Experiments in NAVEX dataset.

	# bugs	Navex	Navex1	XSym
SQLI	detected	155	139	
	exploitable/solvable	105 (67.7%) exploitable	87 (62.6%) solvable	95 (68.3%) solvable
XSS	detected	133	172	
	exploitable/solvable	90 (67.7%) exploitable	120 (70.0 %) solvable	110 (64.0%) solvable

- Navex: Original data in Navex paper – multiple steps.
- Navex1: Apply Navex's constraint solver core on our detected bugs.
- XSym: Use KLEE to solve PHP constraints.

- Among 311 (139 + 172) bugs
  - Navex1 solved 207, XSym solved 205.
  - 48 (15.43%) bugs include built-in functions that are not supported by Navex
    - 30 are now solvable by XSym.
  - Arithmetic functions

Table 3: Statistics of function accuracy comparison between XSYM and NAVEX.

Group	Function Name	#Passed in NAVEX	#Passed in XSYM	#Tests
(1)	md5	0	4	16
	trim	4	8	8
	rtrim	4	8	8
	nl2br	4	8	8
	explode	4	8	8
	implode	4	12	12
	strip_tags	12	16	16
	htmlentities	0	8	8
	htmlspecialchars	0	8	8
(2)	strtr	4	4	4
	strstr	16	16	16
	substr	48	48	48
	is_int	8	8	8
	intval	8	8	8
	empty	8	8	8
	strpos	16	16	16
	strcmp	16	16	16
	urldecode	8	8	8
	is_integer	8	8	8
	is_numeric	8	8	8
	str_replace	32	32	32
	addslashes	8	8	8
	stripslashes	8	8	8
	escapeshellarg	8	8	8
	escapeshellcmd	8	8	8
(3)	rand	\	\	\
	uniqid	\	\	\
	mt_rand	\	\	\
	mktime	\	\	\
(4)	dbx_escape_string	16	N/A	16
	db2_escape_string	16	N/A	16
	mysql_escape_string	16	N/A	16
	mysqli_escape_string	16	N/A	16
	mysql_real_escape_string	16	N/A	16
	mysqli_real_escape_string	16	N/A	16

Table 2: Statistics of function accuracy.

Func Types	# Func	# Tests	# Passed	Proportion
String	47	296	226	76.35%
Arithmetic	21	98	79	80.61%
Others	20	120	63	52.5 %

# Tests	# Passed	Proportion
295	254	85.8 %
98	82	xxx%
120	78	65 %

Three Directions

- Symbolic execution.
  - Many built-in functions are not used in exploit generation.
- A study to suggest to combine both? Exploration.
  - Modeling functions for SE is very important, many scenarios require this. For PHP, we have Navex, for many other languages we don't have. We should propose techniques for modeling other languages. Lessons learnt here can be applied to others.
  - Built-in functions are added gradually. Even if this is for php, there will be more php built-in functions.
  - Compare generally. Good at array cases or not.
  - Goes back general SE, how accuracy.
- Prove correctness of manual modeling.
- TODOs
  - Where the design can lead to problem of current systems/tools.
  - Prove.
  - Think about examples this can be generalized.
  - Second path does not come to php bug finding.

  - Fixed-length array support.
  - Performance, human v.s. machine, explanation.
  - Reasoning built-in functions.
  - Finding new vulnerabilities, not new class.
    - Not recent versions.
  - Check the results in page 1. humanly difficult. The 30 manually checked, whether been reported somehow. Previously unknown.
  - Constraint not solvable does not mean. May not say, if a not supported by Navex, we are new. Manual verification. What's are the 30 are. Replay. Different answers == Critical.