Overview

This is the full evaluation of various security/safety concept sets, performed as part of the SAW 2017 paper "SAFE and Secure: Deeply Integrating Security in a New Hazard Analysis"

This example is derived from / based on the example in Procter's dissertation, which can be downloaded at http://krex.k-state.edu/dspace/handle/2097/32861

Avižienis et. al based Guidewords

Activity 1: Unsafe Interactions

Step 1.1	Step 1.2							
	Manifestations							
Successor Dangers	Pred. Link	Content		Halted	Erratic	Timing	
		High	Low			Early	Late
SC.DontODPatient	AppLogicCommands -> PCA Pump	PCAPump.TicketTooLong	PCAPump.TicketTooShort	PCAPump.NoTicketSupplied	PCAPump.ErraticTicket	PCAPump.EarlyTicket	PCAPump.LateTicket
SC.DontUDPatient							

Process Variable	Process Values							Unit
Ticket Duration	1	2	3	...	598	599	600	Seconds

Step 1.3

Externally Caused Dangers						Proposed Mitigations	
Successor Danger	Name	Process Var. Name	Process Var. Value	Interpretation	Co-occurring Dangers	Run-time Detection	Run-time Handling
SC.DontODPatient	PCAPump.TicketTooLong	Ticket Duration	Higher than safe	The ticket has a time value that is too long	None	None	N / A
SC.DontUDPatient	PCAPump.TicketTooShort	Ticket Duration	Lower than safe	The ticket has a time value that is too short	None	None	N / A
SC.DontUDPatient	PCAPump.NoTicketSupplied	Ticket Duration	None	No ticket is supplied	None	None	Pump notifies the clinician
SC.DontODPatient	PCAPump.ErraticTicket	Ticket Duration	Any	The PCA Pump gets a ticket "out of the blue" when it is already running	None	Concurrent: Timeouts	Rollforward: Pump switches into permanent KVO (and notifies the clinician?)
				The PCA Pump gets a ticket "out of the blue" when it would be unsafe to run		Concurrent: Negative Ticket Values	
SC.DontODPatient	PCAPump.EarlyTicket	Ticket Duration	Any	The PCA Pump gets a ticket "too soon" -- before it has finished handling the previous ticket	None	Concurrent: Timeouts	Rollforward: Pump switches into permanent KVO (and notifies the clinician?)
SC.DontODPatient	PCAPump.LateTicket	Ticket Duration	Any	The PCA pump gets a ticket late, so it's valid past the time window it should be	None	Concurrent: Timestamped tickets	Rollforward: Pump switches into KVO

Explanations

Reference	Explanation
Concurrent: Negative Ticket Values	If the pump can't safely run for some length of time, negative-valued tickets should be sent. That is, rather than send a ticket with a time of zero, if the app will re-assess the patient's health after some period (eg, 300 seconds) it should send a ticket with a value of -300s. That way, any tickets arriving in the interim are caught as inappropriate

Dolev-Yao based Guidewords

Activity 1: Unsafe Interactions

Step 1.1	Step 1.2						
Successor Dangers	**Manifestations**						
	Pred. Link	Send New	Delay	Modify Existing	Drop	Replay	Read
SC.DontODPatient	AppLogicCommands -> PCA Pump	PCAPump.ForgedMessage	PCAPump.LateMessage	PCAPump.ForgedMessage	PCAPump.DroppedMessage	PCAPump.RepeatedMessage	PCAPump.DontLeakPHI
SC.DontUDPatient							
Terminal Dangers							
SC.DontLeakPatientInfo							

Process Variable	**Process Values**							**Unit**
Ticket Duration	1	2	3	...	598	599	600	Seconds

Step 1.3

Externally Caused Dangers						**Proposed Mitigations**	
Successor Danger	Name	Process Var. Name	Process Var. Value	Interpretation	Co-occurring Dangers	Run-time Detection	Run-time Handling
SC.DontODPatient	PCAPump.ForgedMessage	Ticket Duration	Higher than safe	A forged ticket arrives at the expected time but with a forged and higher-than-safe value	None	Concurrent: Hashed and signed messages	Rollforward: Pump switches into permanent KVO (and notifies the clinician?)
			Any	The PCA Pump gets a ticket "out of the blue" when it is already running	None	Concurrent: Timeouts	
				The PCA Pump gets a ticket "out of the blue" when it would be unsafe to run		Concurrent: Negative Ticket Values	
SC.DontODPatient	PCAPump.LateMessage	Ticket Duration	Any	The PCA pump gets a ticket late, so it's valid past the time window it should be	None	Concurrent: Timestamped tickets	Rollforward: Pump switches into KVO
SC.DontUDPatient	PCAPump.DroppedMessage	Process Variable	None	The PCA pump's incoming commands never arrive	None	None	Pump notifies the clinician
SC.DontLeakPatientInfo	PCAPump.DontLeakPHI	Ticket Duration	Any	The PCA pump's incoming commands are visible to a malicious observer	None	None	Encrypt incoming tickets

Explanations

Reference	Explanation
SC.DontLeakPatientInfo	We need to introduce a new danger: leaking patient info. It's not a successor danger, in that no action on the part of the successor component (the patient) is necessary to realize a potential loss. So, it's classified as a "Terminal" danger.

STPA-SafeSec based Guidewords

Activity 1: Unsafe Interactions

Step 1.1	Step 1.2						
Successor Dangers	**Manifestations**						
	Pred. Link	Injection	Delay	Manipulation	Drop		
SC.DontODPatient	AppLogicCommands -> PCA Pump	PCAPump.InjectedMessage	PCAPump.LateMessage	PCAPump.ManipulatedMessage	PCAPump.DroppedMessage		
SC.DontUDPatient							

Process Variable	**Process Values**							**Unit**
Ticket Duration	1	2	3	...	598	599	600	Seconds

Step 1.3

Externally Caused Dangers						**Proposed Mitigations**	
Successor Danger	Name	Process Var. Name	Process Var. Value	Interpretation	Co-occurring Dangers	Run-time Detection	Run-time Handling
SC.DontODPatient	PCAPump.ManipulatedMessage	Ticket Duration	Higher than safe	A forged ticket arrives at the expected time but with a forged and higher-than-safe value	None	Concurrent: Hashed and signed tickets	Rollforward: Pump switches into permanent KVO (and notifies the clinician?)
SC.DontODPatient	PCAPump.InjectedMessage	Ticket Duration	Any	The PCA Pump gets a ticket "out of the blue" when it is already running	None	Concurrent: Timeouts	
				The PCA Pump gets a ticket "out of the blue" when it would be unsafe to run		Concurrent: Negative Ticket Values	
SC.DontODPatient	PCAPump.LateMessage	Ticket Duration	Any	The PCA pump gets a ticket late, so it's valid past the time window it should be	None	Concurrent: Timestamped tickets	Rollforward: Pump switches into KVO
SC.DontUDPatient	PCAPump.DroppedMessage	Ticket Duration	None	The PCA pump never gets a ticket, so it doesn't know if it's safe to run	None	None	Pump notifies the clinician

STPA and STPA-Sec based Guidewords

Activity 1: Unsafe Interactions

Step 1.1	Step 1.2						
	Manifestations						
Successor Dangers	**Pred. Link**	Not Providing	Providing	Early	Late	Stopped too Soon	Applied too Long
SC.DontODPatient	AppLogicCommands -> PCA Pump	PCAPump.NoMessage	PCAPump.InappropriateMessage	PCAPump.EarlyMessage	PCAPump.LateMessage	N / A	N / A
SC.DontUDPatient							

Process Variable	**Process Values**							**Unit**
Ticket Duration	1	2	3	...	598	599	600	Seconds

Step 1.3

Externally Caused Dangers						**Proposed Mitigations**	
Successor Danger	Name	Process Var. Name	Process Var. Value	Interpretation	Co-occurring Dangers	Run-time Detection	Run-time Handling
SC.DontUDPatient	PCAPump.NoMessage	Ticket Duration	None	No ticket is supplied	None	None	Pump notifies the clinician
SC.DontODPatient	PCAPump.InappropriateMessage	Ticket Duration	Higher than safe	A forged ticket arrives at the expected time but with a forged and higher-than-safe value	None	Concurrent: Hashed and signed tickets	Rollforward: Pump switches into permanent KVO (and notifies the clinician?)
			Any	The PCA Pump gets a ticket "out of the blue" when it is already running	None	Concurrent: Timeouts	
				The PCA Pump gets a ticket "out of the blue" when it would be unsafe to run		Concurrent: Tickets with negative values	
SC.DontODPatient	PCAPump.EarlyMessage	Ticket Duration	Any	The PCA Pump gets a ticket "too soon" -- before it has finished handling the previous ticket	None	Concurrent: Timeouts	Rollforward: Pump switches into permanent KVO (and notifies the clinician?)
SC.DontODPatient	PCAPump.LateMessage	Ticket Duration	Any	The PCA pump gets a ticket late, so it's valid past the time window it should be	None	Concurrent: Timestamped tickets	Rollforward: Pump switches into KVO

Activity 2 Using Dolev-Yao's Attacker Model

Activity 2: Internal Faults

Step 2.1

Faults Not Considered

(Section removed since the security models don't constrain the behavior of the environment beyond security concerns)

(The only model with an explicit adversary model is Dolev-Yao, so this page is derived from that, but note that other explicit models are possible, refer to the full submission for references to domain-specific attacker models)

Step 2.2

Internally Caused Dangers

Successor Danger	Guideword	Interpretation	Co-occurring Dangers	Design-time Detection	Run-time Detection	Run-time Error Handling	Run-time Fault Handling
Any	Compromised Software	An adversary is able to gain access to the software while it's being developed	Any	None	TPM-Like Device + Cryptographic Chain-of-Trust	Rollback: Pump switches into permanent KVO (and notifies the clinician?)	None
Any	Compromised Hardware	An adversary is able to gain access to the hardware while it's being developed			"Exotic"		
Any	Adversary Accesses Hardware	An adversary is able to gain access to the software while it's in use		N / A	"Exotic", Physical Security		
Any	Adversary Accesses Software	An adversary is able to gain access to the hardware while it's in use			Access Control, Physical Security		

Comparison Table

	Negative Ticket Values	Timeouts	Timestamps	Hashed and signed messages	Encrypted Tickets	Alarms
Avižienis	?	Y	Y	N	N	Y
Dolev-Yao	?	?	Y	Y	Y	Y
STPA-SafeSec	?	?	Y	Y	N	Y
STPA/STPA-Sec	?	Y	Y	?	N	Y

This comparison (cf. Table 4 in the paper) was created by going through suggested design improvements and marking the technique with a Y if the improvement was directly suggested by a guidephrase (ie, it didn't require splitting the guideword into multiple problems / compensatory techniques); with a ? if the improvement was suggested indirectly (ie, it required an "a-ha!" moment, or prior analyst experience, to suggest multiple ways the same effect could occur); or with a N if the improvement wasn't suggested by a technique.