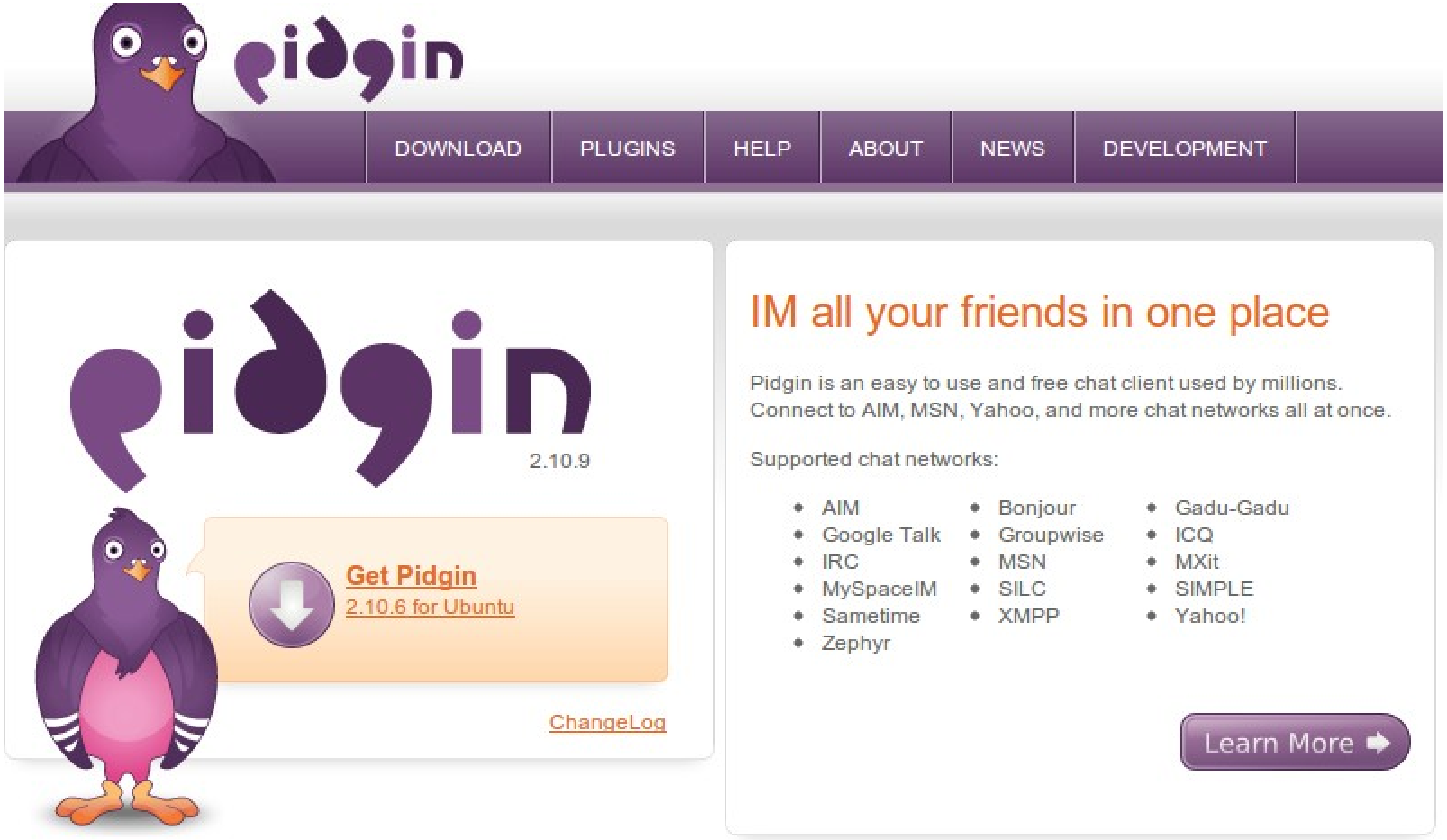
Pidgin Crasher
Searching for Minimised Crashing GUI Event Sequences

Haitao Dan, Mark Harman, Jens Krinke, Lingbo Li, Alexandru Marginean and Fan Wu



IM all your friends in one place

Pidgin is an easy to use and free chat client used by millions. Connect to AIM, MSN, Yahoo, and more chat networks all at once.

Supported chat networks:

- AIM
- Google Talk
- IRC
- MySpaceIM
- Sametime
- Zephyr
- Bonjour
- Groupwise
- MSN
- SILC
- XMPP
- Gadu-Gadu
- ICQ
- MXit
- SIMPLE
- Yahoo!

In total, More than 350KLoC C code

Among them, 76KLoC of GUI program (under pidgin directory)

Overview

- An answer to SSBSE 2014 challenge track
  - Around 2000 lines of code
  - A GUI testing framework with playback and reduction
  - Four algorithms: two for the crashing sequence generation, two for the crashing sequence reduction
  - In the experiments with ***Pidgin***, more than 3000 crashing sequences generated, 600 reduced
- Results
  - The search-based test generation achieved better crashing sequences in terms of effectiveness and efficiency
  - Three different types of 20 different crashing points identified
  - Crashing sequences were reduced with an average reduction factor of 4.88-7.50

Agenda

- Motivations
- The testing framework
- Test generation: **blocked-random** and **greedy**
- Test reduction
- Research questions and experiments
- Conclusions and future work

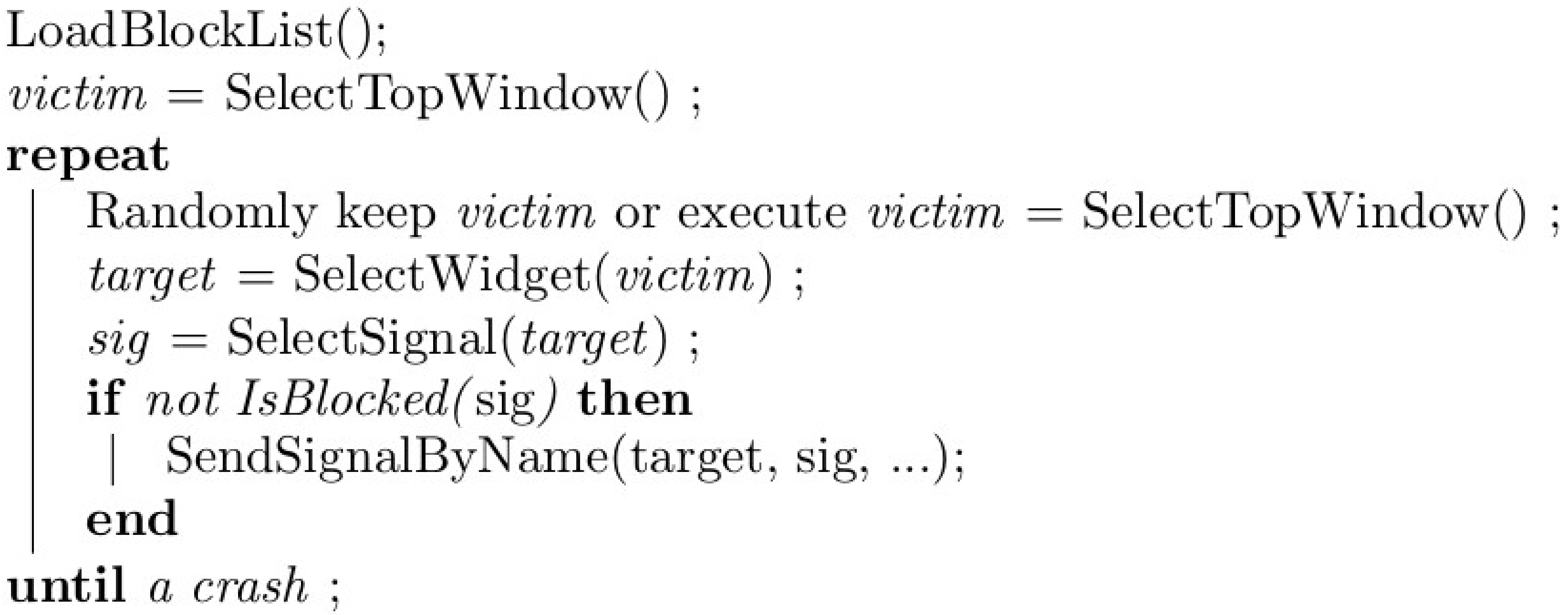
Motivations

- A spin-off project trying to drive tests for ***Pidgin*** with a simple test framework
- GUI testing techniques is lag behind [1]
  - GUI bugs account for most bugs in the GUI program (52.7%) and around a third of crashes
  - Few studies on applying SBSE on GUI testing
- A big portion of bugs that end up with crashes (18.4–22.0%)
- Focus on identify crashing bugs with complex interleavings of the events away from main scenarios

A Special GUI Testing Framework

- Instead of events, we send **signals**
  - Using API function: `g_signal_emit_by_name`
- Target only crashing behaviour, so it does not require a test oracle [2]
- On-the-fly GUI testing which does not need a behaviour model

Test Gen. – Random Blocked



```
LoadBlockList();
victim = SelectTopWindow() ;
repeat
    Randomly keep victim or execute victim = SelectTopWindow() ;
    target = SelectWidget(victim) ;
    sig = SelectSignal(target) ;
    if not IsBlocked(sig) then
        SendSignalByName(target, sig, ...);
    end
until a crash ;
```

Test Gen. – Greedy Search

- Use the previous crashing sequences to guide the selection of new signals to avoid previously discovered crashing points:
  - We select the next signal by computing the furthest **Levenshtein** distance between the current sequence and all previous sequences
- **Levenshtein** distance:
  - A string metric measuring the difference between two sequences.

Crashing Sequence Reduction

- A simple approach applied:
  - Given a crashing sequence *S*, the neighbourhood of it is defined as all the sequences generated by removing one signal from *S*.
  - A sequence *S1* is evaluated as better than another sequence *S2* if and only if *S2* crashes the subject program and is shorter than *S1*.
  - Keep exploring the neighbourhood, until we reach an optimal.

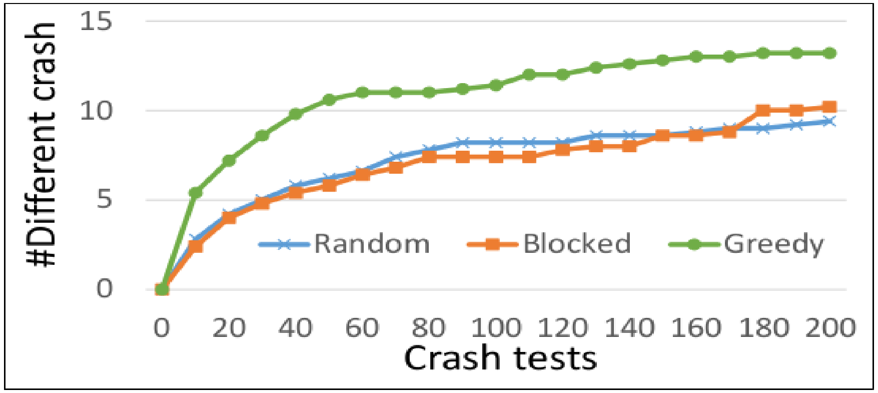
Experiment Settings

- Run ***Pidgin crasher*** in its three different modes: *Random*, *Blocked*, *Greedy* search
- Generate 201 crashing sequences in each mode and repeat our experiments 5 times
- The sequences are then minimised by the reduction process
- We repetitively send signals to trigger different functionalities via the same GTK signal emission API to which we pass NULL for all arguments in the *variable argument list*

Research Questions

- **RQ1** How effectively can ***Pidgin crasher*** find potential bugs?
- **RQ2** What are the coverage of crashing points, convergence and redundancy of the sequences generated by each of the three modes of ***Pidgin crasher***?
- **RQ3** What are the kinds of faults found by ***Pidgin crasher*** ?

Experiment Results

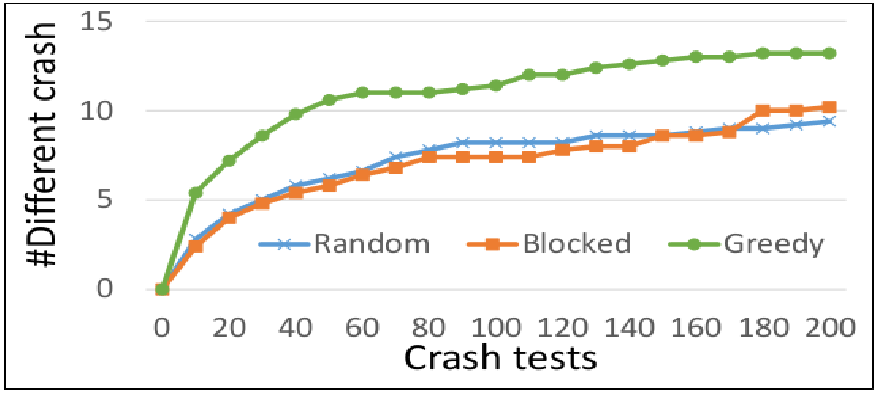


	Avg	Min	Max	Factor
Random	14.5	1	131	4.88
Blocked	58.4	1	673	7.50
Greedy	17.5	1	135	5.91

Crashed Function	Widget	Signal	Location	Library	#Crash			Type
					Rnd	Blk	Grd	
add_room_to_blist_cb	GtkLabel	move-cursor	gtkroomlist.c:250	Pidgin	1	1	3	2
gtk_editable_insert_text	GtkEntry	insert-at-cursor	gtkeditable.c:170	GTK	0	2	13	1
gtk_label_activate_link	GtkLabel	activate-link	gtklabel.c:5838	GTK	45	116	206	1
gtk_menu_set_child_property	GtkMenu	move-scroll	gtkmenu.c:926	GTK	0	1	0	1
gtk_notebook_real_switch_page	GtkNotebook	switch-page	gtknotebook.c:6142	GTK	20	39	52	1
gtk_path_bar_scroll_down	GtkMenu	move-scroll	gtkpathbar.c:803	GTK	0	0	1	2
gtk_path_bar_scroll_down	GtkButton	clicked	gtkpathbar.c:803	GTK	0	0	1	2
gtk_real_menu_item_toggle_size_request	GtkMenuItem	toggle-size-request	gtkmenuitem.c:1452	GTK	811	681	435	1
gtk_tree_model_get_valist	GtkTreeView	row-activated	gtktreemodel.c:1470	GTK	5	12	11	2
join_button_cb	GtkMenuItem	activate	gtkroomlist.c:265	Pidgin	0	0	1	2
join_button_cb	GtkButton	clicked	gtkroomlist.c:265	Pidgin	0	0	1	2
location_button_toggled_cb	GtkToggleButton	toggled	gtkfilechooserdefault.c:4662	GTK	0	0	1	1
menu_add_pounce_cb	GtkMenuItem	activate	gtkconv.c:1169	Pidgin	4	14	41	2
menu_add_pounce_cb	GtkMenuItem	activate-item	gtkconv.c:1169	Pidgin	6	18	51	2
menu_invite_cb	GtkMenuItem	activate	gtkconv.c:1250	Pidgin	12	7	46	2
menu_invite_cb	GtkMenuItem	activate-item	gtkconv.c:1250	Pidgin	12	5	49	2
purple_blist_node_get_type	GtkTreeView	row-collapsed	blist.c	Pidgin	0	0	1	1
purple_blist_node_set_bool	GtkTreeView	row-collapsed	blist.c	Pidgin	0	0	3	2
regenerate_options_items	GtkMenuItem	activate-item	gtkconv.c:3343	Pidgin	49	52	43	2
regenerate_options_items	GtkMenuItem	activate	gtkconv.c:3343	Pidgin	40	57	46	2

The coverage of crashing points: 11 13 19

Experiment Results



	Avg	Min	Max	Factor
Random	14.5	1	131	4.88
Blocked	58.4	1	673	7.50
Greedy	17.5	1	135	5.91

Crashed Function	**Widget**	**Signal**	**Location**	**Library**	**#Crash**			**Type**
					Rnd	**Blk**	**Grd**	
add_room_to_blist_cb	GtkLabel	move-cursor	gtkroomlist.c:250	Pidgin	1	1	3	2
gtk_editable_insert_text	GtkEntry	insert-at-cursor	gtkeditable.c:170	GTK	0	2	13	1
gtk_label_activate_link	GtkLabel	activate-link	gtklabel.c:5838	GTK	45	116	206	1
gtk_menu_set_child_property	GtkMenu	move-scroll	gtkmenu.c:926	GTK	0	1	0	1
gtk_notebook_real_switch_page	GtkNotebook	switch-page	gtknotebook.c:6142	GTK	20	39	52	1
gtk_path_bar_scroll_down	GtkMenu	move-scroll	gtkpathbar.c:803	GTK	0	0	1	2
gtk_path_bar_scroll_down	GtkButton	clicked	gtkpathbar.c:803	GTK	0	0	1	2
gtk_real_menu_item_toggle_size_request	GtkMenuItem	toggle-size-request	gtkmenuitem.c:1452	GTK	811	681	435	1
gtk_tree_model_get_valist	GtkTreeView	row-activated	gtktreemodel.c:1470	GTK	5	12	11	2
join_button_cb	GtkMenuItem	activate	gtkroomlist.c:265	Pidgin	0	0	1	2
join_button_cb	GtkButton	clicked	gtkroomlist.c:265	Pidgin	0	0	1	2
location_button_toggled_cb	GtkToggleButton	toggled	gtkfilechooserdefault.c:4662	GTK	0	0	1	1
menu_add_pounce_cb	GtkMenuItem	activate	gtkconv.c:1169	Pidgin	4	14	41	2
menu_add_pounce_cb	GtkMenuItem	activate-item	gtkconv.c:1169	Pidgin	6	18	51	2
menu_invite_cb	GtkMenuItem	activate	gtkconv.c:1250	Pidgin	12	7	46	2
menu_invite_cb	GtkMenuItem	activate-item	gtkconv.c:1250	Pidgin	12	5	49	2
purple_blist_node_get_type	GtkTreeView	row-collapsed	blist.c	Pidgin	0	0	1	1
purple_blist_node_set_bool	GtkTreeView	row-collapsed	blist.c	Pidgin	0	0	3	2
regenerate_options_items	GtkMenuItem	activate-item	gtkconv.c:3343	Pidgin	49	52	43	2
regenerate_options_items	GtkMenuItem	activate	gtkconv.c:3343	Pidgin	40	57	46	2

The coverage of crashing points: 11 13 19

Different Crashes

- Type I: happening in the call-back function directly uses a NULL-pointer from the passed arguments to access memory without checking to ensure it is non-NULL.
- Type II: happening in call-back functions that makes an invalid assumption about the resources available in the current state.

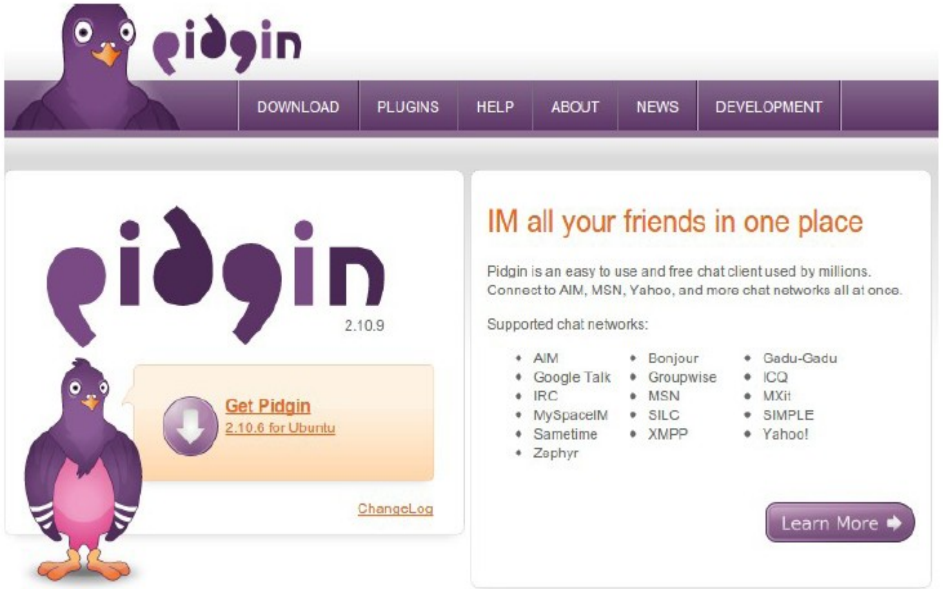
Conclusions

- Using ***Pidgin crasher***, we identified three types of 20 different crashing points.
- Suggestions:
  - Check all ***Pidgin*** return values from any function that may return NULL-pointers;
  - GTK+ signal-emitting APIs that take variable argument lists such as `g_signal_emit_by_name` should be deprecated.

Future Work

- Further analysis to the crashing points
- Use realistic input and generate more interesting crashing sequences
- Characterise the crashing sequences
- Classify GUI bugs

Motivations



IM all your friends in one place

Pidgin is an easy to use and free chat client used by millions. Connect to AIM, MSN, Yahoo, and more chat networks all at once.

Supported chat networks:

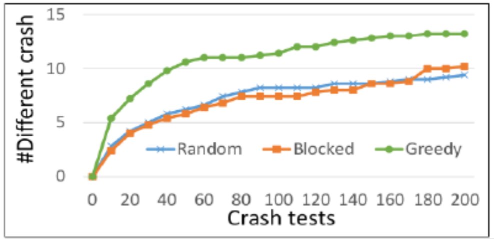
- AIM
- Google Talk
- IRC
- MySpaceIM
- Sametime
- Zephyr
- Bonjour
- Groupwise
- MSN
- SILC
- XMPP
- Gadu-Gadu
- ICQ
- MXit
- SIMPLE
- Yahoo!

In total, More than 350KLoC C code

Among them, 76KLoC for GUI program (under pidgin directory)

- A spin-off project trying to drive tests for ***Pidgin*** with a simple test framework
- GUI bugs account for most bugs and one third of crashes in GUI programs but GUI testing techniques is lag behind [1]
- Focus on the complex interleaving of the events away from main scenarios

Experiment Results



	Avg	Min	Max	Factor
Random	14.5	1	131	4.88
Blocked	58.4	1	673	7.50
Greedy	17.5	1	135	5.91

Crashed Function	Widget	Signal	Location	Library	#Crash			Type
					Rnd	Blk	Grd	
add_room_to_blist_cb	GtkLabel	move-cursor	gtkroomlist.c:250	Pidgin	1	1	3	2
gtk_editable_insert_text	GtkEntry	insert-at-cursor	gtkeditable.c:170	GTK	0	2	13	1
gtk_label_activate_link	GtkLabel	activate-link	gtklabel.c:5838	GTK	45	116	206	1
gtk_menu_set_child_property	GtkMenu	move-scroll	gtkmenu.c:926	GTK	0	1	0	1
gtk_notebook_real_switch_page	GtkNotebook	switch-page	gtknotebook.c:6142	GTK	20	39	52	1
gtk_path_bar_scroll_down	GtkMenu	move-scroll	gtkpathbar.c:803	GTK	0	0	1	2
gtk_path_bar_scroll_down	GtkButton	clicked	gtkpathbar.c:803	GTK	0	0	1	2
gtk_real_menu_item_toggle_size_request	GtkMenuItem	toggle-size-request	gtkmenuitem.c:1452	GTK	811	681	435	1
gtk_tree_model_get_valist	GtkTreeView	row-activated	gtktreemodel.c:1470	GTK	5	12	11	2
join_button_cb	GtkMenuItem	activate	gtkroomlist.c:265	Pidgin	0	0	1	2
join_button_cb	GtkButton	clicked	gtkroomlist.c:265	Pidgin	0	0	1	2
location_button_toggled_cb	GtkToggleButton	toggled	gtkfilechooserdefault.c:4662	GTK	0	0	1	1
menu_add_pounce_cb	GtkMenuItem	activate	gtkconv.c:1169	Pidgin	4	14	41	2
menu_add_pounce_cb	GtkMenuItem	activate-item	gtkconv.c:1169	Pidgin	6	18	51	2
menu_invite_cb	GtkMenuItem	activate	gtkconv.c:1250	Pidgin	12	7	46	2
menu_invite_cb	GtkMenuItem	activate-item	gtkconv.c:1250	Pidgin	12	5	49	2
purple_blist_node_get_type	GtkTreeView	row-collapsed	blist.c	Pidgin	0	0	1	1
purple_blist_node_set_bool	GtkTreeView	row-collapsed	blist.c	Pidgin	0	0	3	2
regenerate_options_items	GtkMenuItem	activate-item	gtkconv.c:3343	Pidgin	49	52	43	2
regenerate_options_items	GtkMenuItem	activate	gtkconv.c:3343	Pidgin	40	57	46	2

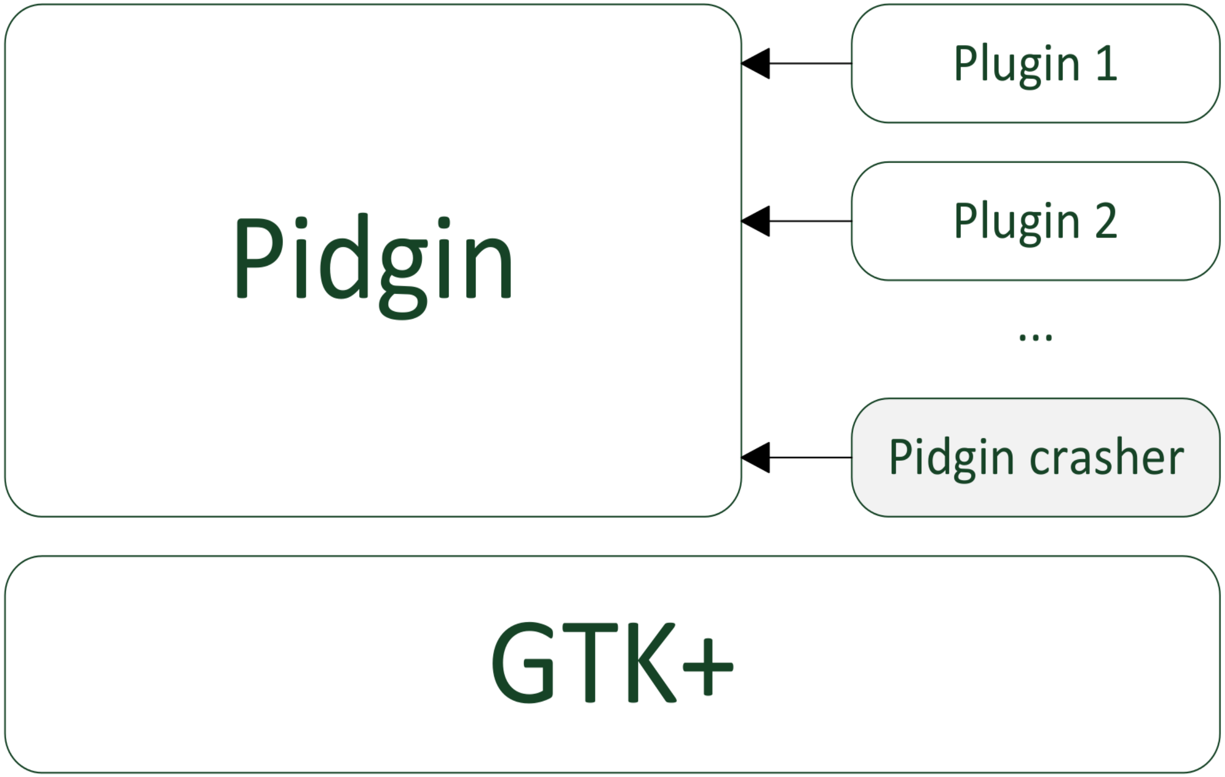
Conclusions

- Using ***Pidgin crasher***, we identified three types of bugs found caused by 20 different UI signals.
- Suggestions:
  - We suggest to check all ***Pidgin*** return values from any function that may return NULL-pointers;
  - and that GTK+ signal-emitting APIs that take variable argument lists such as `g_signal_emit_by_name` should be deprecated.

References

- [1] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai. Have things changed now?: an empirical study of bug characteristics in modern open source software. In Proceedings of the 1st workshop on Architectural and system support for improving software dependability (ASID'06), 2006.
- [2] Mark Harman, Phil McMinn, Muzammil shahbaz, and Shin Yoo. A comprehensive survey of trends in oracles for software testing. Technical Report Research Memoranda CS-13-01, Department of Computer Science, University of Sheffield, 2013.

The Test Framework



Pidgin

Plugin 1

Plugin 2

...

Pidgin crasher

GTK+

Test Generation

Targeting complex signal interleavings that are unlikely to be experienced in general use to discover crashing sequences

Test Gen. – Greedy

- The algorithm: formally,

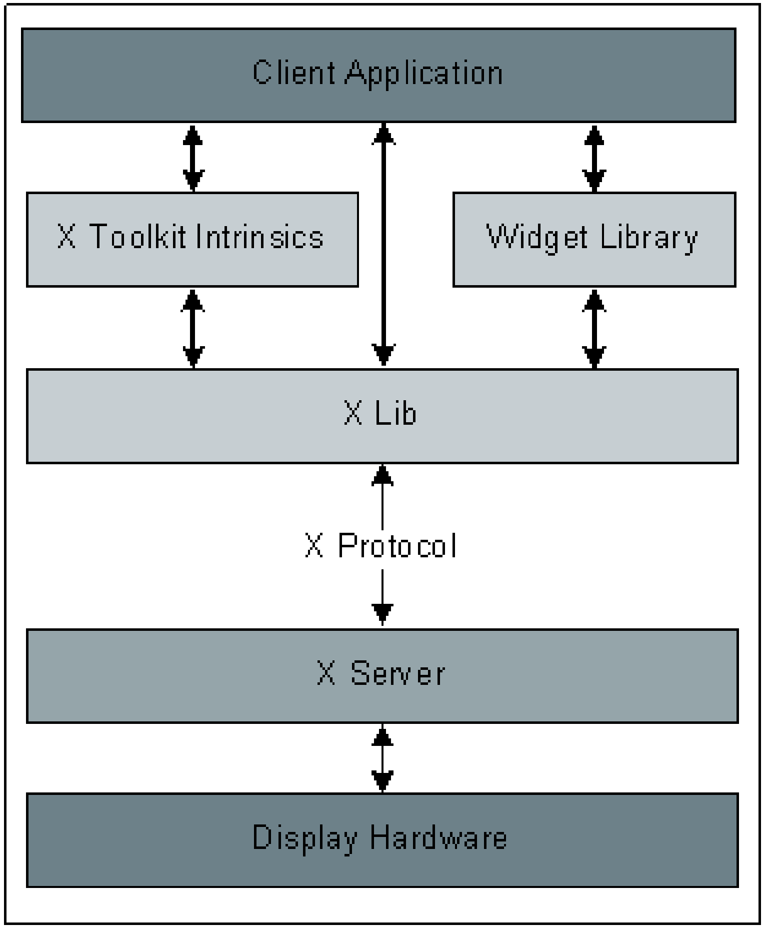
$$\forall_{s_i \in \mathcal{S}} : M(\mathcal{P}, s_1 ... s_k s_i) \leq M(\mathcal{P}, s_1 ... s_k s_{k+1})$$

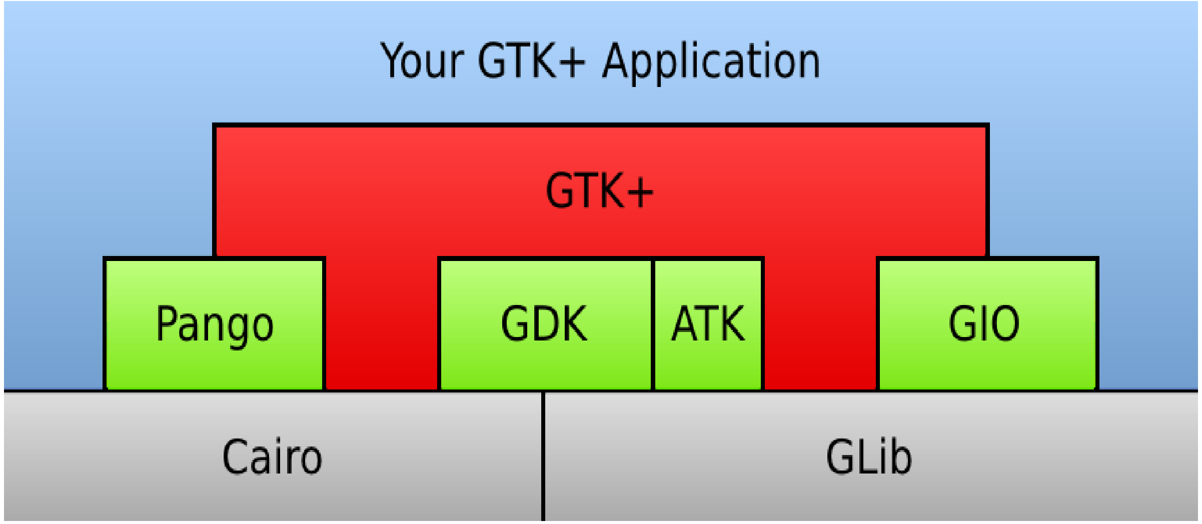
Where $\mathcal{P} = \{S_1, ... S_n\}$ is the set of previous crashing sequences,

$M(\mathcal{P}, S) = \min_{S_i \in \mathcal{P}} \{D(S_i, S)\}$

$D(x, y)$ is the **Levenshtein** distance between x and y

GTK and XWindows


Client Application
X Toolkit Intrinsics
Widget Library
X Lib
X Protocol
X Server
Display Hardware


Your GTK+ Application
GTK+
Pango
GDK
ATK
GIO
Cairo
GLib