aggregated at hourly intervals which we can refer to as Table 1.



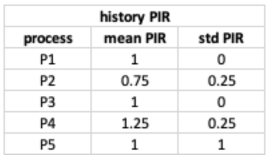
HISTORY			
time (Hrs)	Process	Users	Computers
1	P1	U1	C1
1	P2	U1	C2
1	P3	U3	C3
1	P4	U4	C4
2	P1	U1	C1
2	P2	U1	C2
2	P2	U2	C2
2	P1	U1	C1
2	P2	U2	C2
3	P4	U4	C4
3	P4	U5	C5
3	P4	U5	C6
4	P5	U6	C6
4	P1	U1	C1

- From Table 1 we can create a table of all unique process events which will be updated daily. This table, referred to as table 2, will be used to identify new processes in the network and new user and computer interactions with a process that already exists. This history can be held indefinitely as we do not expect it to grow very fast - from experience new processes in the network are not common. Table 2 will be made up of – [Process_name|User_ID|Computer_ID] combo.
- From Table 1, compute the process interaction ratio (PIR) at an hourly basis – refer this as Table 3



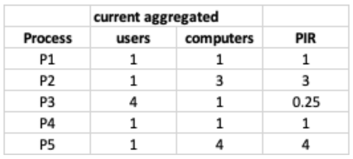
		history aggregated		
process	hour	users	computers	PIR
P1	1	1	1	1
P2	1	1	1	1
P3	1	1	1	1
P4	1	1	1	1
P1	2	1	1	1
P2	2	2	1	0.5
P4	3	2	3	1.5
P5	4	1	1	1
P1	4	1	1	1

From Table 3 PIR process events we can go further and compute the PIR mean and standard deviation based on each process for the whole day.



	history PIR	
process	mean PIR	std PIR
P1	1	0
P2	0.75	0.25
P3	1	0
P4	1.25	0.25
P5	1	1

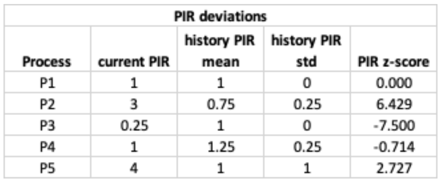
- Generate / collect new events from a new time instance of interest. We can then create a table (Table 4) which contains new PIR values for all new process instances collected. Keep a snapshot of raw hourly process events in order to later verify when anomalous behavior started on that day.



	current aggregated		
Process	users	computers	PIR
P1	1	1	1
P2	1	3	3
P3	4	1	0.25
P4	1	1	1
P5	1	4	4

- Using Table 3 and Table 4 to compute deviations of new PIR events (Table 4) from history (Table 3) using the formula:
    - $PIR\ z\_score = \frac{new\ PIR - PIR\ mean}{PIR\ standard\ deviation}$

create new table (Table 5) of PIR *z-score* based on each process for the new process events.



PIR deviations				
Process	current PIR	history PIR mean	history PIR std	PIR z-score
P1	1	1	0	0.000
P2	3	0.75	0.25	6.429
P3	0.25	1	0	-7.500
P4	1	1.25	0.25	-0.714
P5	4	1	1	2.727

- From Table 5, identify extremely high and low PIR z-score – these will be processes that have seen significant change and are the ones we need to pick out. The criteria for what is extremely high depends on the magnitude of Z-score and how many data points were involved in computing the mean and standard deviation of PIR.
- Update Table 3 by averaging historical PIR values with new ones and this becomes the new history. This process continues on a daily basis and keeps generating new alarms while keeping history updated. Bear in mind that we don't have to keep all history as there are computationally efficient mechanisms of cheaply maintaining mean and standard deviations by only using most previous value and an updated counter by weighting the values.

Minimizing False Alarms

There are going to be many moments when high PIR scores will be identified but will not be malicious. This is why we need to go a step further and definitively point the highest probable cases where and when (close to real time as possible) malicious events occurred. We should not flag high PIR events caused by new computers or users introduced into the network for the first time since it happens frequently in an enterprise network. To pinpoint hours where there is high certainty of malice, we use hourly Table 4 process events and consult Table 2, to identify for each hour and process:

- counts of first-time computer(s) hosted a process - Fc
- counts of first-time user(s) started a process – Fu
- counts of first-time computer(s) introduced to network – Nc
- counts of first-time user(s) introduced to network – Nu

For each and every hourly event we compute these four variables and use them to come up with a new variation of PIR measure:

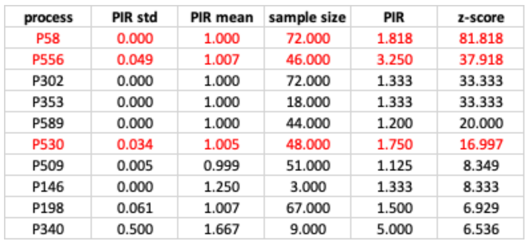
- hourly $PIR = \frac{Fc - Nc}{Fu - Nu}$

The distribution of hourly PIRs can be computed based on history and the probability of each hourly PIR is computed. Very low probability hourly PIRs can be identified based on a p-value basis and flagged as most likely to be anomalous.

Results of PIR algorithm and examples

Table 8 below shows results from running the PIR algorithm on LANL data set of process events collected for a seven-day period from over 30,000 events comprising 25000 users, 9700 computers and 11967 unique processes. The results constitute the top most anomalous events on day 3 based on:

- PIR z-score – this is a measure of deviation from historical PIR mean. The higher the PIR signifies that the PIR ratio has increased significantly. For instance, a PIR z-score of 6.0 means the PIR has increased six-fold standard deviations above historical PIR mean.
- PIR – this is a measure of how many computers are hosting a process in with respect to the users. High PIR signifies users are executing new processes in new machines which could be an indicator of compromise.
- No. of data points – PIR z-score, PIR standard deviation and mean are based on number of historical data points observed and are more reliable if they are based on a huge sample of data points. In our estimates, we can only rely on statistical values based on at least 24 data points [*].



process	PIR std	PIR mean	sample size	PIR	z-score
P58	0.000	1.000	72.000	1.818	81.818
P556	0.049	1.007	46.000	3.250	37.918
P302	0.000	1.000	72.000	1.333	33.333
P353	0.000	1.000	18.000	1.333	33.333
P589	0.000	1.000	44.000	1.200	20.000
P530	0.034	1.005	48.000	1.750	16.997
P509	0.005	0.999	51.000	1.125	8.349
P146	0.000	1.250	3.000	1.333	8.333
P198	0.061	1.007	67.000	1.500	6.929
P340	0.500	1.667	9.000	5.000	6.536

Highlighted in red are processes that show all the hallmarks of the criteria we have set above and were later verified to have anomalous behavior. The results in table above are based on daily calculation of PIR. In order to identify where the deviation begins we need to look at hourly changes in PIR for the process identified. The Table 9 below shows streams of hourly PIR for process *P556*. Highlighted in red is 77th hour where the PIR shoots up significantly to about 10. The events indicate that one new user executed process *P556* in ten new machines not associated with the process before. It's also good to note that these machines are not new in the network. It is normal for PIR ratio to increase if a new computer/user is introduced in the network and this is an important check that should be made before alarming.

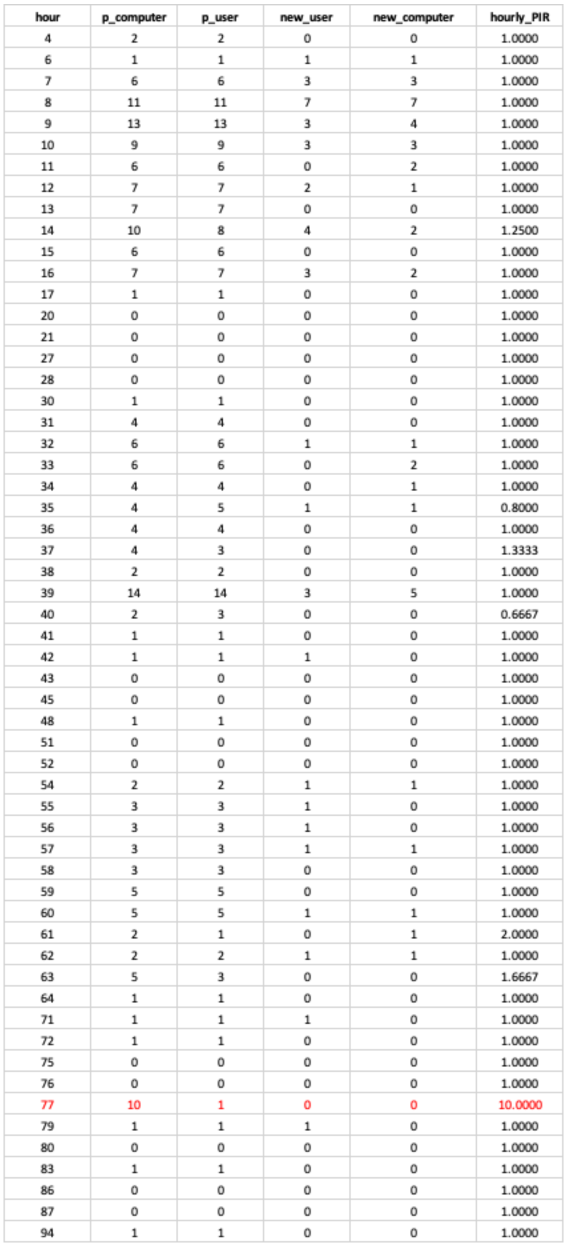
hr	new computer hosting process	new user executing process	no. of new users in network	no. of new computers in network	Hourly PIR
3	1	1	0	0	1
5	1	1	1	1	1
6	5	5	4	4	1
7	13	13	13	12	1
8	11	11	8	8	1
9	6	6	3	3	1
10	3	3	0	0	1
12	2	2	1	1	1
13	3	3	0	0	1
14	2	2	0	0	1
15	1	1	0	0	1
16	1	1	0	0	1
17	2	2	0	0	1
20	0	0	0	0	1
22	1	1	0	0	1
25	0	0	0	0	1
27	0	0	0	0	1
28	1	1	0	0	1
29	1	1	0	0	1
30	2	2	1	1	1
31	4	4	0	0	1
32	2	2	1	0	1
33	6	4	1	1	1.5
34	2	2	0	0	1
35	4	4	0	0	1
36	3	3	1	0	1
37	2	2	0	0	1
38	2	2	1	1	1
39	5	5	1	0	1
40	0	0	0	0	1
41	1	1	0	0	1
42	4	4	1	1	1
43	1	1	0	0	1
46	0	0	0	0	1
50	0	0	0	0	1
51	2	2	0	0	1
54	1	1	0	0	1
55	1	2	0	0	0.5
56	2	2	2	0	1
57	1	1	0	0	1
58	3	3	0	0	1
59	1	1	0	0	1
60	1	1	1	0	1
61	1	0	0	0	1
63	0	0	0	0	1
75	0	0	0	0	1
77	10	1	0	0	10
83	1	1	0	0	1

The table 10 below shows the 10 new events behind the elevated PIR. We have tagged the events that we have been able to match with the LANL Redteam list of anomalous events. The User ID (U78@DOM1) is a new user that had not executed *P556* before and all the host computers in the table were hosting the process for the first time.

time in sec	Hour	process	User ID	Host Computer	in Redteam ?
20411943	77	P556	U78@DOM1	C1089	yes
20411949	77	P556	U78@DOM1	C2039	No
20411955	77	P556	U78@DOM1	C2057	yes
20412116	77	P556	U78@DOM1	C1382	yes
20412120	77	P556	U78@DOM1	C1611	yes
20412127	77	P556	U78@DOM1	C2254	yes
20412133	77	P556	U78@DOM1	C742	yes
20412283	77	P556	U78@DOM1	C1993	No
20412550	77	P556	U78@DOM1	C1139	No
20412556	77	P556	U78@DOM1	C1503	yes

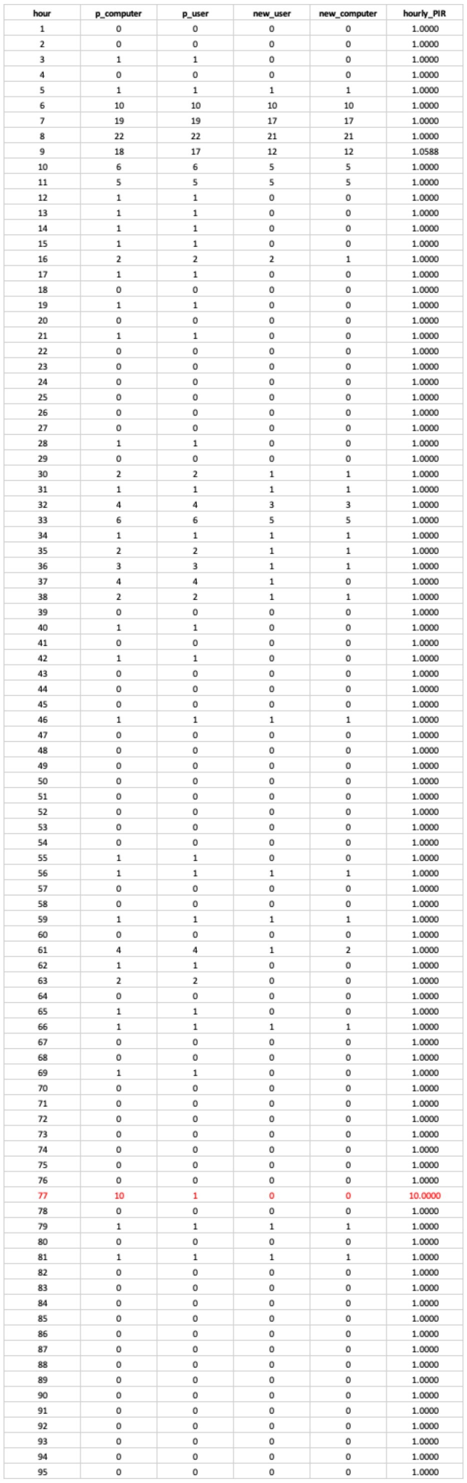
Table 11 & 12 below shows anomalous PIR process events from P530 and P28. It is interesting to note that these aberrant events all occur in the 77th hour and are all caused by the same user *U78@DOM1*

Process P530



hour	p_computer	p_user	new_user	new_computer	hourly_PIR
4	2	2	0	0	1.0000
6	1	1	1	1	1.0000
7	6	6	3	3	1.0000
8	11	11	7	7	1.0000
9	13	13	3	4	1.0000
10	9	9	3	3	1.0000
11	6	6	0	2	1.0000
12	7	7	2	1	1.0000
13	7	7	0	0	1.0000
14	10	8	4	2	1.2500
15	6	6	0	0	1.0000
16	7	7	3	2	1.0000
17	1	1	0	0	1.0000
20	0	0	0	0	1.0000
21	0	0	0	0	1.0000
27	0	0	0	0	1.0000
28	0	0	0	0	1.0000
30	1	1	0	0	1.0000
31	4	4	0	0	1.0000
32	6	6	1	1	1.0000
33	6	6	0	2	1.0000
34	4	4	0	1	1.0000
35	4	5	1	1	0.8000
36	4	4	0	0	1.0000
37	4	3	0	0	1.3333
38	2	2	0	0	1.0000
39	14	14	3	5	1.0000
40	2	3	0	0	0.6667
41	1	1	0	0	1.0000
42	1	1	1	0	1.0000
43	0	0	0	0	1.0000
45	0	0	0	0	1.0000
48	1	1	0	0	1.0000
51	0	0	0	0	1.0000
52	0	0	0	0	1.0000
54	2	2	1	1	1.0000
55	3	3	1	0	1.0000
56	3	3	1	0	1.0000
57	3	3	1	1	1.0000
58	3	3	0	0	1.0000
59	5	5	0	0	1.0000
60	5	5	1	1	1.0000
61	2	1	0	1	2.0000
62	2	2	1	1	1.0000
63	5	3	0	0	1.6667
64	1	1	0	0	1.0000
71	1	1	1	0	1.0000
72	1	1	0	0	1.0000
75	0	0	0	0	1.0000
76	0	0	0	0	1.0000
77	10	1	0	0	10.0000
79	1	1	1	0	1.0000
80	0	0	0	0	1.0000
83	1	1	0	0	1.0000
86	0	0	0	0	1.0000
87	0	0	0	0	1.0000
94	1	1	0	0	1.0000

Process P58



hour	p_computer	p_user	new_user	new_computer	hourly_PIR
1	0	0	0	0	1.0000
2	0	0	0	0	1.0000
3	1	1	0	0	1.0000
4	0	0	0	0	1.0000
5	1	1	1	1	1.0000
6	10	10	10	10	1.0000
7	19	19	17	17	1.0000
8	22	22	21	21	1.0000
9	18	17	12	12	1.0588
10	6	6	5	5	1.0000
11	5	5	5	5	1.0000
12	1	1	0	0	1.0000
13	1	1	0	0	1.0000
14	1	1	0	0	1.0000
15	1	1	0	0	1.0000
16	2	2	2	1	1.0000
17	1	1	0	0	1.0000
18	0	0	0	0	1.0000
19	1	1	0	0	1.0000
20	0	0	0	0	1.0000
21	1	1	0	0	1.0000
22	0	0	0	0	1.0000
23	0	0	0	0	1.0000
24	0	0	0	0	1.0000
25	0	0	0	0	1.0000
26	0	0	0	0	1.0000
27	0	0	0	0	1.0000
28	1	1	0	0	1.0000
29	0	0	0	0	1.0000
30	2	2	1	1	1.0000
31	1	1	1	1	1.0000
32	4	4	3	3	1.0000
33	6	6	5	5	1.0000
34	1	1	1	1	1.0000
35	2	2	1	1	1.0000
36	3	3	1	1	1.0000
37	4	4	1	0	1.0000
38	2	2	1	1	1.0000
39	0	0	0	0	1.0000
40	1	1	0	0	1.0000
41	0	0	0	0	1.0000
42	1	1	0	0	1.0000
43	0	0	0	0	1.0000
44	0	0	0	0	1.0000
45	0	0	0	0	1.0000
46	1	1	1	1	1.0000
47	0	0	0	0	1.0000
48	0	0	0	0	1.0000
49	0	0	0	0	1.0000
50	0	0	0	0	1.0000
51	0	0	0	0	1.0000
52	0	0	0	0	1.0000
53	0	0	0	0	1.0000
54	0	0	0	0	1.0000
55	1	1	0	0	1.0000
56	1	1	1	1	1.0000
57	0	0	0	0	1.0000
58	0	0	0	0	1.0000
59	1	1	1	1	1.0000
60	0	0	0	0	1.0000
61	4	4	1	2	1.0000
62	1	1	0	0	1.0000
63	2	2	0	0	1.0000
64	0	0	0	0	1.0000
65	1	1	0	0	1.0000
66	1	1	1	1	1.0000
67	0	0	0	0	1.0000
68	0	0	0	0	1.0000
69	1	1	0	0	1.0000
70	0	0	0	0	1.0000
71	0	0	0	0	1.0000
72	0	0	0	0	1.0000
73	0	0	0	0	1.0000
74	0	0	0	0	1.0000
75	0	0	0	0	1.0000
76	0	0	0	0	1.0000
77	10	1	0	0	10.0000
78	0	0	0	0	1.0000
79	1	1	1	1	1.0000
80	0	0	0	0	1.0000
81	1	1	1	1	1.0000
82	0	0	0	0	1.0000
83	0	0	0	0	1.0000
84	0	0	0	0	1.0000
85	0	0	0	0	1.0000
86	0	0	0	0	1.0000
87	0	0	0	0	1.0000
88	0	0	0	0	1.0000
89	0	0	0	0	1.0000
90	0	0	0	0	1.0000
91	0	0	0	0	1.0000
92	0	0	0	0	1.0000
93	0	0	0	0	1.0000
94	0	0	0	0	1.0000
95	0	0	0	0	1.0000

References

1. https://en.wikipedia.org/wiki/CUSUM
2. http://mathworld.wolfram.com/ExponentialMovingAverage.html
3. https://arxiv.org/abs/0710.3742