

Automated Security Testing

Laboratory Demonstrator

Rudolf Ramler Software Competence Center Hagenberg GmbH (SCCH) rudolf.ramler@scch.at | +43 50 343 872



Background

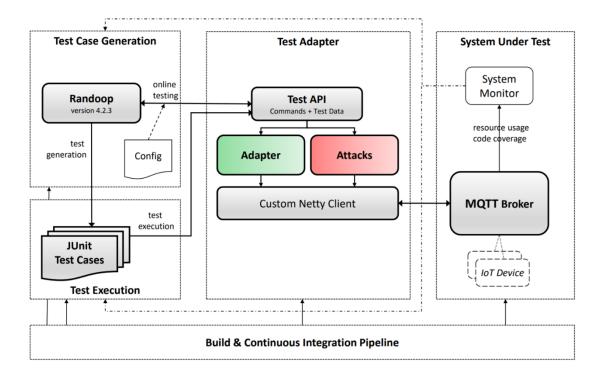
- A demonstrator for automated security testing based on attack patterns has been developed as part of the "ICT of the Future" project IoT4CPS
- The demonstrator named *MqttRazzer* is a framework for generating random tests including security attacks for or via an MQTT broker
- For further details see
 - Sochor, H., Ferrarotti, F., Ramler, R.: An Architecture for Automated Security Test Case Generation for MQTT Systems. In International Conference on Database and Expert Systems Applications (pp. 48-62). Springer, 2020.
 - Sochor, H., Ferrarotti, F., Ramler, R.: Automated security test generation for MQTT using attack patterns. In Proceedings of the 15th International Conference on Availability, Reliability and Security (pp. 1-9). ACM, 2020.

		An Architecture for Automated Security
Automated Security Test	Generation for MOTT	Test Case Generation for MQTT Systems
Using Attac		
Dame Socher Breis Br		
Software Competience Conten Software Comp	electr Online Software Comprisers Online	Hannes Socher ⁰⁰ O, Flavio Ferrarottio, and Radolf Ramlero
Hagenberg Cashil (CC34) Hagenberg Co	ebit (CC10) Hagenberg Cebit (CC130)	Software Competence Center Hagenberg,
Tagebry Auton Tagebry heres action pathal Devis Devis	Autria Togenberg Autria Egenthal rate/Frankengenthal	Boftwarepark 21, 4222 Hagenberg, Austria Danmas ancher, flavis, farraretti, radolf, ranlar Jürch, at
AIBTRACT	1 INTRODUCTION	(ARABIR. BECKET, FLATSE, FRETETES, FREELE JARCEL AN
the densative increase of attacks and mathetical activities has made	The Henney Queries Transport (#Q77) period (1, 4)	
evening a major measure in the development of international optim physical optimum and rated the need to address this resurren- tion in tening. The goal of security testing is to down are calorer	to a puppling and walkly used, lights right publication for post- red [7]. Its lay characteristics are based with officiency, data space	Abstract. Manage Queuing Televentry Transport (MQTT) protocol is
	terini, entitions to have of consectivity and completest fellows, sends design and new incoherentation. The fact that this new to	among the preferred publish/subscribe protocols used for Machine-to- Machine (MZM) communication and Internet of Things (IoT). Although
		the MQTT protocol itself is quite simple, the concurrent iteration of
have the challenge of systematically exploring a potentially nor-	a wide mage of a planting, from analise view determining pro-	broken and clients and its intrinsic non-determinism, coupled with the
muchil separa and possible barreled internation attempts to this approve description as approach for an internation of text	network on the public value to make to the model streng	diversity of platforms and programming large ages is which the postonel is implemented and run, makes the measurer task of security testing deal-
		lenging. We address precisely this problem by proposing an architecture
attenues likeptation bat on he and he waking summare of an addition. The approach confidence studies but our generation	her must so these surgers and data faits them among the climate	for security tost generation for systems relying on the MQTT protocol.
	The combody graving marker of computed devices in the	This architecture enables automated test case generation to reveal vul- nershilties and discrepancies between different implementations. As a
Transport (HCPT) protocol. We have explicit the proposal initiag represents to five preprint and widely instability of 277 Inv.	Internet of Things (107) () and the consequent increase in the need for communication between between a distributed entries had	dustrial consequences, when implemented, our architectural dusign can be
ters, providing 1.000 internation sequences in form of restantish	a strong position effort on the puppleing of the stight produced	used to unover erronous behaviours that entail latent security risks in MOTT broker and client implementations. In this paper we describe
entropy and built data in parts in indication for a section	despeed for private astronyly of the oil and gas industry. The fast	in MQTT brown and clean impairmentations, in this paper we describe the key components of our architecture, our prototypical implementation
in a soluti 2 access, whereas is not called destroying a solution of the solut	that the protocol is now privately used to bit means that the force has to shift investor reserves WICC hand is stress. As	using a random test case generator, core during devisions and the use of
		meanity attacks in tracting. Moreover, we present first evaluations of the architectural dusies and the prototypical implementation with encour-
CCS CONCEPTS Security and privacy Network anarity, Spinore and apply	rate making from the advances interpreteding with different department to advance plate of PETT locations.	saine initial rests.
an many	from stuple and formal \$1,277 participation and concepted topol.	
KI-TWORDS		Keywords: Security testing - Astonisted testing - IoT - MQTT
Investigation in the state of t	Annual Articles of the second structure and the second structure of the second	
inter peteren, Hight V.M. Editoren Turnet	N XXIII with more from 10000 inple segmentar elementary ?". In this paper we address presidely the prelident property on sy-	1 Introduction
		The Internet of Things (IoT) connects millions of devices of different orbot.
Investig The Constant in MCCTT Long Allock Palmon. In The Allo Menual and Conference in Analyticity Malability and America (All Calif.)	approach against familiants download maximum text new generation.	physical systems (CPSs). However, the increasing connectivity of eyber-physical
Report of a case for an interaction in stand ACM, New York, NY, UNA, Program	and get before these being an first and for expression button of information and implementations of the http://linkar.	objects in critical environments like industrial production, public transport, or
		energy distribution, also increases the vulnerability of these IoT-based, cyber-
formation is enderlight o best region of d as part of Birsteil for present a mean and a present a size in the period. For any set of main a desired	big charts also decomposition among them, and then you take new process for intention reporting information related in security take	physical systems. Ensuring security in such environments introduces major chal- lenges and high effort for quality assurance. Conventional approaches for qual-
n de bei pes "og sjele bronnenskov" fin met sveni 'e skan her for akterionel i kannel i bisket viter sjele bronnel i geniget framer gjele i sjele samen i state state finste bis ingeliget framer sjele gjele sjele state state state finste bis ingeliget i sjele state sta	and extended these in Mary 7 learned agreement. A distinguistic edge-statistic of the agreement we present in this	ity assurance and testing are characterized by manual tasks and script-based
		automation attempts, which limit the flexibility necessary for adapting to het-
An and the second of the second se	in a second of the same providing A link patients are likepoints	orogeneous, dynamic and self-organising IoT-based systems.
Contrast NET Contrasts View, St. W	Next Accessing by Safe annual lane (21,211,175	Adequate quality assurance and testing approaches are required to cope with a base number and variety of interaction scenarios typical for IoT systems.
		 Inger Hutther and Werkey of Histaccion Scinarios (ypical for 101 Systems,
		 Kousi et al. (Rds.): URCA 2020 Workshops, OCIS 1265, pp. 48-42, 2020. https://doi.org/10.1007/979-3-020-10020-4_1
		and a francisk of the same same and a fight



Core Components

- 1. Test case generation
- 2. Test execution
- 3. Test adapter
- 4. System under test

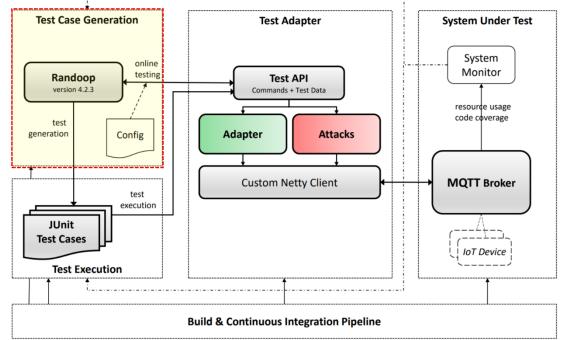




Test Case Generation

- Open source tool Randoop¹ a feed-back directed random test generator
- Randoop uses the *test adapter* to access the MQTT broker
- *Config* specifies which adapter methods are used in generating test sequences
- Randoop outputs generated sequences as *JUnit test cases*

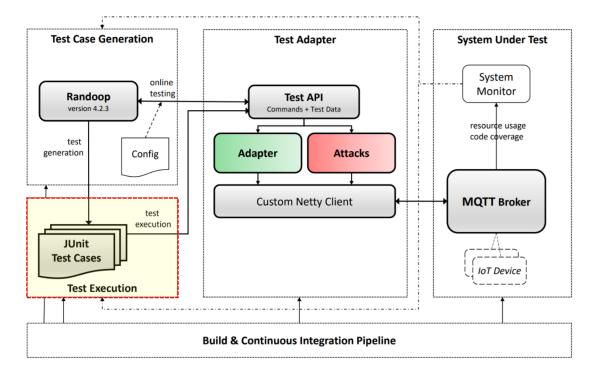
¹ https://randoop.github.io/randoop/





Test Execution

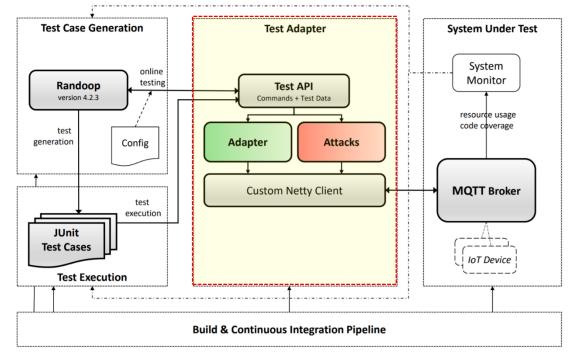
- Sequences generated by Randoop are stored as JUnit test cases
- JUnit test runner is used to execute the tests
- Tests exercise the adapter to access the MQTT broker





Test Adapter

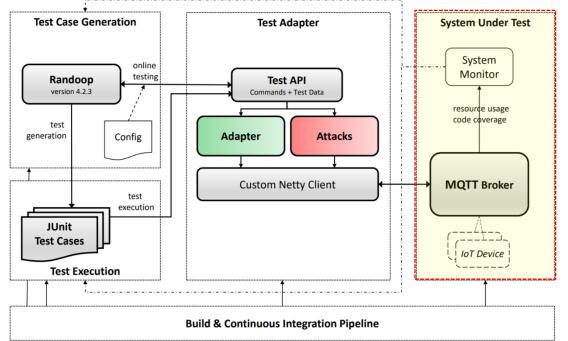
- API for interacting with an MQTT broker in testing
- The API provides
 - 1. Valid MQTT Commands
 - 2. Attacks based on invalid or malformed commands and command sequences
- A modified *Netty lib* is used to communicate with the broker; security checks have been removed to allow sending malformed/invalid data





System Under Test

- The SUT is an IoT system or device accessible via MQTT and/or an MQTT broker
- In test generation and execution, the SUT is accessed via a test adapter
- System specific monitoring (e.g. MQTT broker loggin) is optionally used to directly observe the SUT's behavior

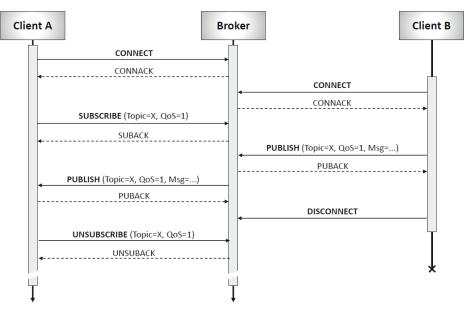




Example Generated Test Case

Covered scenario: Two clients interacting with a MQTT broker

```
@Test
=public void test01() throws Throwable {
     /* Creating Client A and connecting to broker */
     MgttClientAdapter clientA = new MgttClientAdapter();
     String strl = clientA.connectQoS0();
     assertTrue(strl.equals("MqttConnAck[/* ... */]"));
     /* Creating Client B and connecting to broker */
     MgttClientAdapter clientB = new MgttClientAdapter();
     String str2 = clientB.connectQoS0();
     assertTrue(str2.equals("MgttConnAck[/* ... */]"));
     /* Client A subscribing to topic X */
     String str3 = clientA.subscribeIntervallTopicXOoSl();
     assertTrue(str3.equals("MgttSubAck[/* ... */]"));
     /* Client B publishing to topic X and disconnecting */
     MattMsgId mattMsgIdl = clientB.publishIntervallTopicXOoS1()
     assertNotNull(mgttMsgIdl);
     clientB.disconnectQoS1();
     /* Client A receiving message and unsubscribing */
     String str4 = clientA.unsubscribeIntervallTopicQoSl();
     assertTrue(str4.equals("MottUnsubAck[/* ... */]"));
```





List of Demonstration Attacks

- Sample attacks have been derived from common attack patterns (e.g. CAPEC, CVE)
- Following attacks have been implemented as part of the test adapter

#	Description	#	Description
0	Check if the broker accepts two clients with the same id	14	connect with invalid protocol specifier (protocol="MQQT")
1	Invalid length of variable header (+1)	15	connect with invalid protocol version (protocol ver="42")
2	Invalid length of variable header (-1)		connect with bad will flag combination
4	send publish message with payload size of 128MB	10	connect with usr/pwd flag set but without giving
5	Subscribe without payload	τo	credentials
7	subscribe with invalid wildcard in topic name	19	trigger keep alive (keepAlive=1)
8	subscribe with escape sequences in topic name	20	connect with big keep alive (keepAlive=INT_MAX)
9	publish with escape sequences in payload	22	connect with invalid client identifier
10	publish with wildcards in topic name	23	subscribe with huge '/' payload
11	connect with invalid QoS (Both QoS Bits set -> QoS=3)	24	connect with empty client identifier
13	connect with long client identifier	25	connect bad Username (username UTF16 encoded)



Demonstrator: Tutorial

- Required setup for generating and running tests
 - JDK 1.8+
 - MQTT broker running (default is localhost:1883)
- Test Generation
 - Usage: mqttrazzer-gen.bat MethodList Timeout
 - Example: mqttrazzer-gen.bat etc\methods_MqttSingleClientAdapter.txt 10
- Test Execution
 - Usage: mqttrazzer-test.bat



Demonstator: Step 1 – Setup

- Mosquitto Broker running MQTT v3.1.1 broker
- Java OpenJDK 15
- Current working directory:
 c:\work\mqttrazzer

01.10.2020 0 01.10.2020 0 28.09.2020 2 01.10.2020 0 01.10.2020 0	01:08 22:20	<dir> <dir></dir></dir>			
28.09.2020 2 01.10.2020 0	22:20	<dir></dir>			
01.10.2020 0					
			50	config.properties	
		<dir></dir>		doc	
		<dir></dir>		etc	
1.10.2020 0		<dir></dir>		libs	
0.09.2020 2				LICENSE.TXT	
8.09.2020 1				MqttRazzer-1.0.0.jar	
0.09.2020 1				mqttrazzer-gen.bat	
0.09.2020 1				mqttrazzer-test.bat	
0.09.2020 1		10701	4 457	README.md	
1.10.2020 0		<dir></dir>	40	SPC	
		e(s)		260 bytes	
	6 U1P	(S) 203	880 220	6 816 bytes free	



Demonstrator: Step 2a – Test Generation

Running *mqttrazzer-gen.bat* with list of adapter methods given in *methods_MqttSingleClient.txt* for a time limit of *10* seconds

Randoop test generator is started

Log output produced by test adapter from communication with MQTT broker (commands sent and response received)

	Command Prompt	-		×
>	<pre>C:\work\mqttrazzer>mqttrazzer-gen etc\methods_MqttSingleClient.txt 10 PUBLIC MEMBERS=25 Explorer = ForwardGenerator(steps: 0, null steps: 0, num_sequences_generated: 0; allSequences: 0, regresson seqs: 0, error seqs: 0=0=0, invalid seqs: 0, subsumed, , num_failed_output_test: 0; runtimePrimitivesSeen:38)</pre>	_seq	uences	: 0
	Progress update: steps=1, test inputs generated=0, failing inputs=0 (Thu Oct 01 T 2020 24MB used)subscribeTopic0 Timeout reached	01:	04:02	CES
	pingQoS0 Timeout reached unsubscribeTopic0 Timeout reached publishReceiveQoS2 disconnectQoS2 disconnect			
	connectQoS2Mqtt31 Received CONNACK connectQoS2Mqtt31 Received CONNACK unsubscribeTopic0 Received UNSUBACK connectQoS2Mqtt31 Received CONNACK			
	unsubscribeTopic0			\sim



Demonstrator: Step 2b – Test Generation Results

Randoop test generation results; summary about explored sequences

Source files containing JUnit test cases written by Randoop

Class files after successful compilation moved to tests\bin

Average method execution time (normal termination): 131 Average method execution time (exceptional termination): NaN Approximate memory usage 23MB		
<pre>Explorer = ForwardGenerator(steps: 18, null steps: 17, num_sequences_generated allSequences: 1, regresson seqs: 1, error seqs: 0=0=0, invalid seqs: 0, sul , num failed output test: 0;</pre>		uences
runtimePrimitivesSeen:38)		
No error-revealing tests to output		
About to look for failing assertions in 1 regression sequences.		
Regression test output: Regression test count: 1		
Writing regression JUnit tests Created file C:\work\mqttrazzer\tests\src\at\scch\mqttrazzer\RegressionTest0.ja	ava	
Created file C:\work\mqttrazzer\tests\src\at\scch\mqttrazzer\RegressionTest.jav Wrote regression JUnit tests. About to look for flaky methods.	va	
Invalid tests generated: 0		
C:\work\mqttrazzer\tests\src\at\scch\mqttrazzer\RegressionTest.class C:\work\mqttrazzer\tests\src\at\scch\mqttrazzer\RegressionTest0.class 2 file(s) moved.		
C:\work\mgttrazzer>		



Demonstrator: Step 3a – Test Execution

	Command Prompt	– 🗆 X
	C:\work\mqttrazzer>dir tests\src\at\scch\mqttrazzer Volume in drive C is Windows Volume Serial Number is 6483-8755	^
	Directory of C:\work\mqttrazzer\tests\src\at\scch\mqttrazzer	
Java source files containing JUnit test cases written by Randoop	01.10.2020 01:39 <dir> . 01.10.2020 01:39 <dir> . 01.10.2020 01:39 207 RegressionTest.java 01.10.2020 01:39 5 262 RegressionTest0.java 2 File(s) 5 469 bytes 2 Dir(s) 203 879 239 680 bytes free</dir></dir>	
	C:\work\mqttrazzer>dir tests\bin\at\scch\mqttrazzer Volume in drive C is Windows Volume Serial Number is 6483-8755	
	<pre>Directory of C:\work\mqttrazzer\tests\bin\at\scch\mqttrazzer</pre>	
Class files after successful compilation ready for execution	01.10.2020 01:39 <dir> . 01.10.2020 01:39 <dir> . 01.10.2020 01:39 540 RegressionTest.class 01.10.2020 01:39 4 222 RegressionTest0.class 2 File(s) 4 762 bytes 2 Dir(s) 203 879 239 680 bytes free</dir></dir>	
	C:\work\mqttrazzer>	¥



Demonstrator: Step 3b – Test Execution Results

Batch file mqttrazzer-test.bat executing JUnit test runner

Log output showing MQTT commands and responses from broker; log produced by adapter called from executed JUnit tests

Successful execution of generated tests (i.e. no deviations found in regression test run)

Command Prompt	_	×
C:\work\mqttrazzer>mqttrazzer-test.bat		^
JUnit version 4.12		
.subscribeTopic0		
Timeout reached		
.pingQoS0		
Timeout reached		
.unsubscribeTopic0		
Timeout reached		
.publishReceiveQoS2		
.disconnectQoS2		
disconnect		
.connectQoS2Mqtt31 Received CONNACK		
unsubscribeTopic0		
Received UNSUBACK		
Time: 3,851		
OK (6 tests)		
(1 uonk) matthozzon		
C:\work\mqttrazzer>		



Evaluation Results

	Mosquitto	Moquette	ActiveMQ	emqx	VerneMQ
URL	https://mosquitto.org	https://github.com/and sel/moquette	https://activemq.apach e.org	https://www.emqx.io	https://vernemq.com
Version	1.6.8	0.13	5.15.12	4.0.6	1.10.2
Errors	reference	>500	219	198	585
Failures	reference	>300	18	64	0

Comparison of behavior of MQTT broker with reference implementation (Mosquitto) by running regression tests generated for reference on other broker implementations Analysis results¹: **28 Security relevant issues discovered**

¹ Sochor, H., Ferrarotti, F., Ramler, R.: Automated security test generation for MQTT using attack patterns. In Proceedings of the 15th International Conference on Availability, Reliability and Security (pp. 1-9). ACM, 2020.



For further information please contact:

Rudolf Ramler

Projectpartner

rudolf.ramler@scch.at

Software Competence Center Hagenberg Gmbh, Austria, https://www.scch.at



Federal Ministry Republic of Austria Climate Action, Environment, Energy, Mobility, Innovation and Technology

The IoT4CPS project is partially funded by the "ICT of the Future" Program of the FFG and the BMK.

