Crash Report Accumulation During Continuous Fuzzing with CASR

Ilya Yegorov

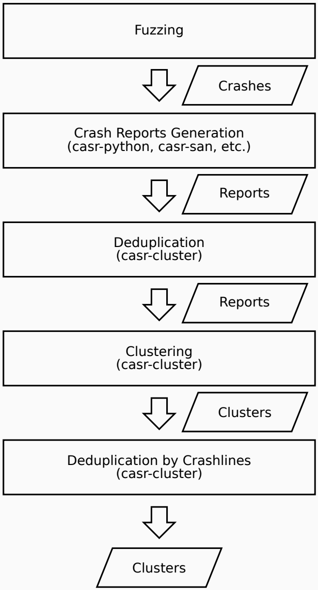
May 17, 2024

ISP RAS

Motivation

- During continuous fuzzing we get a lot of crashes
- Many of new reports are similar (i.e. belong the same cluster) or duplicate old ones
- It will be very useful to automatically remove duplicates and recognize similar crashes to minimize analysis work

Current CASR Approach to Crash Analysis

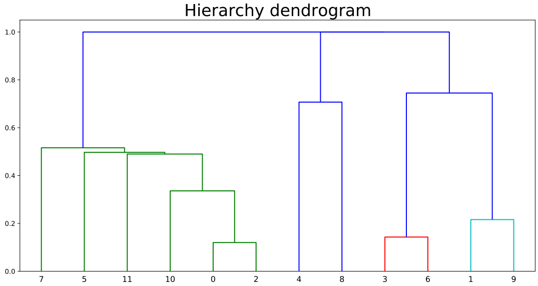


Clustering Algorithm

The casr-cluster tool performs a deduplication algorithm on the crash reports, primarily based on the stack traces.

$$dist(CL_i, CL_j) = \max_{a \in CL_i, b \in CL_j}(dist(a, b))$$

Hierarchical clustering is started based on the distance matrix obtained in the first stage. The distance between two clusters is defined as the maximum of the pairwise distance between crashes retrieved from the two clusters.



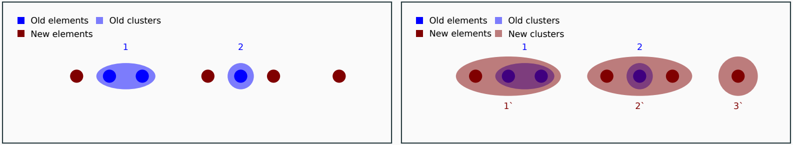
Formulation of the Problem

Adding new crashes we want to save old clustering structure, i.e.

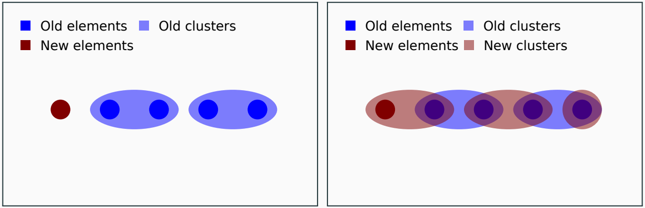
- We may not change, move or remove old crash reports
- We may not remove or reduce old crash report clusters
- We may add new crashes to old clusters or create new ones
- We may not add new crashes duplicating some old one

First Possible Approach: Reclustering

We cannot just recluster new and old crash reports, because we may lose old clustering structure, but we can try to merge old clusters with recluster ones:

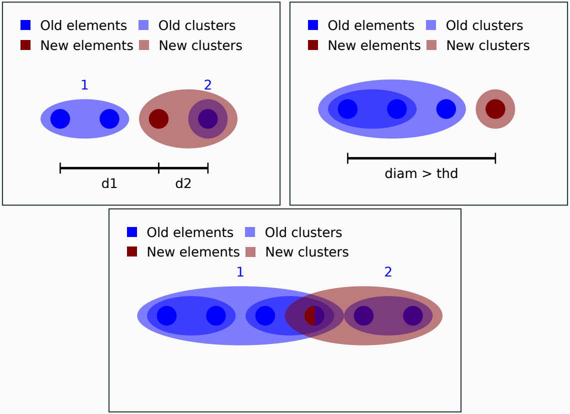


Unfortunately, clustering algorithm is sensitive to input data, i.e. resulting clusters can differ significantly when adding just one element:



Second Possible Approach: Minimal Diameter

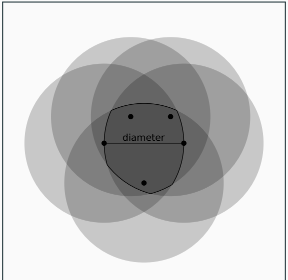
- We can add crash report to cluster with minimal resulting diameter, if it isn't superior to the threshold
- All the remaining ones are clustered to new clusters



Suggested Approach

Let's divide traces of new reports into several groups:

- $trace \in Dup(Cluster) \overset{def}{\iff}$
  $\exists trace' \in Cluster : dist(trace, trace') = 0$
- $trace \in Inner(Cluster) \overset{def}{\iff}$
  $trace \notin Dup(Cluster) \wedge diam(Cluster \cup \{trace\}) = diam(Cluster)$
- $trace \in Outer(Cluster) \overset{def}{\iff}$
  $diam(Cluster) < diam(Cluster \cup \{trace\}) < THRESHOLD$
- $trace \in Oot(Cluster) \overset{def}{\iff} diam(Cluster \cup \{trace\}) \geq THRESHOLD$



Dealing with Groups

Add some support sets:

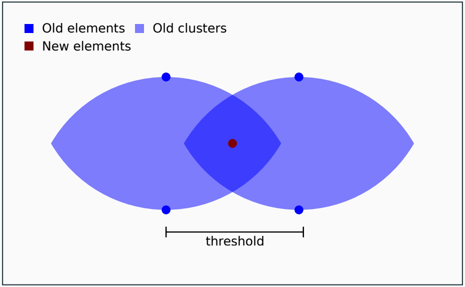
- $Dup = \{trace | \exists Cluster \in Clusters : trace \in Dub(Cluster)\}$
- $Inners(trace) = \{Cluster | trace \in Inner(Cluster)\}$
- $Outers(trace) = \{Cluster | trace \in Outer(Cluster)\}$
- $OOT = \{trace | \forall Cluster \in Clusters : trace \in Oot(Cluster)\}$

it's obvious that $\forall trace :$
$(trace \in Dup \vee Inners(trace) \neq \emptyset \vee Outers(trace) \neq \emptyset) \oplus trace \in OOT$
Dealing with Groups:

- If $newtrace \in Dup \Rightarrow$ just shed it out
- Else if $Inners(newtrace) \neq \emptyset \Rightarrow$
  add new trace in some $Cluster \in Inners(newtrace)$
- Else if $Outers(newtrace) \neq \emptyset \Rightarrow$
  add new trace in some $Cluster \in Outers(newtrace)$
- Cluster traces from OOT and add result as new clusters

How to Choose Cluster for Adding

What if $|Inners(newtrace)| > 1$ or $|Outers(newtrace)| > 1$?



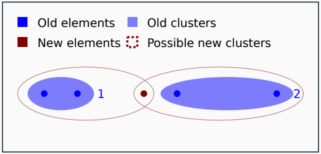
We considered several ways to choose cluster for adding:

1. Diam: $\underset{Cluster \in Clusters}{Argmin} \; diam(Cluster \cup \{trace\})$
2. Delta: $\underset{Cluster \in Clusters}{Argmin} \; |diam(Cluster \cup \{trace\}) - diam(Cluster)|$
3. Dist: $\underset{Cluster \in Clusters}{Argmin} \; dist(Cluster, \{trace\})$

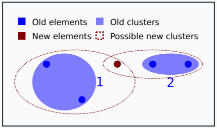
Maybe we can choose more general from them?

The Most General Condition

Unfortunately, there is no more general condition:



$diam_1 = 1, diam'_1 = 3, \Delta_1 = 2, dist_1 = 2$
$diam_2 = 3, diam'_2 = 4, \Delta_2 = 1, dist_2 = 1$
$diam'_1 < diam'_2 \wedge \Delta_1 > \Delta_2 \Rightarrow Diam \not\Rightarrow Delta \wedge Diam \not\Leftarrow Delta$
$diam'_1 < diam'_2 \wedge dist_1 > dist_2 \Rightarrow Diam \not\Rightarrow Dist \wedge Diam \not\Leftarrow Dist$



$diam_1 = \sqrt{2} \approx 1.4, diam'_1 = 2, \Delta_1 = 2 - \sqrt{2} \approx 0.6, dist_1 = \sqrt{2} \approx 1.4$
$diam_2 = 1, diam'_2 = 2, \Delta_2 = 1, dist_2 = 1$
$\Delta_1 < \Delta_2 \wedge dist_1 > dist_2 \Rightarrow Delta \not\Rightarrow Dist \wedge Delta \not\Leftarrow Dist$

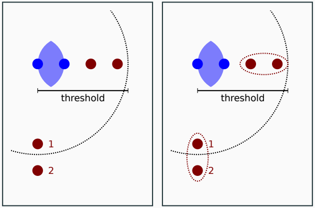
Silhouette scores

Target	Report number	Unique crashlines	Clustering	Inner strategy: Diam / Outer strategy: Diam	Inner strategy: Diam / Outer strategy: Delta	Inner strategy: Diam / Outer strategy: Dist	Inner strategy: Dist / Outer strategy: Diam	Inner strategy: Dist / Outer strategy: Delta	Inner strategy: Dist / Outer strategy: Dist
class_parser (pytorch)	115	no	0.441265337	-0.047098	-0.041209	-0.043204	-0.046231	-0.041209	-0.043204
		yes	0.112159166	-0.127086	-0.115775	-0.097212	-0.127086	-0.115775	-0.097212
load (xlnt)	29	no	0.283660748	0.149053	0.139211	0.155019	0.149053	0.139211	0.155019
		yes	0.166276934	0.029686	0.021481	0.029686	0.029686	0.021481	0.029686
load-afl++ (xlnt)	34	no	0.43487928	-0.020831	-0.020831	-0.020831	-0.020831	-0.020831	-0.020831
		yes	0.204925503	-0.085788	-0.085788	-0.085788	-0.085788	-0.085788	-0.085788
yaml_fuzzer (ruamel-yaml)	11	no	0.491333771	0.450171	0.409008	0.409008	0.450171	0.409008	0.409008
		yes	0	0	0	0	0	0	0
webp-afl++ (image-rs)	16	no	0.553874403	0.502958	0.502958	0.502958	0.502958	0.502958	0.502958
		yes	0.288377452	0.282653	0.282653	0.282653	0.282653	0.282653	0.282653
DenylistFuzzer (json-sanitizer)	12	no	0.217716308	0.208908	0.208908	0.208908	0.208908	0.208908	0.208908
		yes	0	0	0	0	0	0	0
parse_model-afl++ (onnx)	38	no	0.512385945	0.419341	0.418342	0.419484	0.419341	0.418342	0.419484
		yes	0	0.171795	0.185153	0.185153	0.171795	0.185153	0.185153
check_model-afl++ (onnx)	90	no	0.447840753	0.195523	0.195523	0.194373	0.197061	0.197061	0.197061
		yes	0.348014746	0.142245	0.142245	0.142245	0.142245	0.142245	0.142245
FuzzHLOParseUnverified-afl++ (tensorflow)	38	no	0.537232026	0.415701	0.415701	0.415701	0.415701	0.415701	0.415701
		yes	0	0	0	0	0	0	0
encode_jpeg-afl++ (torchvision)	19	no	0.263499931	0.249396	0.255729	0.255729	0.249396	0.255729	0.255729
		yes	0	0.023622	0.023622	0.023622	0.023622	0.023622	0.023622

Negative	The Best on Target

Development of the Idea

Using Silhouette score for clustering estimation we can get negative score:



Points 1 and 2 will belong different clusters! $\Rightarrow$

Got the idea of *Tolerance Level*, i.e. how tolerant are old clusters to new traces by manipulating *outer* traces: current idea — *Loyal Level*, suggested — *Hard Level*:

Add new traces only in clusters from $Inners(newtrace)$;

Non *Dup* traces with empty *Inners* and non empty *Outers* cluster with traces from OOT. Thus, further *Outer* means both *Outer* and OOT.

Hard Level Results

Silhouette scores

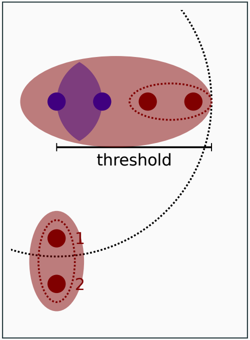
Target	Report number	Unique crashlines	Clustering	Inner strategy	
				Diam	Dist
class_parser (pytorch)	115	no	0.441265337	0.020529	0.020529
		yes	0.112159166	-0.079201	-0.079201
load (xlnt)	29	no	0.283660748	0.154005	0.154005
		yes	0.166276934	0.018443	0.018443
load-afl++ (xlnt)	34	no	0.43487928	0.018443	0.018443
		yes	0.204925503	0.043763	0.043763
yaml_fuzzer (ruamel-yaml)	11	no	0.491333771	0.350809	0.350809
		yes	0	0	0
webp-afl++ (image-rs)	16	no	0.553874403	0.380594	0.380594
		yes	0.288377452	0.214926	0.214926
DenylistFuzzer (json-sanitizer)	12	no	0.217716308	0.103437	0.103437
		yes	0	0	0
parse_model-afl++ (onnx)	38	no	0.512385945	0.362626	0.362626
		yes	0	-0.006146	-0.006146
check_model-afl++ (onnx)	90	no	0.447840753	0.165282	0.164609
		yes	0.348014746	0.052373	0.052373
FuzzHLOParseUnverified-afl++ (tensorflow)	38	no	0.537232026	0.16577	0.16577
		yes	0	0	0
encode_jpeg-afl++ (torchvision)	19	no	0.263499931	0.145461	0.145461
		yes	0	0	0

Negative	The Best on Target

Development of the Idea

With *Hard Level* resulting clusters turn out very small ⇒ Silhouette score is near null. But we can combine *Hard Level* approach with *cluster merge* approach — *Soft Level Approach*:

1. Hard Level old clusters updating (only for *Inner* traces)
2. Clustering *Outer* traces
3. Trying to merge new clusters to old ones



Soft Level Results

Silhouette scores

Target	Report number	Unique crashlines	Clustering	Inner strategy	
				Diam	Dist
class_parser (pytorch)	115	no	0.441265337	0.132457	0.132457
		yes	0.112159166	0.06883	0.06883
load (xlnt)	29	no	0.283660748	0.231124	0.231124
		yes	0.166276934	0.09904	0.09904
load-afl++ (xlnt)	34	no	0.43487928	0.37076	0.37076
		yes	0.204925503	0.205429	0.205429
yaml_fuzzer (ruamel-yaml)	11	no	0.491333771	0.471911	0.471911
		yes	0	0	0
webp-afl++ (image-rs)	16	no	0.553874403	0.502958	0.502958
		yes	0.288377452	0.292402	0.292402
DenylistFuzzer (json-sanitizer)	12	no	0.217716308	0.208908	0.208908
		yes	0	0	0
parse_model-afl++ (onnx)	38	no	0.512385945	0.434943	0.434943
		yes	0	0.135349	0.135349
check_model-afl++ (onnx)	90	no	0.447840753	0.189364	0.189731
		yes	0.348014746	0.116754	0.116754
FuzzHLOParseUnverified-afl++ (tensorflow)	38	no	0.537232026	0.433882	0.433882
		yes	0	0	0
encode_jpeg-afl++ (torchvision)	19	no	0.263499931	0.251405	0.251405
		yes	0	0.044593	0.044593

Negative	The Best on Target

Hierarchical Approach

Alternative way to deal with *Outer* traces:

1. Hard Level old clusters updating (only for *Inner* traces)
2. Clustering *Outer* traces with old clusters as single elements to new clusters
3. Save each *Outer* trace:
   - New cluster containing the trace also contains some old cluster $\Rightarrow$ add the trace to the old one
   - New cluster containing the trace contains only new traces $\Rightarrow$ save the cluster as new one

Hierarchical Approach Results

Silhouette scores

Target	Report number	Unique crashlines	Clustering	Inner strategy	
				Diam	Dist
class_parser (pytorch)	115	no	0.441265337	0.338481	0.338481
		yes	0.112159166	0.10546	0.10546
load (xlnt)	29	no	0.283660748	0.27649	0.27649
		yes	0.166276934	0.143009	0.143009
load-afl++ (xlnt)	34	no	0.43487928	0.434879	0.434879
		yes	0.204925503	0.213924	0.213924
yaml_fuzzer (ruamel-yaml)	11	no	0.491333771	0.471911	0.471911
		yes	0	0	0
webp-afl++ (image-rs)	16	no	0.553874403	0.502958	0.502958
		yes	0.288377452	0.274624	0.274624
DenylistFuzzer (json-sanitizer)	12	no	0.217716308	0.213312	0.213312
		yes	0	0	0
parse_model-afl++ (onnx)	38	no	0.512385945	0.433054	0.433054
		yes	0	0.125361	0.125361
check_model-afl++ (onnx)	90	no	0.447840753	0.432131	0.43472
		yes	0.348014746	0.344663	0.344663
FuzzHLOParseUnverified-afl++ (tensorflow)	38	no	0.537232026	0.470631	0.470631
		yes	0	0	0
encode_jpeg-afl++ (torchvision)	19	no	0.263499931	0.251405	0.251405
		yes	0	0.02116	0.02116

Negative	The Best on Target

Conclusion

- The method allows automatically accumulate new crash reports
- The method allows automatically triage new crash reports
- The method allows the use during continuous integration fuzzing process
- The method has been implemented as a part of CASR system

Automatic accumulation with proposed method allows you to avoid wasting time analysis new variants of old crashes.

Questions?