Some Research Topics

Chris Hankin

Imperial College London

Outline

- Context – RITICS and KIOS
- Some Contributions
  - Monitoring
  - Measuring
  - Diversifying
  - Defending
- The Broader Network

Key Questions / Challenges for RITICS Phase 1 (2014-2018)

Do we understand the harm threats pose to our ICS systems and business?

Can we confidently articulate these threats as business risk?

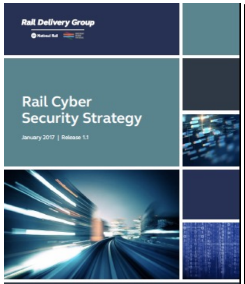
What could be novel effective and efficient interventions?

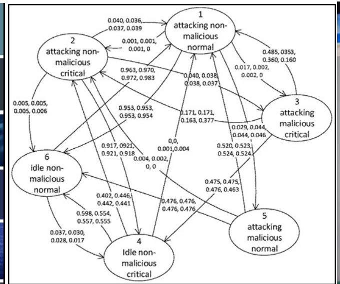
RITICS Phase 1 Projects

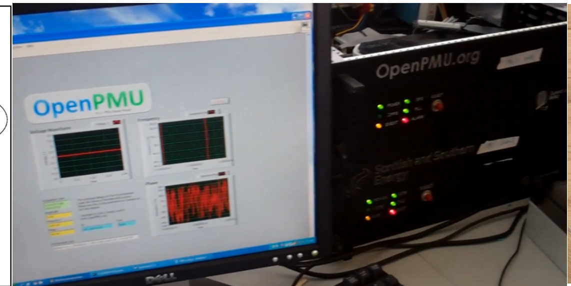
- RITICS (Hankin, Chana, Imperial College London)
- MUMBA (Rashid, Lancaster/Bristol)
- CEDRICS (Bloomfield, Popov, City)
- SCEPTICS (Easton, Chothia, Birmingham)
- CAPRICA (Sezer, Queen's University Belfast)

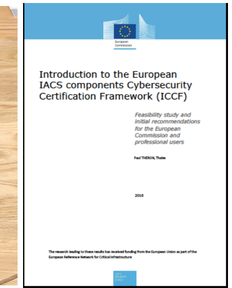
Impact of Phase 1

- Creation of a new research community
- Contribution to new Cyber Security Strategy for UK railways.
- Tools for building models of complex cyber physical systems.
- Testbeds.
- A serious game for studying security decisions.
- Secure implementation of gateway module compatible with IEC and IEEE standards.
- Contribution to European work on certification of ICS components.

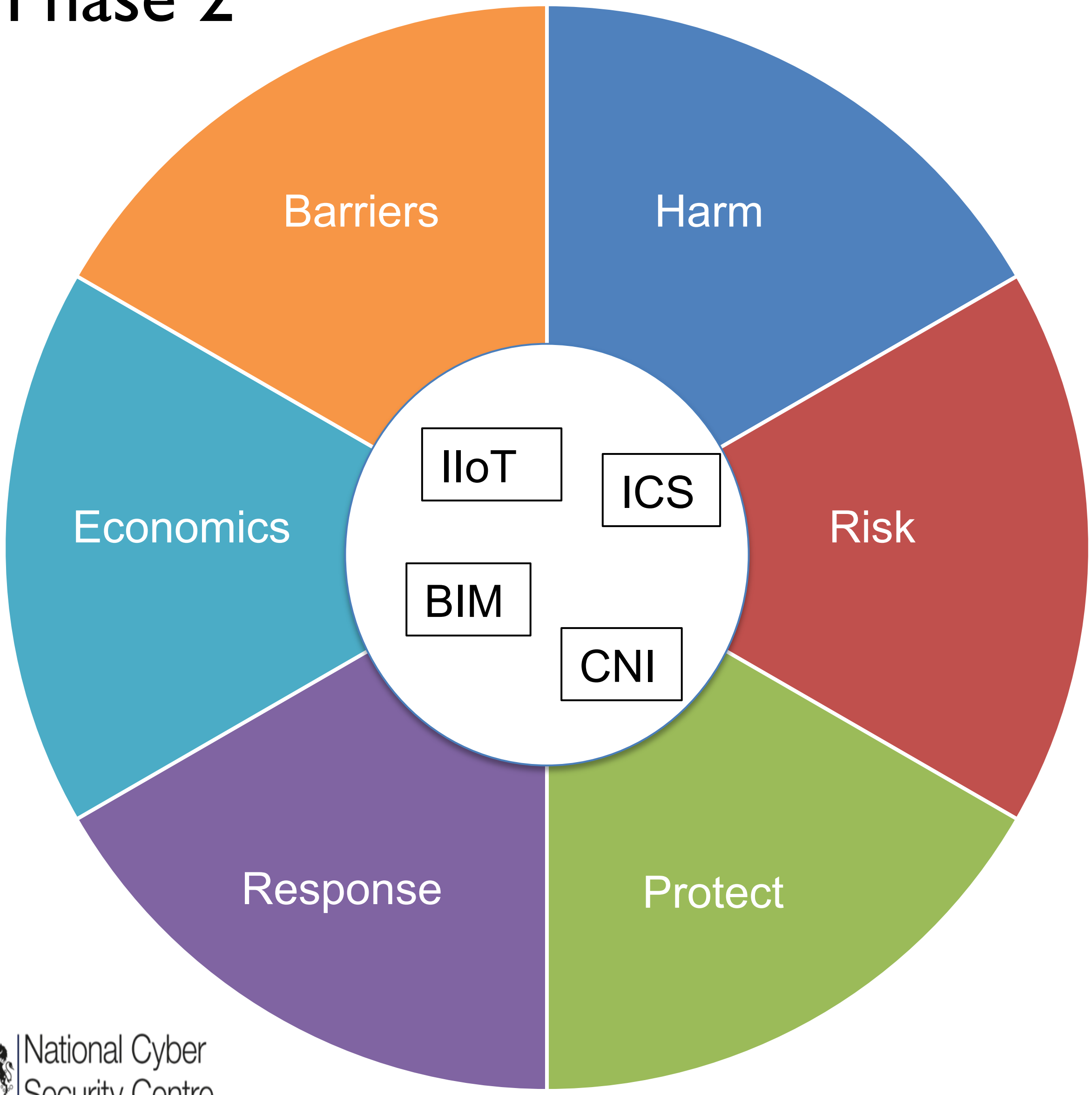








RITICS Phase 2



EPSRC
Engineering and Physical Sciences Research Council

National Cyber Security Centre

The RITICS Programme

- NIS Directive – baseline, barriers, impact
- Safety and Security
- Autonomous Systems
- Incident Response and Forensics
- Cyber Controls
- Interconnected Systems
- Supply Chain

NIS Directive

How many shades of NIS: Understanding Organisational Cybersecurity and Sectoral Differences - Bristol

Effective Solutions for the NIS Directive: Supply Chain Requirements for Third Party Devices - Birmingham

Establishing a Scientific Baseline for Measuring the Impact of the NIS Directive on Supply Chain Resilience - Glasgow

Second Call

AIR4ICS: Agile Incident Response For Industrial Control Systems – DMU

Cloud-enabled Operation, Security Monitoring, and Forensics (COSMIC) – QUB

Developing Pedagogy to Optimise Forensic Training in Safety-Related Industrial Control Systems (ICS) – Glasgow

Interconnected safe and secure systems (IS^3) - City

Third Call

Diversity-by-design: Quantifying vulnerability similarity of Interconnected Networks - Cardiff

Emergence of cybersecurity capability across interdependent critical infrastructure, from the nexus of business, engineering and public policy interests – Glasgow/Belfast

NDN for Secure Industrial IoT Networking - Belfast

Three contributions:

- Measuring Cyber-physical security
- Software Diversity
- AI and Intrusion Detection

Security Metrics

With Martín Barrère, Demetrios Eliades,
Nicolas Nicolaou and Thomas Parisini

Agenda

1. Introduction
2. Base security metric (weighted AND/OR graphs)
3. Extended security metric (AND/OR hypergraphs)
4. Analytical evaluation
5. Case study on water transport networks
6. Conclusion and future work

Introduction

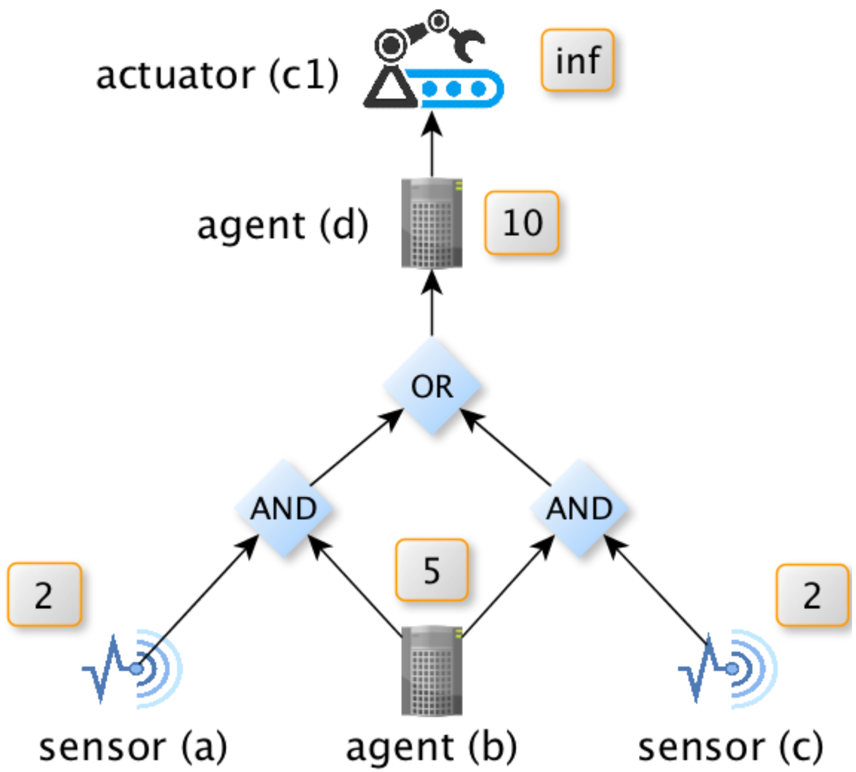
- Goal: **security metric** for ICS networks

- AND/OR graphs to model **complex interdependencies** between cyber-physical components

- Identify **critical ICS nodes**, with **minimal compromise cost**, that could disrupt the operation of the system
  - NP-complete problem
  - Multiple overlapping security measures

- Measure security levels, **compare different ICS settings**

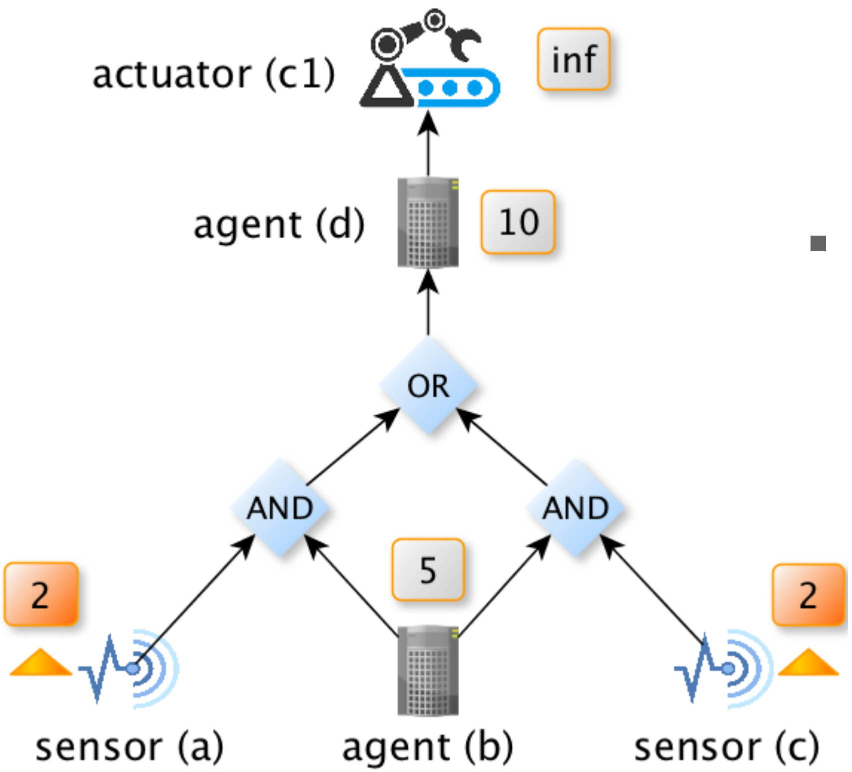
ICS network model (simple example)



- AND/OR graph with sensors, software agents and actuators
- Adversarial model: an attacker can compromise any network node $n \in V_{AT}$ at a certain cost $\varphi(n)$ with $\varphi : V_{AT} \to \mathbb{R}_{\geq 0}$
- Compromised node: component unable to operate properly

Least-effort attack strategy (critical nodes)

- Objective: **set of nodes**, with minimal **cost (effort)** for an attacker, such that if **compromised**, the system would enter into a non-operational state



- Solution:
  - Critical nodes: a, c
  - Total cost: 4

- Problem: identifying critical nodes in AND/OR graphs => NP-complete
(Desmedt et al. (2004); Jakimoski et al. (2004); Souza et al. (2013))

MAX-SAT resolution approach

MAX-SAT problem: find a truth assignment that maximises the weight of the satisfied clauses (or minimise the weight of falsified clauses)

1. AND/OR logical transformation:

$$f_G(c1) = c1 \wedge (d \wedge ((a \wedge b) \vee (b \wedge c)))$$

2. Attacker's objective:

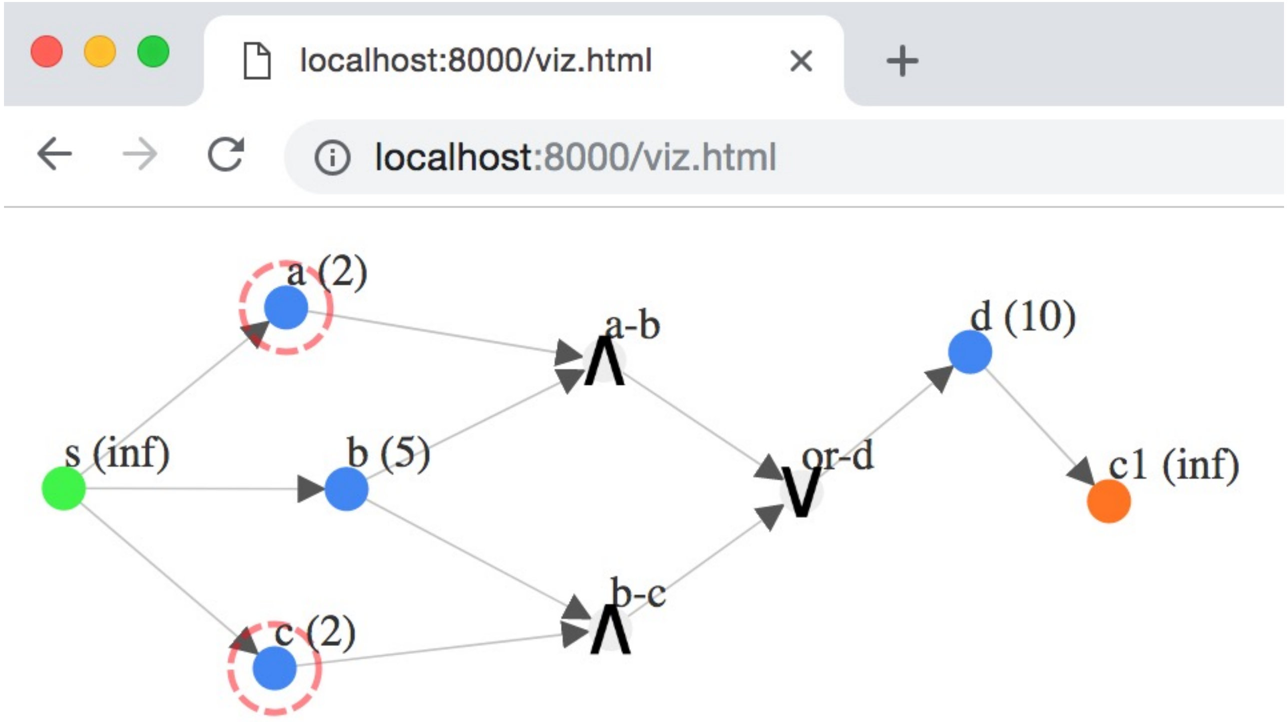
$$\neg f_G(c1) = \neg(c1 \wedge (d \wedge ((a \wedge b) \vee (b \wedge c))))$$

3. MAX-SAT problem specification:
   - Falsification penalty scores

a	b	c	d	$c1$
$\varphi(a) = 2$	$\varphi(b) = 5$	$\varphi(c) = 2$	$\varphi(d) = 10$	$\varphi(c1) = inf$

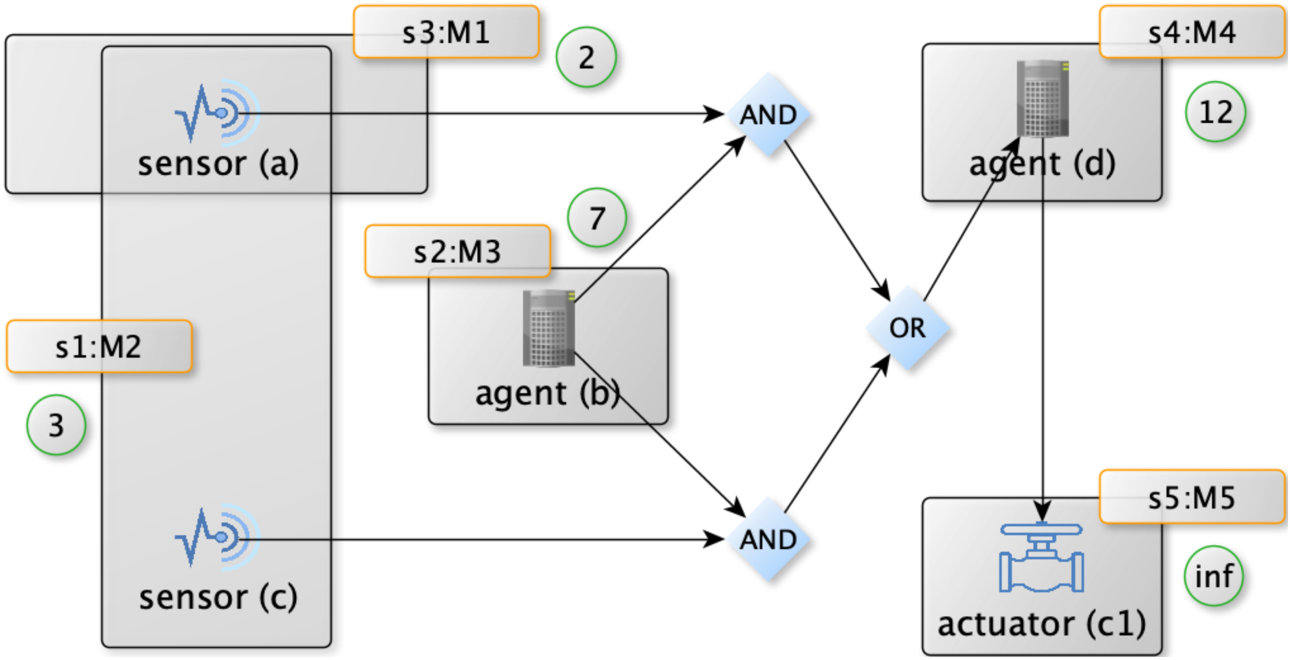
   - MAX-SAT solution: minimises the penalty induced by falsified weighted variables

Visualisation system (META4ICS)



- **META4ICS:** Metric Analyser for Industrial Control Systems
  Available at: https://github.com/mbarrere/meta4ics

Multiple overlapping security measures



- Set of security measure instances: $S = \{s1, s2, \ldots\}$
- Cost function (attacker's effort): $\psi : S \rightarrow \mathbb{R}_{\geq 0}$

Measure instance	$s1$	$s2$	$s3$	$s4$	$s5$
Measure type	$M2$	$M3$	$M1$	$M4$	$M5$
Attacker's cost $\psi(s_j)$	3	7	2	12	inf
Protection range	$\{a, c\}$	$\{b\}$	$\{a\}$	$\{d\}$	$\{c1\}$

Extended security metric (formulation)

$$\mu(G,t) = \underset{X \subseteq V_{AT}}{\text{argmin}} \Big(\sum_{x_i \in X} \varphi(x_i) + \sum_{s_j \in S(X)} \psi(s_j) \Big)$$

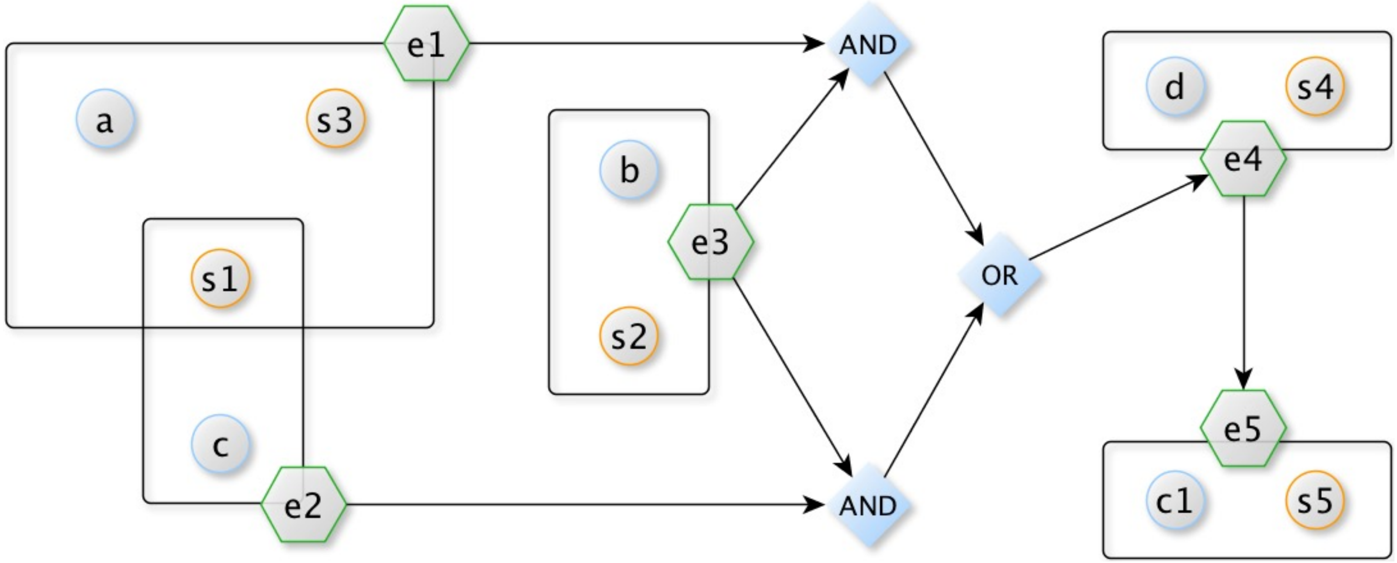
s.t.

$$wcc(\sigma(G,X)) \geq 2 \vee X = \{t\}$$

- Inputs: AND/OR graph, target node
- Solution node set: $X \subseteq V_{AT}$
- Functions:
  - $S(X)$ => set of measure instances $\{s_i, \ldots, s_j\}$ protecting X
  - $\sigma(G,X)$ => removes nodes in X from G
  - $wcc(G)$ => weakly connected components

AND/OR hypergraph-based approach

- Hypergraphs: generalisation of standard graphs where graph edges (hyperedges) can connect any number of vertices



e_1	e_2	e_3	e_4	e_5
$\{a, s1, s3\}$	$\{c, s1\}$	$\{b, s2\}$	$\{d, s4\}$	$\{c1, s5\}$

$$f_H(e5) = e5 \wedge e4 \wedge ((e1 \wedge e3) \vee (e3 \wedge e2))$$

AND/OR hypergraph resolution

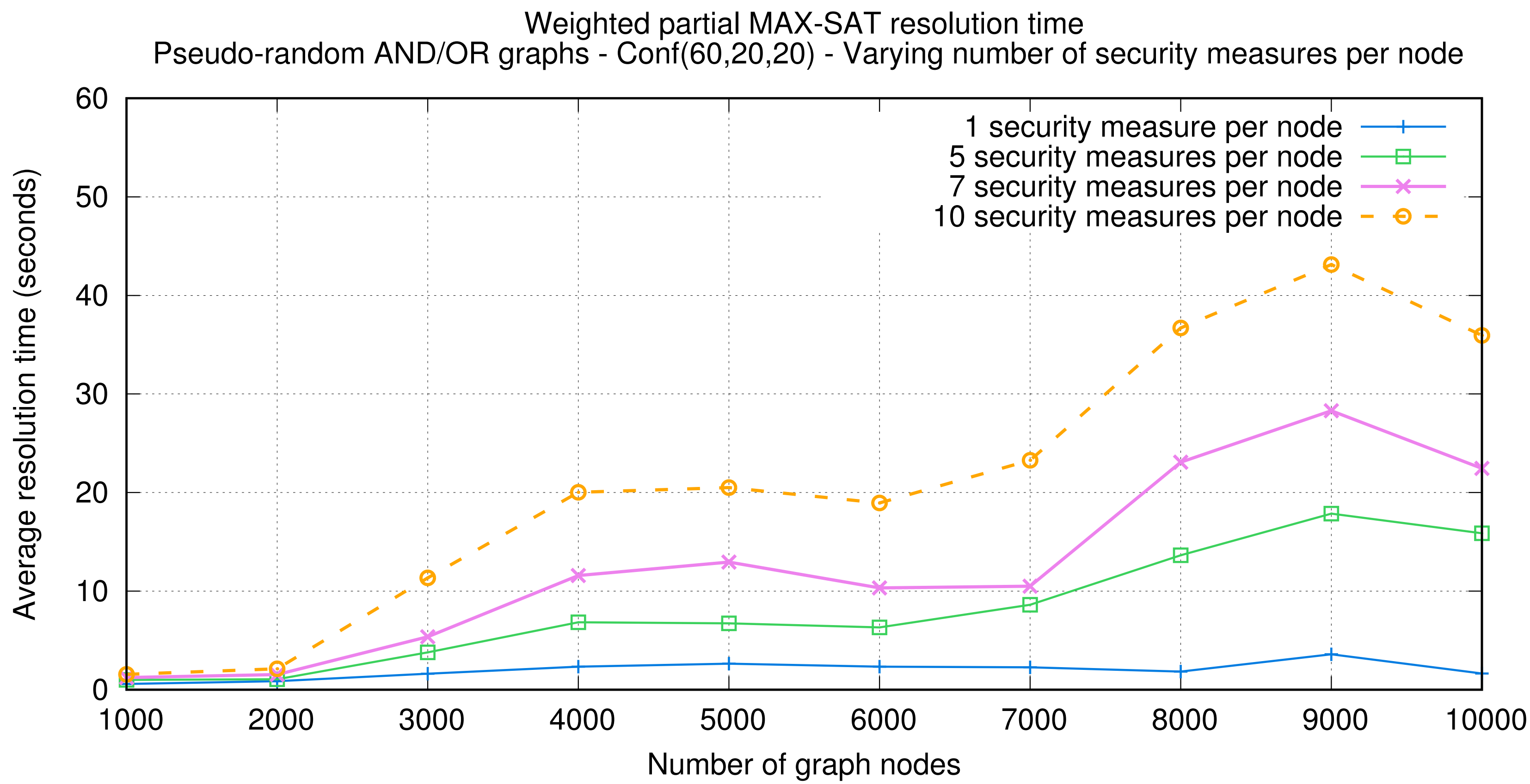
$$h_G(c1) = (c1 \vee s5) \wedge (d \vee s4) \wedge \\ (((a \vee s1 \vee s3) \wedge (b \vee s2)) \vee ((b \vee s2) \wedge (c \vee s1)))$$

- Attacker's compromise costs (used as MAX-SAT penalty values):
  - Atomic nodes: $\varphi(n) = 1, \forall n \in V_{AT}$
  - Measure instances:

Measure instance	$s1$	$s2$	$s3$	$s4$	$s5$
Attacker's cost $\psi(s_i)$	3	7	2	12	inf

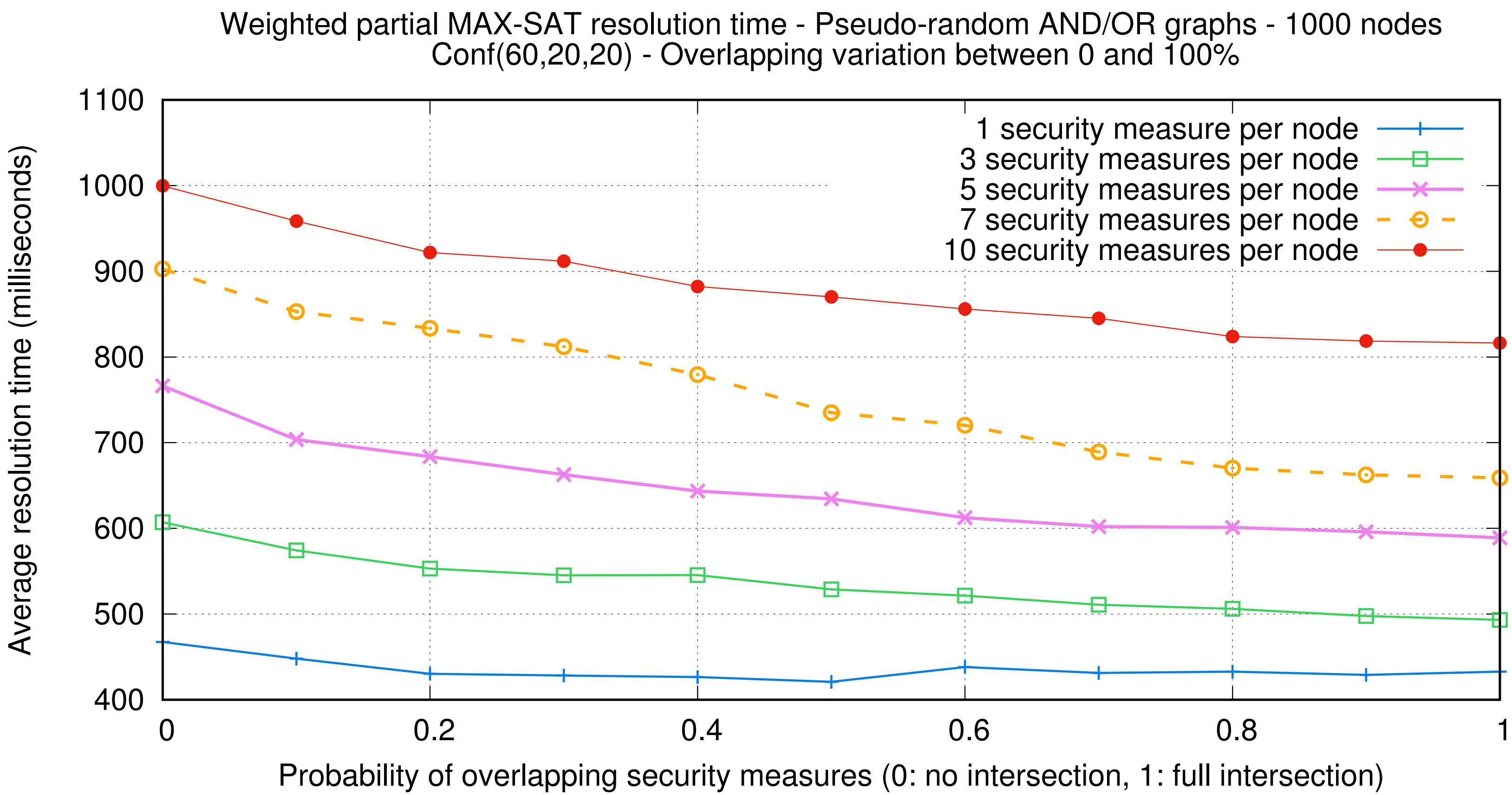
- First attempt: falsify $(b \vee s2)$
  - Cost: 1 + 7 = 8
- Second attempt: falsify $(a \vee s1 \vee s3)$ and $(c \vee s1)$
  - Cost for set $\{a, s1, s3, c\}$: 1 + 3 + 2 + 1 = 7 (MIN)

Disjoint security measures



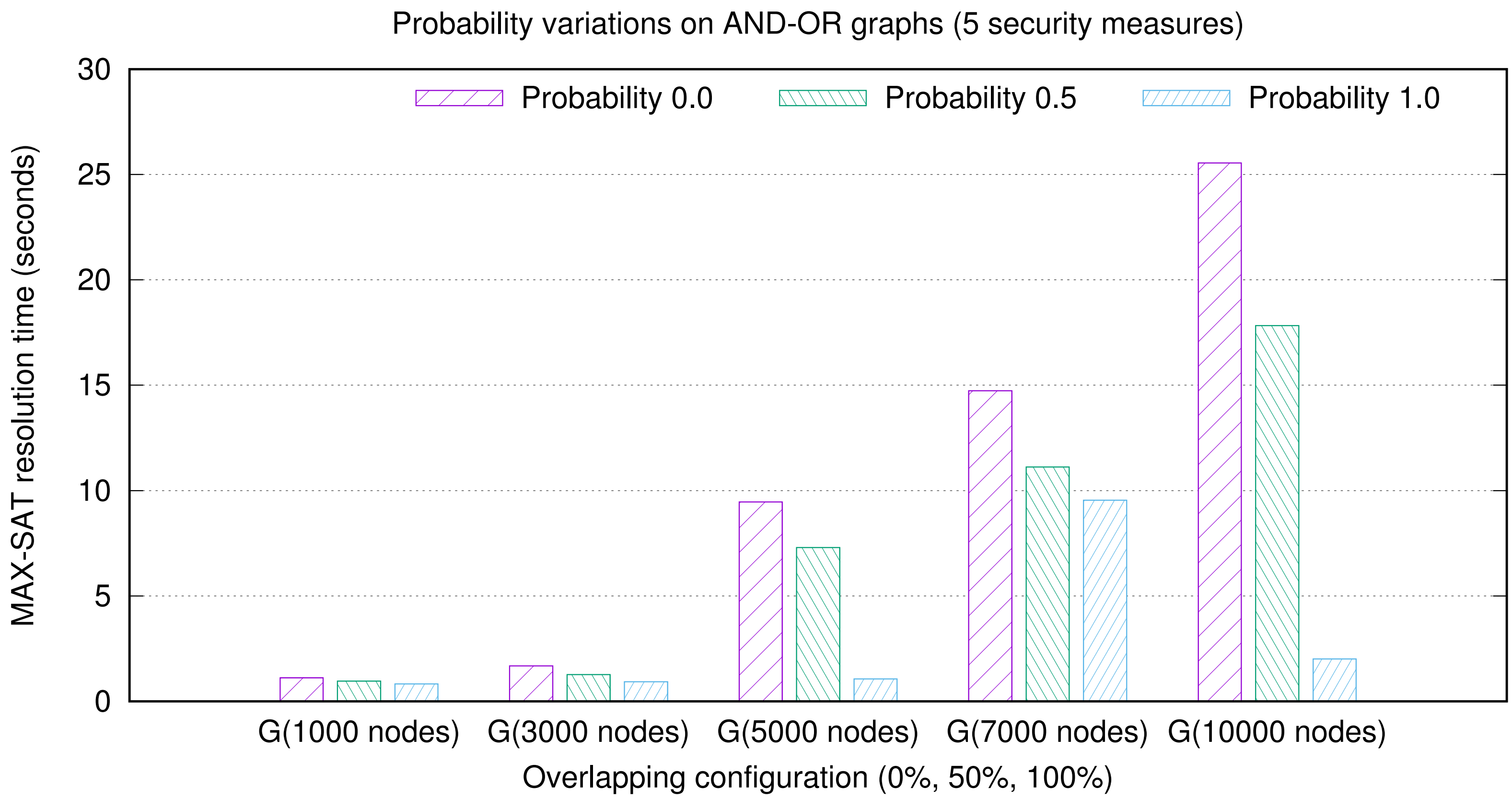
- Scalability evaluation while increasing graph size
- Fast resolution for base problem (one measure per node)

Overlapping security measures (1)



- Variation analysis of overlapping measures
- Graph with 1000 nodes

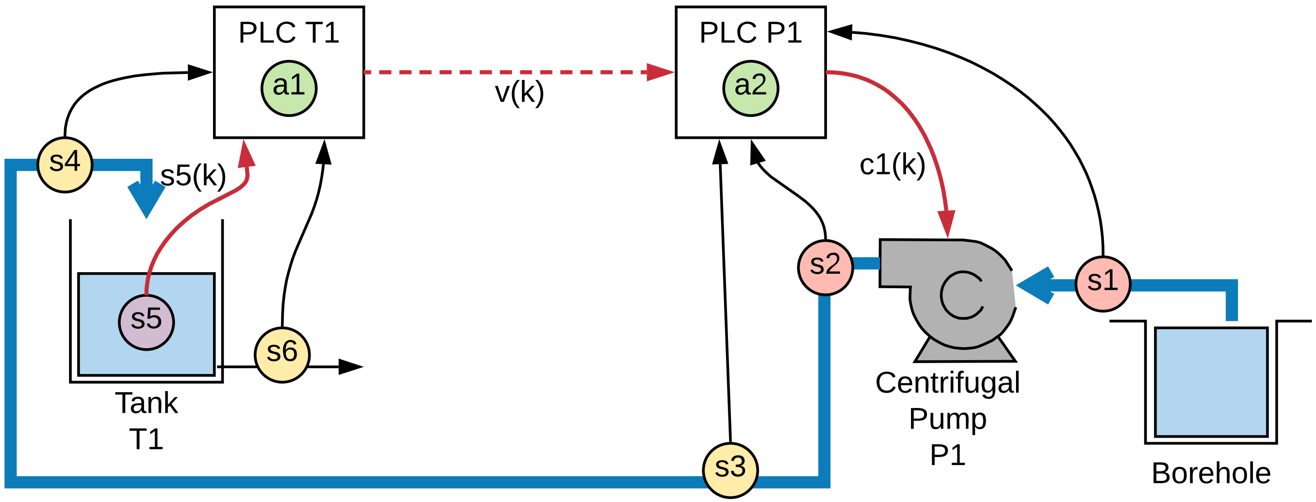
Overlapping security measures (2)



- Overlapping analysis on graphs of different sizes
- Same pattern (more overlapping => faster resolution)

Case study

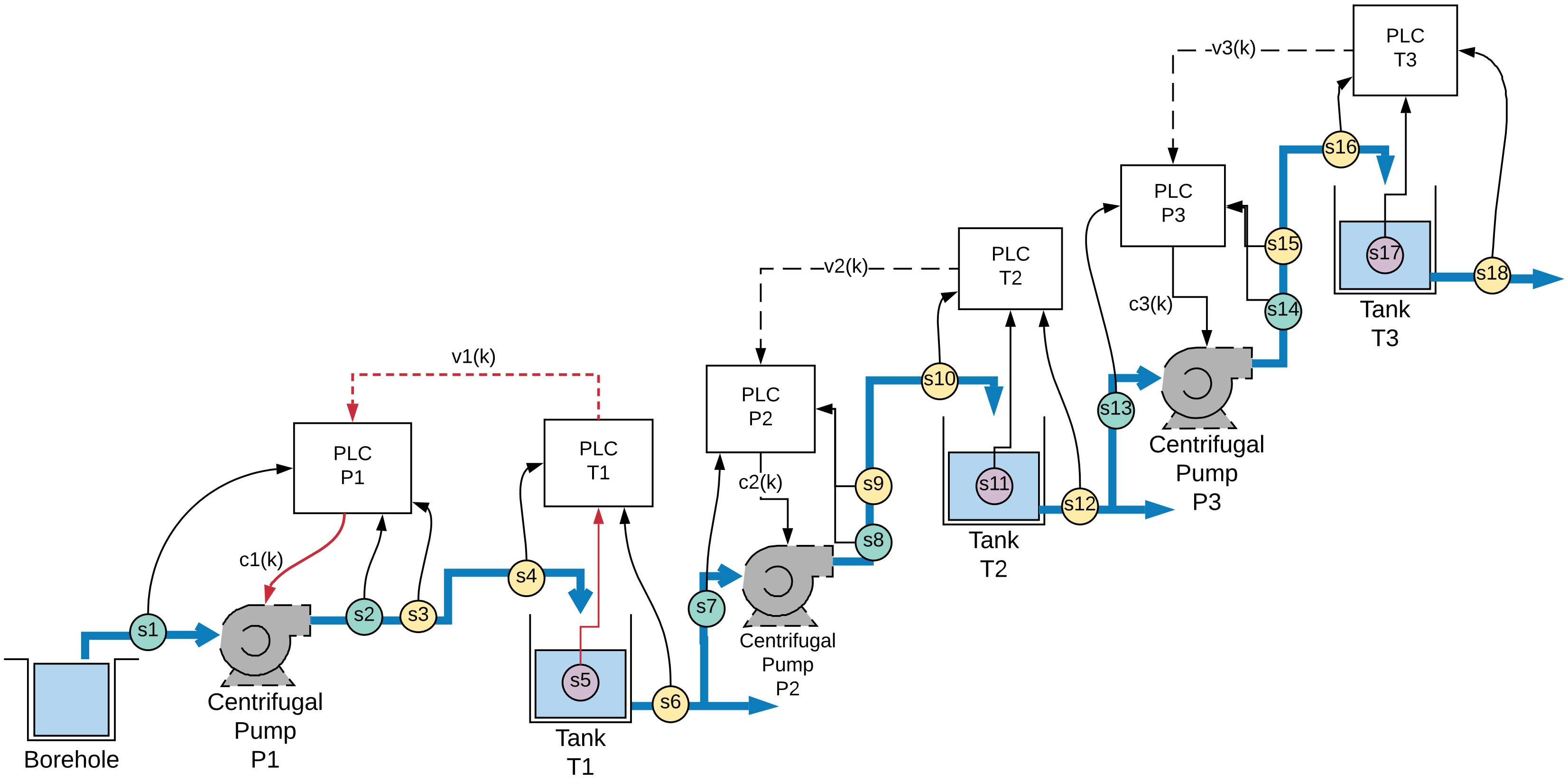
- Focus: water transport networks (base sub-system)



- Pressure sensors s1 (before) and s2 (after) the pump P1
- Flow sensor s3 (pump outflow)
- Water level sensor s5 at tank T1 with flow sensors (s4 and s6)
- Two PLCs: agent a1 (tank T1) and agent a2 (pump P1)

Case study

- Base sub-system repeated in larger infrastructures



Data collection and preparation

- Measures acquired from utility operators and public sources
- Attacker's cost: three-point rating scale

Factor / Rate	**1**	**2**	**3**
Skills ($f1$)	no special skills/knowledge	advanced skills/knowledge	expert skills/knowledge
Tools ($f2$)	off-the-shelf tools	non-conventional tools required	specialized tools
Time ($f3$)	$\leq$ 10 min	10-30 min	$\geq$ 30 min

- Cost function

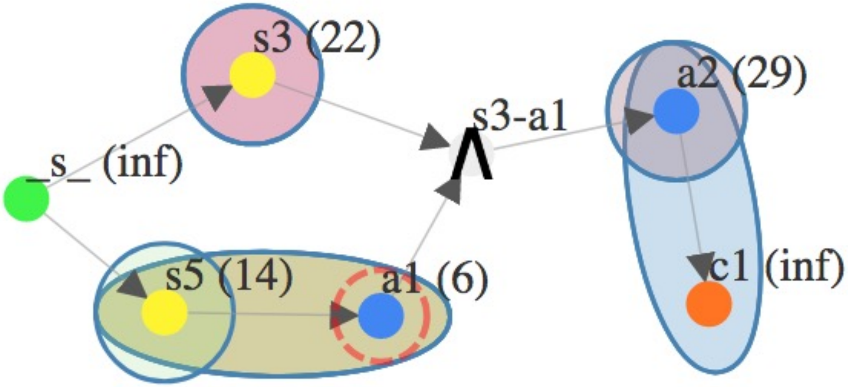
$$\psi(m) = f1 \times f2 \times f3$$

Measure	**Skills**	**Tools**	**Time**	**Attack cost**	**Description**
F1	1	1	1	**1**	Fenced area (wire)
F2	1	2	1	**2**	Fenced area (locked underground facility)
B1	1	1	2	**2**	Building + regular lock
B2	2	2	2	**8**	Building + secure lock
A1	2	3	2	**12**	Door alarm
A2	3	2	3	**18**	Alarm on telemetry box
A3	1	1	3	**3**	Patrol unit
P1	1	2	1	**2**	Locked box
P2	2	2	2	**8**	Cable protection

Base subsystem (no redundancy)

Components	Security measures	Total cost
$s3$	{F2-1, P1-2, A2-2}	22
$s5$	{F1-2, B1-1, A3-1, P2-2}	14
$a1$	{F1-2, B1-1, A3-1}	6
$a2$	{F1-1, B2-1, P1-1, A2-1}	29
$c1$	{F1-1, B2-1}	9 + inf (special case)

- Solution:
  - Critical nodes: agent a1
  - Security measures:
    F1-2, B1-1, A3-1
  - Total cost: 6



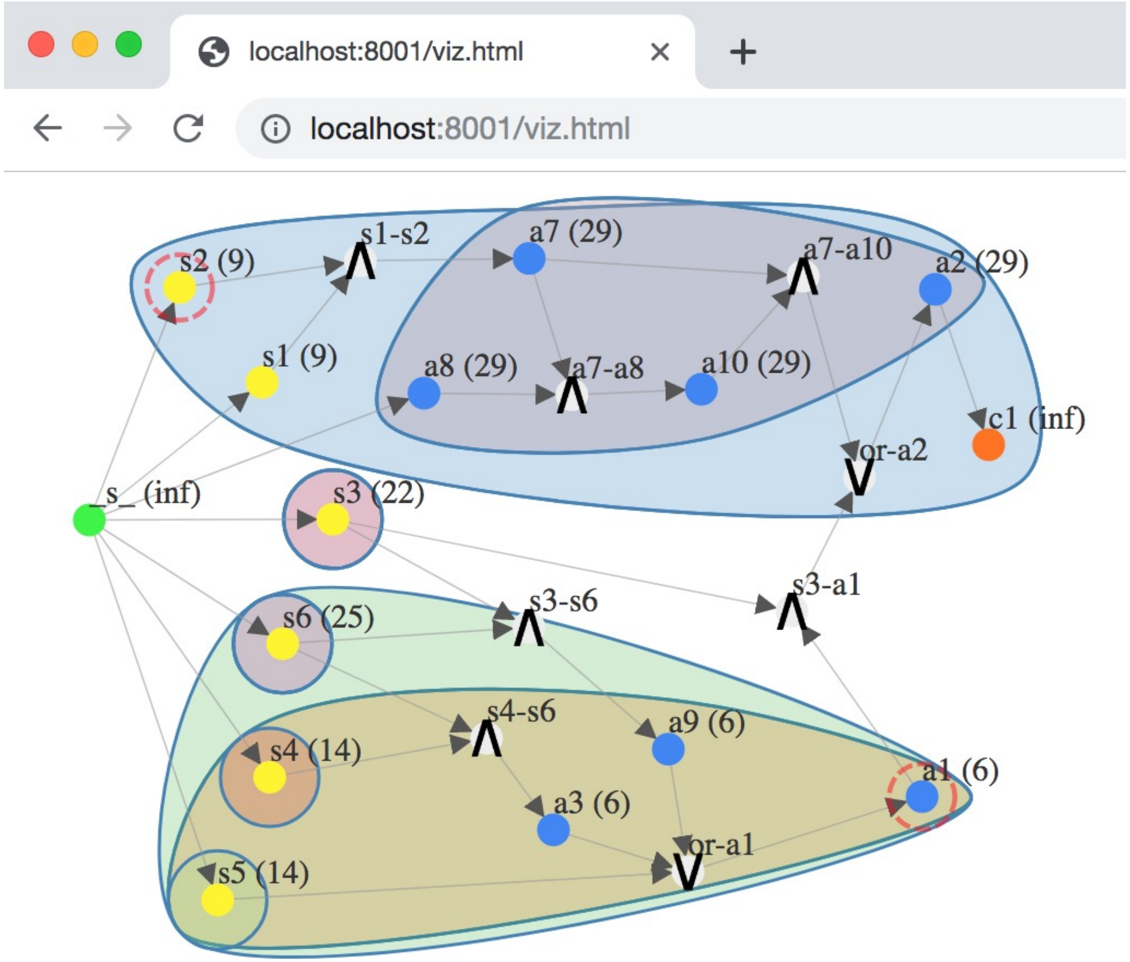
META4ICS display

Extended subsystem (with redundancy)

Components	Security measures	Total cost
$a2, a7, a8, a10$	{F1-1, B2-1, P1-1, A2-1}	29
$a1, a3, a9$	{F1-2, B1-1, A3-1}	6
$s1, s2$	{F1-1, B2-1}	9
$c1$	{F1-1, B2-1}	9 + inf (special case)
$s3$	{F2-1, P1-2, A2-2}	22
$s4$	{F1-2, B1-1, A3-1, P2-1}	14
$s5$	{F1-2, B1-1, A3-1, P2-2}	14
$s6$	{F2-2, P1-3, A2-3, A3-1}	25

Measure instance	Measure type	Attacker cost	Protection range
F1-1	F1	1	$\{a2, a7, a8, a10, c1, s1, s2\}$
F1-2	F1	1	$\{a1, a3, a9, s4, s5\}$
F2-1	F2	2	$\{s3\}$
F2-2	F2	2	$\{s6\}$
B1-1	B1	2	$\{a1, a3, a9, s4, s5\}$
B2-1	B2	8	$\{a2, a7, a8, a10, c1, s1, s2\}$
A2-1	A2	18	$\{a2, a7, a8, a10\}$
A2-2	A2	18	$\{s3\}$
A2-3	A2	18	$\{s6\}$
A3-1	A3	3	$\{a1, a3, a9, s4, s5, s6\}$
P1-1	P1	2	$\{a2, a7, a8, a10\}$
P1-2	P1	2	$\{s3\}$
P1-3	P1	2	$\{s6\}$
P2-1	P2	8	$\{s4\}$
P2-2	P2	8	$\{s5\}$

Extended scenario (META4ICS display)



- Solution: nodes a1 and s2, instances F1-2, B1-1, A3-1, F1-1, B2-1
- Total cost: 15

Conclusion

- Identification of security-critical nodes in ICS environments
- Security metric as least-effort attack strategy
- AND/OR graph-based models
  - Base problem (weighted AND/OR graphs)
  - Multiple overlapping security measures (AND/OR hypergraphs)
- Combination of AND/OR graphs with MAX-SAT optimisation techniques
- Experimental results indicate very good scalability
- Practical analysis of a realistic water transport network

Future work

- Evaluation on other ICS environments
  - Smart grid, power plants
- Integrate attack graphs at the cyber level
- Consider budget constraints
- Automated generation of AND/OR graph models for ICS

Software Diversity

With Tingting Li and Feng Cheng

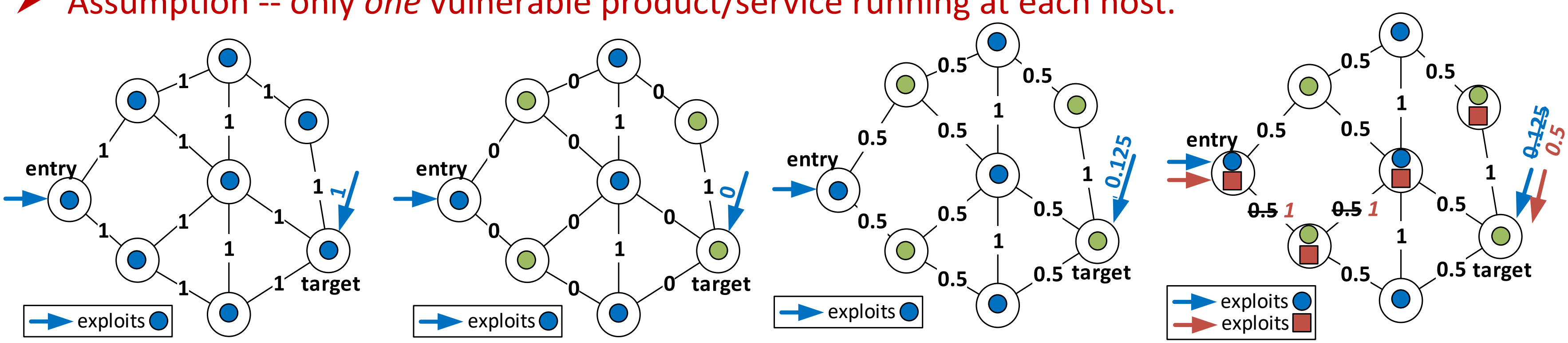
Motivation

Software Diversity -- an Effective Defense Strategy

- Software mono-culture promotes and accelerates the spread of malware.
- Diversification can mitigate the infection of malware between similar products and reduce the likelihood of repeating application of single exploits.
- More essential when combating zero-day exploits.

Existing work on Diversity-inspired defence

- From the software development: *n-version programming, code randomization, etc..*
- From the perspective of security management:
  - Optimal product assignment by distributed colouring algorithm [CCS'04]
  - Diversifying routing nodes in a network [TDSC'15]
  - Security metrics to evaluate network diversify and resilience. [ESORICS'10, TIFS'16]
- Assumption -- Products share *no* vulnerabilities between each other.
- Assumption -- only *one* vulnerable product/service running at each host.



(a) Mono-culture single-label hosts

(b) diversifying single-label hosts

(c) diversifying single-label hosts with *similarity*

(d) diversifying *multi-label* hosts with *similarity*

Main Contributions

Objectives

- An accurate model to determine the infection of potential exploits across a network.
- An efficient way to find product assignments to minimise the prevalence of 0day exploits.

Main Contributions

- Modelled various services/products at each host and exposed attack vectors.
- Proposed the metric *vulnerability similarity of products*; Statistical study of *CVE/NVD*.
- Formally model the multi-labelling network by a discrete *Markov Random Field (MRF);* Optimised by the *sequential tree-reweighted message passing (TRW-S)* algorithm.
- A case study inspired by *Stuxnet Propagation* to
  - Find the optimal assignment against the collaboration of multiple 0day exploits
  - Evaluate the optimal result in a *NetLogo simulation* in terms of MTTC.
- Scalability analysis of our optimisation method against
  - Large-scale networks with *up to 10,000 hosts*.
  - High-density networks with *up to 50 degrees (# edges) per host*.
  - High-complexity networks with *up to 30 products/services per host*.
  - Most heavy cases converged from a couple of seconds to ~3 minutes.

Similarity Metrics

Similarity of Products Vulnerability based on CVE/NVD

- Firstly define the similarity between a pair of products;
  - capture statistically how similar the vulnerabilities found on two products are.
  - likelihood of being compromised by the same exploit.

Definition 1 *Let x_i, x_j be a pair of products, $\mathbf{V}_{x_1}$ and $\mathbf{V}_{x_j}$ are vulnerabilities of x_i and x_j respectively. The vulnerability similarity between x_i and x_j can be obtained by the* Jaccard similarity coefficient*:*

$$sim(x_i, x_j) = \frac{|\mathbf{V}_{x_i} \cap \mathbf{V}_{x_j}|}{|\mathbf{V}_{x_i} \cup \mathbf{V}_{x_j}|}$$

- Common Vulnerability Enumerations (CVE) and National Vulnerability Database (NVD)

CVE-ID	CVE-2016-7153
Overview	The HTTP2 protocol does not consider the role of the TCP congestion window in providing information about content length, which makes it easier for remote attackers to obtain cleartext data by leveraging a web-browser configuration in which third-party cookies are sent, aka a "HEIST" attack.
Release Date	September 6th, 2016
CVSS Severity	Base Score: *5.0 MEDIUM*; Vector: *AV:N/AC:L/Au:N/C:P/I:N/A:N* ; Impact Subscore: *2.9*; Exploitability Subscore: *10.0*
CVSS V2 Metrics	Access Vector: *Network exploitable*; Access Complexity: *Low*; Authentication: *Not required to exploit*;
Vulnerable software &Versions	`cpe:/a:microsoft:edge:-` `cpe:/a:microsoft:internet_explorer:-` `cpe:/a:google:chrome:-` `cpe:/a:apple:safari` `cpe:/a:mozilla:firefox` `cpe:/a:opera:opera_browser:-`

Common Platform Enumerations(CPE):

- well-formed naming scheme for IT systems, platforms and packages.

`cpe:/PART:VENDOR:PRODUCT:VERSION`

Similarity Metrics

Similarity of Products Vulnerability based on CVE/NVD

- 84,229 vulnerabilities in NVD; CPE serves to sort vulnerabilities according to affected products.
- Compare most vulnerable OS products and Web Browsers [CVE Details] from 1999 to 2016.

	WinXP2	Win7	Win 8.1	Win10	Ubt14.04	Deb8.0	Mac10.5	Suse13.2	Fedora
WinXP2	1.00 (479)								
Win7	0.278 (328)	1.00 (1028)							
Win8.1	0.009 (10)	0.228 (298)	1.00 (572)						
Win10	0 (0)	0.124 (164)	0.697 (421)	1.00 (453)					
Ubt14.04	0 (0)	0 (0)	0 (0)	0 (0)	1.00 (612)				
Deb8.0	0 (0)	0 (0)	0 (0)	0 (0)	0.208(195)	1.00 (519)			
Mac10.5	0 (0)	0.081 (109)	0 (0)	0 (0)	0 (0)	0 (0)	1.00(424)		
Suse13.2	0 (0)	0 (0)	0 (0)	0 (0)	0.170(161)	0.112 (102)	0 (0)	1.00(492)	
Fedora	0 (0)	0 (0)	0 (0)	0 (0)	0.083(75)	0.049 (41)	0.001(1)	0.116 (89)	1.00(367)

	IE8	IE10	Edge	Chrome	Firefox	Safari	SM	Opera
IE8	1.0 (349)							
IE10	0.386 (240)	1.0 (513)						
Edge	0.014 (7)	0.121 (73)	1.0 (194)					
Chrome	0 (0)	0 (0)	0.001 (2)	1.0 (1661)				
Firefox	0 (0)	0 (0)	0.001 (2)	0.005 (15)	1.0 (1502)			
Safari	0 (0)	0 (0)	0.002 (2)	0.009 (21)	0.003 (6)	1.0 (766)		
SeaMonkey	0 (0)	0 (0)	0 (0)	0.001 (3)	0.450 (683)	0.001(1)	1.0(492)	
Opera	0 (0)	0 (0)	0.003 (1)	0.003 (6)	0.004 (7)	0.004(4)	1.00(492)	1.00(225)

Similarity Metrics

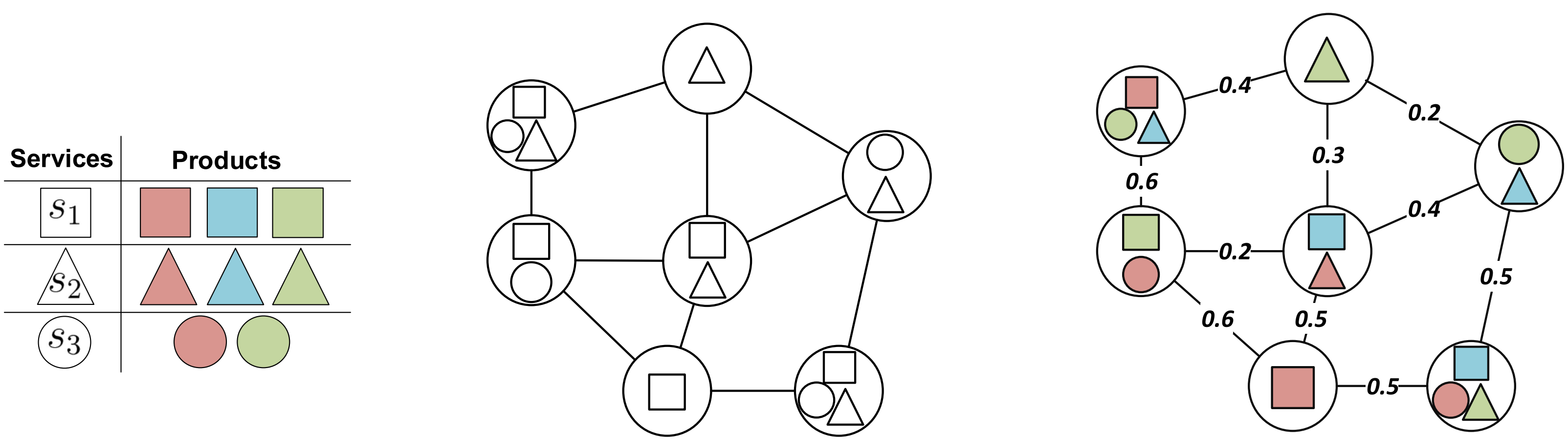
Similarity of Hosts and Infection Models

- Each host runs a set of services: $S_{h_i} = \{s_1, \ldots, s_k\},\ where\ S_{h_i} \in 2^S$
- Each service is provided by a set of products: $p(s_j) = \{p^i_{s_j}, \ldots, p^k_{s_j}\},\ where\ p^i_{s_j} \in P$
- An assignment of products for a host: $\alpha(h_i, S_{h_i}) = (\alpha'(h_i, s_1), \ldots, \alpha'(h_i, s_k)) = (p^m_{s_1}, \ldots, p^n_{s_k})$
- Estimate the infection rate by comparing the assigned products (i.e. similarity of hosts).

$$sim(\alpha(h_i, S_{h_i}), \alpha(h_j, S_{h_j})) =$$

$$\begin{cases} \max\limits_{\forall s_k \in S_{h_i} \cap S_{h_j}} \{sim(\alpha'(h_i, s_k), \alpha'(h_j, s_k))\}, & for\ sophisticated\ attackers \\ \underset{\forall s_k \in S_{h_i} \cap S_{h_j}}{random} \{sim(\alpha'(h_i, s_k), \alpha'(h_j, s_k))\}, & for\ naive\ attackers \end{cases}$$

$$r(h_i, h_j) = sim(\alpha(h_i, S_{h_i}), \alpha(h_j, S_{h_j}))$$



(a) available products

(b) a network

(c) the infection model with assigned products

Optimal Assignment of Diverse Products

Formal Model by Markov Random Field (MRF)

- Each service has various selections of products → up to $|P|$ labels.
- Each host provides multiple services → up to $|P| \times |S|$ labels.
- Sufficient flexibility and generality.
- Existence of feasible/efficient optimisation solution.
- The optimal diversification problem → find an optimal label for each service at each host

Optimisation

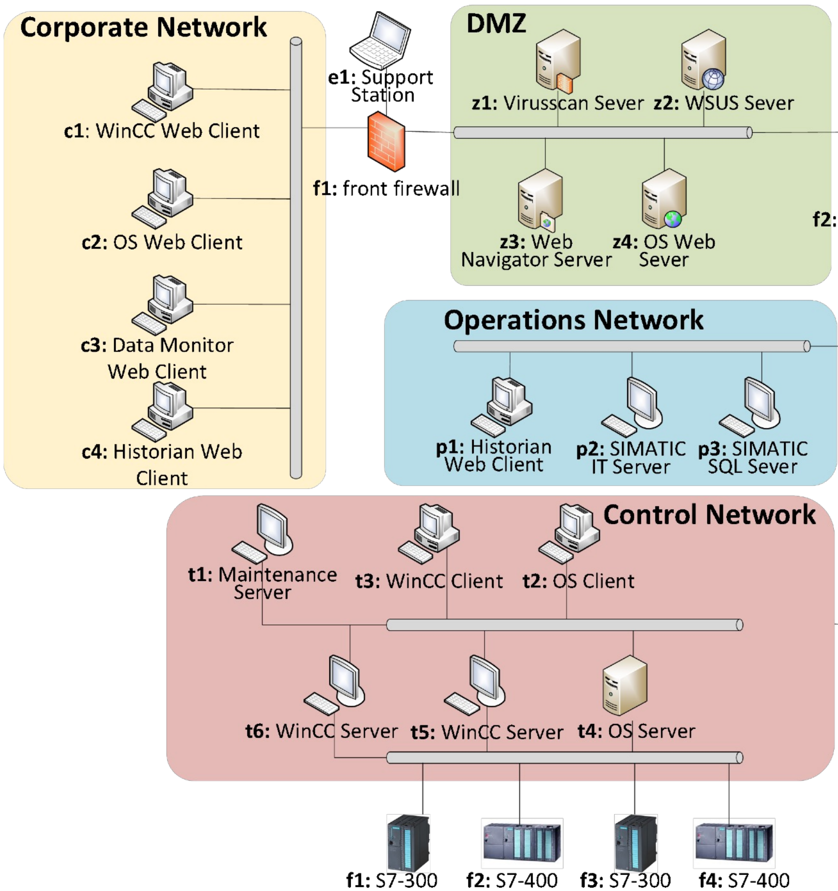
- Given an infection model $\mathbf{G} := (H, L, \alpha)$, the energy function is given as:

$$E(\mathbf{G}) = \sum_{h_i \in H, s_k \in S_i} \phi(h_i, s_k) + \sum_{(h_i, h_j) \in L} \psi(\alpha(h_i, S_{h_i}), \alpha(h_j, S_{h_j}))$$

$$\begin{aligned} \hat{\alpha} &= \operatorname*{argmin}_{\alpha} E(\mathbf{G}) \\ &= \operatorname*{argmin}_{\alpha} \sum_{h_i \in H} \sum_{s_j \in S_{h_i}} Pr_{const} + \sum_{(h_i, h_j) \in L} \sum_{s_k \in S_{h_i} \cap S_{h_j}} sim(\alpha(h_i, S_{h_i}), \alpha(h_j, S_{h_j})) \end{aligned}$$

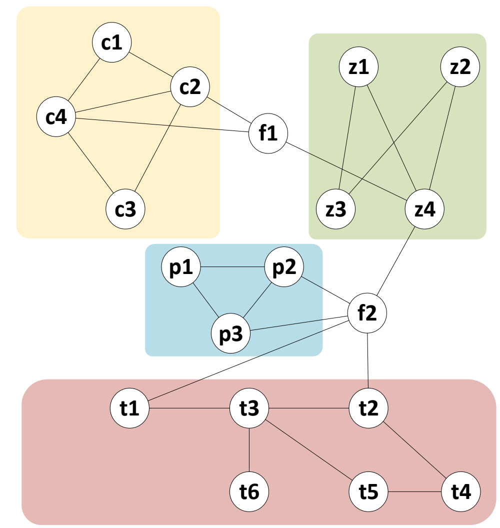
- Solved by *tree-reweighted message passing algorithm (TRW-S)* [TPAMI'15].
- Generally guarantees an optimal solution; outperforms others algorithms in heavy tasks.

Case Study– Stuxnet Propagation



(a) A four-zone networked ICS architecture

[Tofino Security'11]

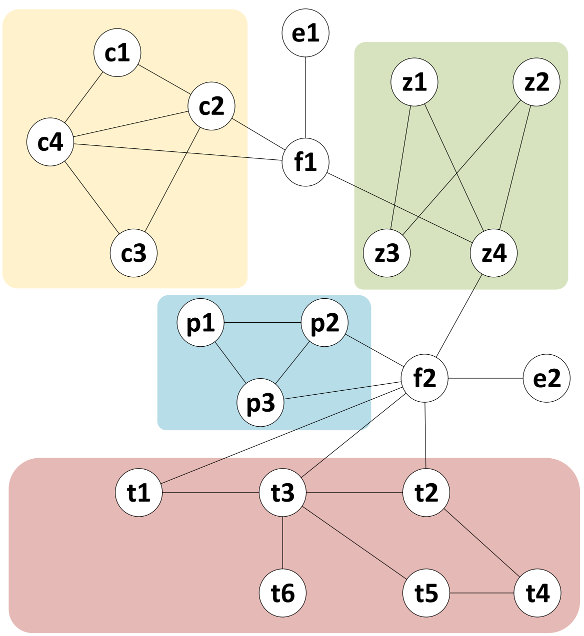


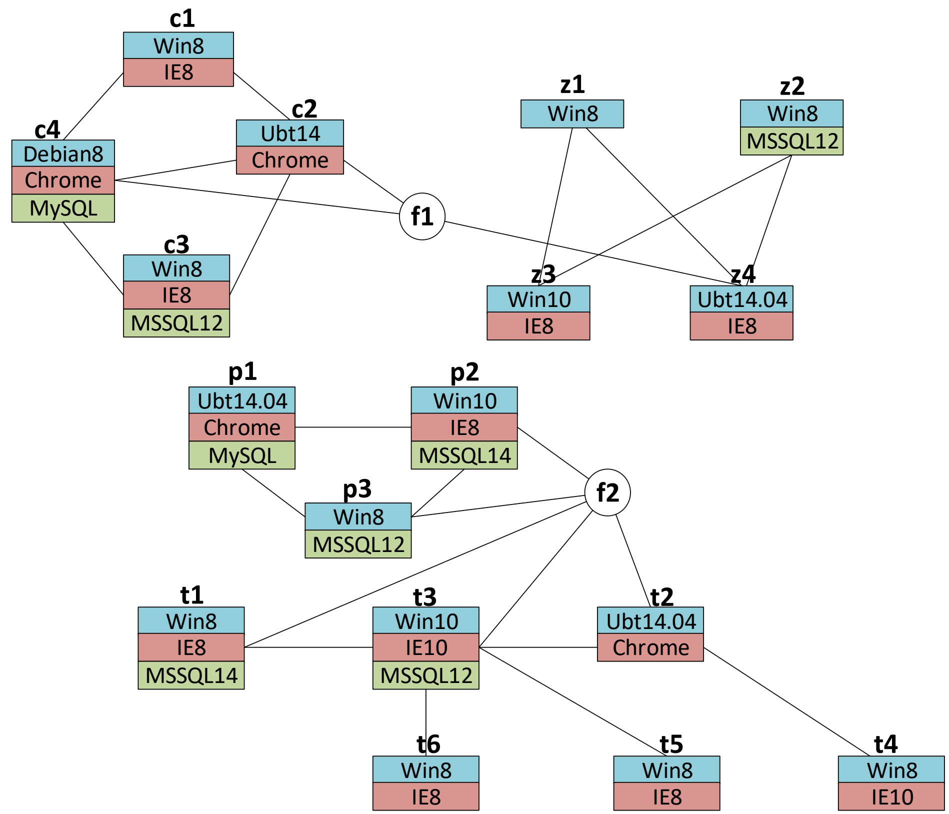
(b) The corresponding network topology

Services	Products	c1	c2	c3	c4	z1	z2	z3	z4	p1	p2	p3	t1	t2	t3	t4	t5	t6
s_1: OS	Windows 8	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Windows 10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ubuntu 14.04		✓		✓	✓			✓	✓				✓				
	Debian 8.0				✓									✓				
s_2: Web Browser	IE8	✓	✓	✓	✓			✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
	IE10	✓	✓	✓	✓			✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
	Chrome 50		✓		✓				✓	✓				✓				
s_3: Database Server	MS SQL 2012			✓	✓		✓			✓	✓	✓	✓		✓			
	MS SQL 2014			✓	✓		✓			✓	✓	✓	✓		✓			
	MySQL 5.5				✓					✓								
	MariaDB 10				✓					✓								

(c) Available products for each host [WinCC Manual]

Case Study– Stuxnet Propagation



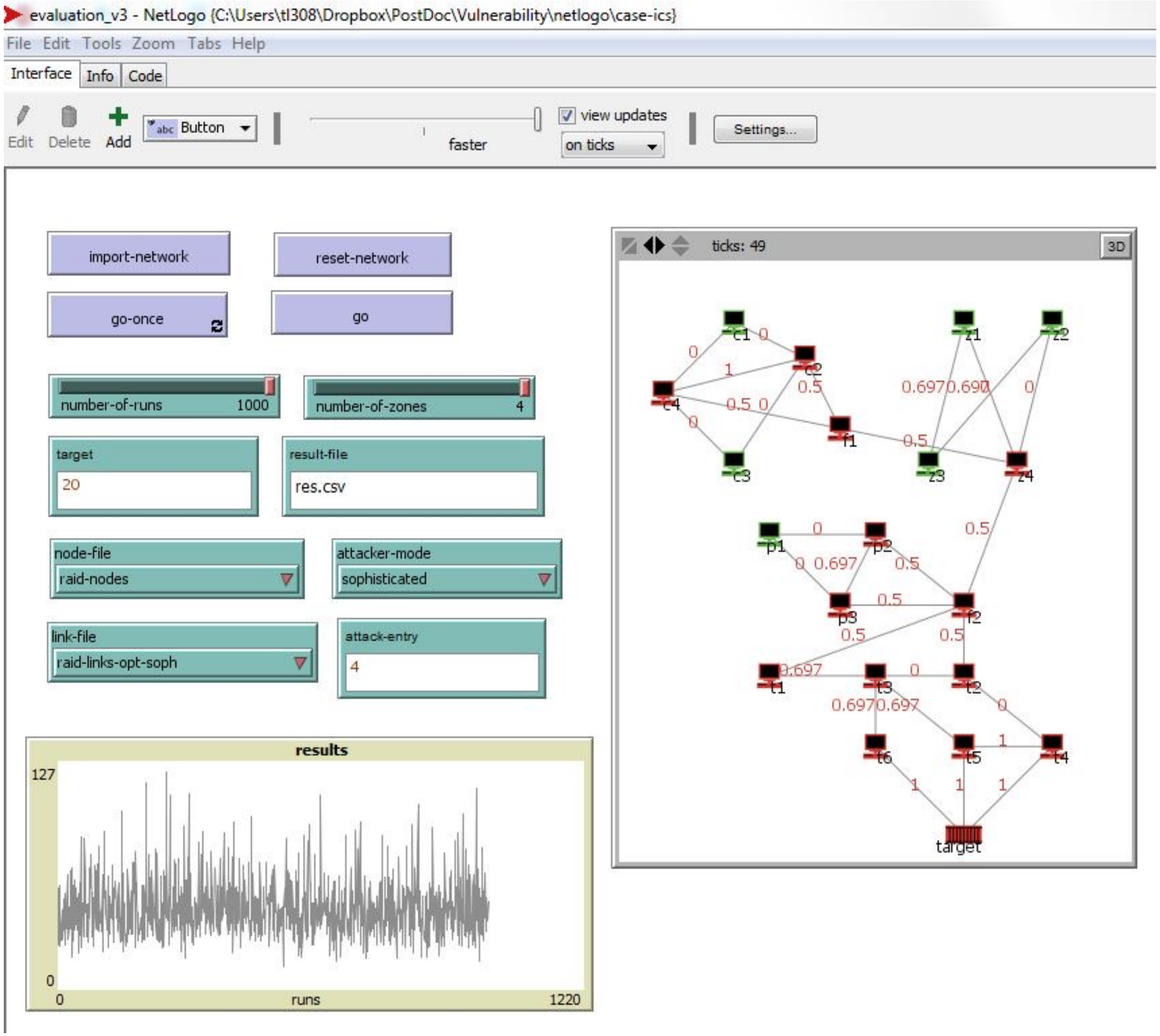


- The solution uses product similarities obtained from CVE/NVD statistically.
- Minimizes the infection rate at each edge by choosing most diverse product pairs.
- Locally optimal assignment might be discarded for global optimum *(e.g. c4 and c2).*

Case Study– Stuxnet Propagation

NetLogo Simulation for Evaluation

- NetLogo is an agent-based modelling tool.
- Programmable modelling environment
- Simulate behaviours of systems and natural phenomena over time.

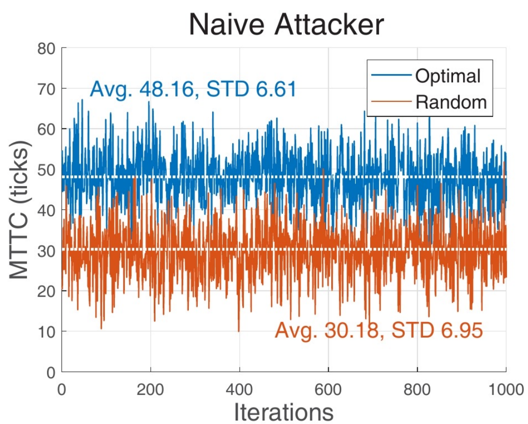


- Two attacker modes.
- # Ticks = MTTC
- Six hosts are seamlessly protected.
- Real-time plot of MTTC

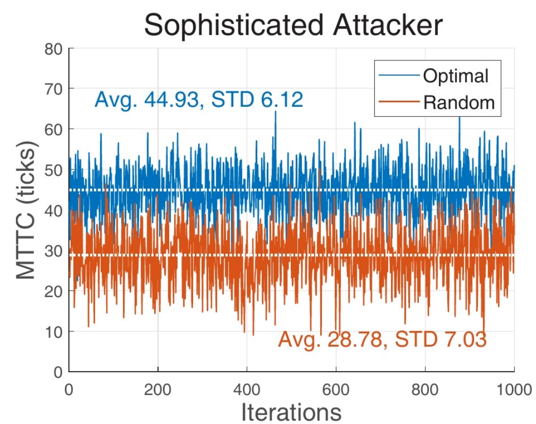
Case Study– Stuxnet Propagation

NetLogo Simulation for Evaluation

- Compare the OPTIMAL assignment with a RANDOM assignment in terms of MTTC.

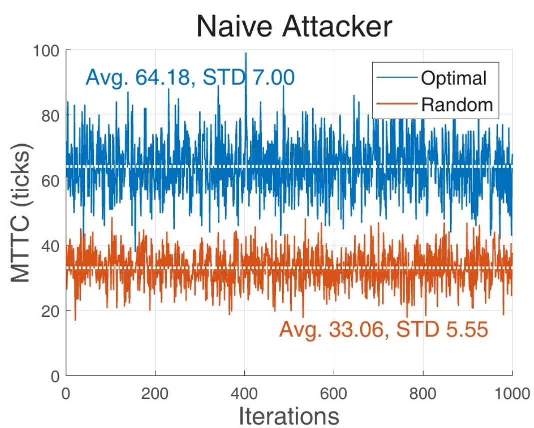


(a)

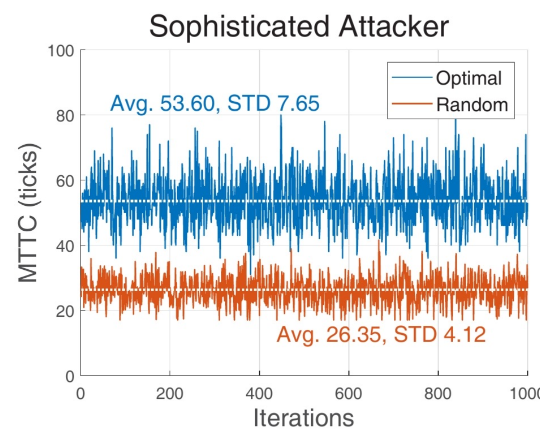


(b)

- An extra experiment with a larger network (100 nodes, ~300 edges and 5 services per host).



(a)

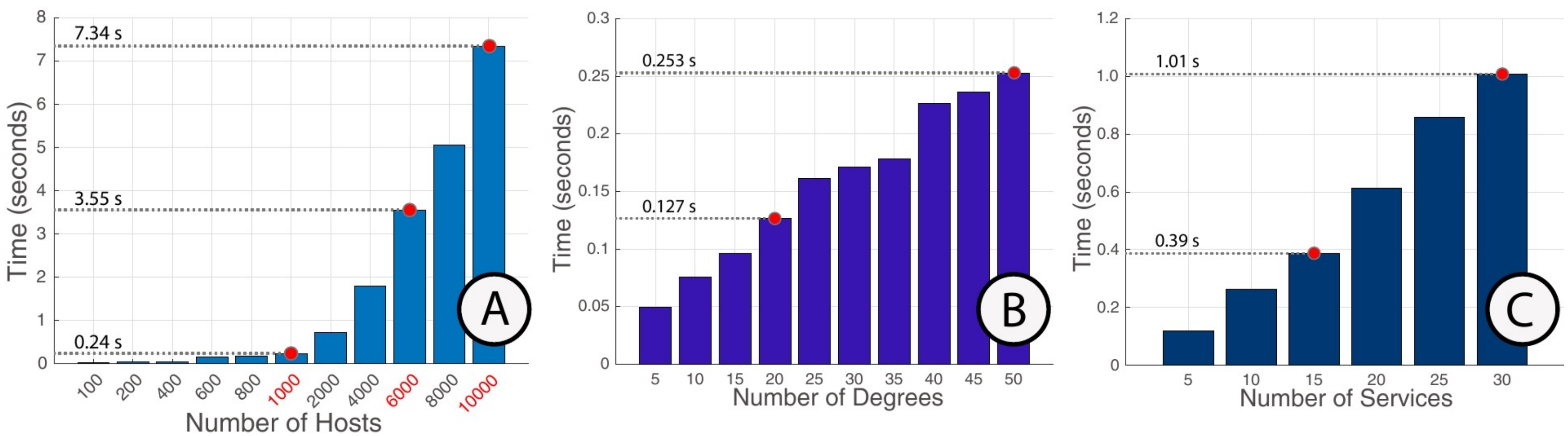


(b)

Scalability Analysis on Random Networks

Scalability with Randomly generated networks

- Computational time consumed by optimising a set of randomly generated networks.
- Three key parameters: # hosts, # degrees (edges per host), # services per host.



- (A) fixed # degree = 3, # services = 3
- (B) fixed # hosts = 100, # services = 3
- (C) fixed # hosts = 100, # degrees = 30
- Mid-range computer: Intel i5 2.8GHz CPU, a 8GB RAM and a nVidia GTX 750
- The # hosts has a major impact. Still converged within 7.342 seconds for 10,000 hosts.

Scalability Analysis on Random Networks

Scalability with Randomly generated networks

- Scalability analysis against high-density, high-complexity and large-scale networks.
- Performs well and converges within about 3 minutes.

Table 1: Computational time (in seconds) for networks of various densities over different # hosts

	# deg.	# serv.	# hosts								
			100	**200**	**400**	**600**	**800**	**1000**	**2000**	**4000**	**6000**
mid-density	**20**	**15**	0.239	0.438	1.099	1.478	1.944	2.784	6.706	16.517	33.392
high-density	**40**	**25**	0.640	1.766	3.553	5.881	8.135	10.999	27.484	82.500	151.110

Table 2: Computational time (in seconds) for various sizes of networks over different # degrees

	# hosts	# serv.	#deg.									
			5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
mid-scale	**1000**	**15**	0.759	1.577	1.954	2.693	3.294	4.040	4.652	5.174	5.758	6.309
large-scale	**6000**	**25**	21.239	40.940	59.216	77.583	95.750	117.810	144.470	152.040	167.190	189.710

Table 3: Computational time (in seconds) for various sizes of networks over different # services

	# hosts	# deg.	# edges	#serv.					
				5	**10**	**15**	**20**	**25**	**30**
mid-scale	**1000**	**20**	$\sim$ **20,000**	0.603	1.608	2.709	4.008	5.253	6.974
large-scale	**6000**	**40**	$\sim$ **240,000**	10.306	27.214	51.587	90.407	134.340	188.050

Conclusion and Future work

- Proposed an efficient way to mitigate zero-day infection over a network by optimally diversifying products deployed on the hosts
- Introduced the similarity metric to capture how similar the vulnerabilities of two products are. Applied in statistical study on CVE/NVD database.
- Estimated the infection rate of malware between products by the similarity metrics.
- A multi-label model is adopted to capture the spread of multiple zero-day exploits.
- Developed an efficient and scalable optimisation method.
- Other sources/ways to measure the similarity metric.
- Consider interdependency between services and preference over products.

Intrusion Detection

Work by Deeph Chana and Feng Cheng

NIDS – Anomaly Detection (Cheng, Li and Chana)

Package Level Detection by *Bloom Filter*

- Construct a *signature database* by observing regular communication patterns.
- Incorporate the signature database into the *bloom filter detector.*
- *Detect* anomalous data packets *at packet-content level.*

Time-series Level Detection by *Long Short Term Memory (LSTM)*

- Address *temporal dependence* between consecutive packets
- *Learn the most likely* packet signatures from seen packets by *LSTM*.
- *Further classification* of packets *at time-series level.*

Evaluation by Public ICS Database and Comparison

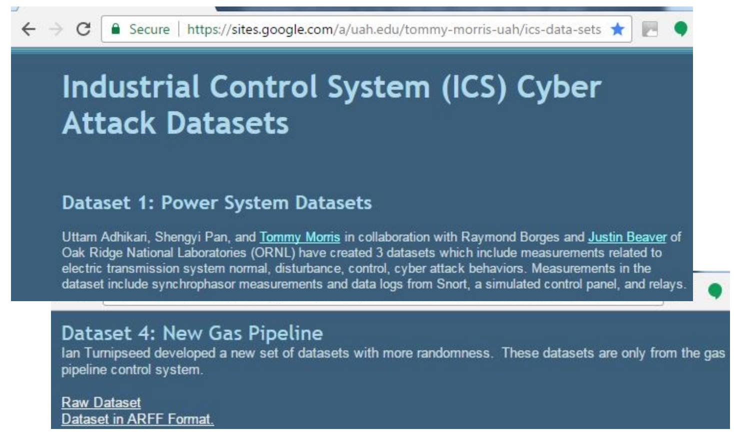
- Apply to a public *ICS dataset* created from a SCADA system for a gas pipeline.
- Significantly outperform other existing approach and produce *state-of-the-art* results.

Public ICS Dataset by Mississippi State SCADA Lab

Mississippi ICS Attack Dataset

https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

- A lab-scale testbed of a gas pipeline SCADA
- Deep inspection of Modbus data log
- **214,580** normal packets + **60,048** attack packets
- **20** unique features in *ARFF* Format.
- **7** common types of attacks.


Secure | https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

Industrial Control System (ICS) Cyber Attack Datasets

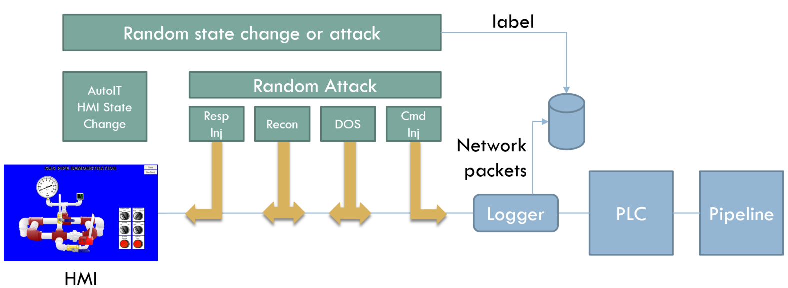
Dataset 1: Power System Datasets

Uttam Adhikari, Shengyi Pan, and Tommy Morris in collaboration with Raymond Borges and Justin Beaver of Oak Ridge National Laboratories (ORNL) have created 3 datasets which include measurements related to electric transmission system normal, disturbance, control, cyber attack behaviors. Measurements in the dataset include synchrophasor measurements and data logs from Snort, a simulated control panel, and relays.

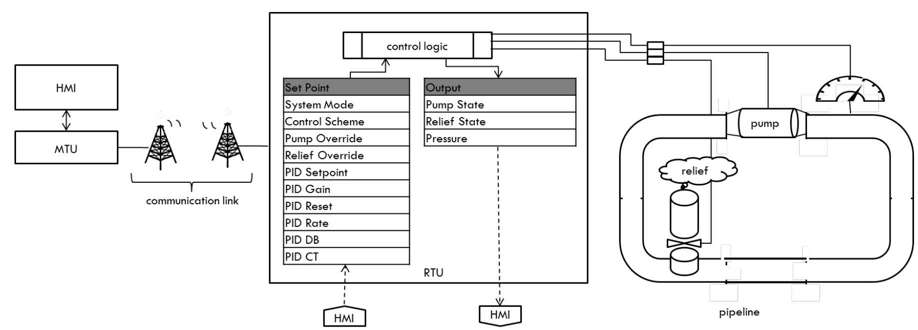
Dataset 4: New Gas Pipeline

Ian Turnipseed developed a new set of datasets with more randomness. These datasets are only from the gas pipeline control system.

Raw Dataset
Dataset in ARFF Format.



Data Collection Model *(Turnipseed 2015)*

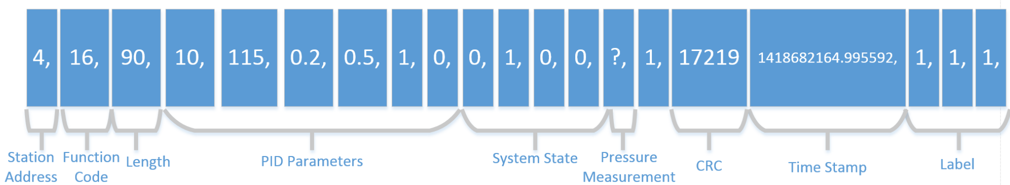


A gas pipeline SCADA *(Turnipseed 2015)*

Public ICS Dataset by Mississippi State SCADA Lab

Mississippi ICS Attack Dataset

ARFF Data Format



Seven Types of Attacks

Type of Attacks	Abbreviation
Normal	Normal(0)
Naïve Malicious Response Injection	NMRI(1)
Complex Malicious Response Injection	CMRI(2)
Malicious State Command Injection	MSCI(3)
Malicious Parameter Command Injection	MPCI(4)
Malicious Function Code Injection	MFCI(5)
Denial of Service	DOS(6)
Reconnaissance	Recon(7)

Feature	Type
address	Network
function	Command Payload
length	Network
setpoint	Command Payload
gain	Command Payload
reset rate	Command Payload
deadband	Command Payload
cycle time	Command Payload
rate	Command Payload
system mode	Command Payload
control scheme	Command Payload
pump	Command Payload
solenoid	Command Payload
pressure measurement	Response Payload
crc rate	Network
command response	Network
time	Network
binary attack	Label
categorized attack	Label
specific attack	Label

Package Level Detection by Bloom Filters

Signature database

- observe a large *normal* time-series dataset.
- A sequence of data packets:

$$\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$$

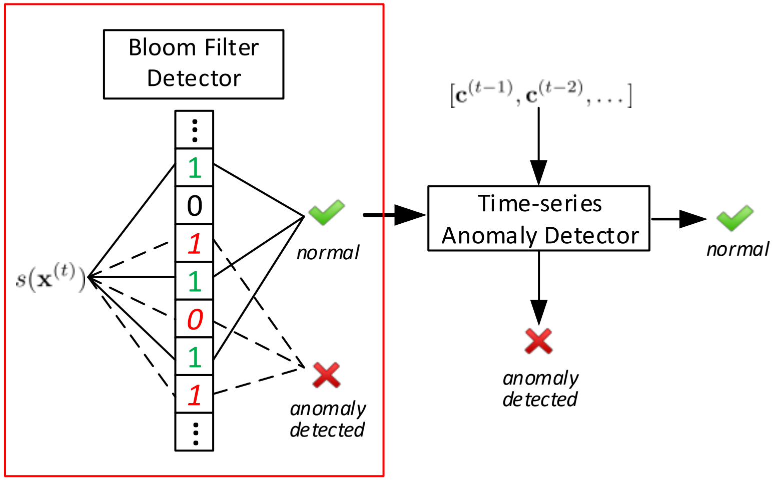
- A single packet with ***m*** features:

$$\mathbf{x}^{(t)} = \{x_1^{(t)}, x_2^{(t)}, \ldots, x_m^{(t)}\}$$

- A signature of a packet:

$$s(\mathbf{x}^{(t)}) = g(c_1^{(t)}, c_2^{(t)}, \ldots, c_o^{(t)})$$

- Assign a unique value to each different combination of original features.



Feature Discretization

- Find optimal granularity of discretization (too coarse --> high FN; too fine --> high FP)

$$\underset{n_1, n_2, \ldots, n_l}{\operatorname{argmax}} \sum_{i=1}^{l} w_i n_i, \quad err_v < \theta$$

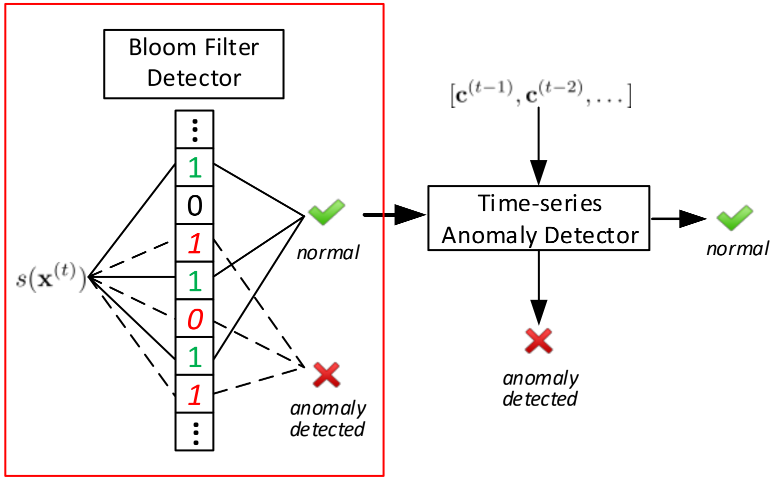
$$n_1, \ldots, n_l \text{ – the number of discretized values;} \quad err_v \text{ – validation error/ FP rate}$$
$$w_1, \ldots, w_l \text{ – relative importance of a feature;} \quad \theta \text{ – acceptable FP rate}$$

- Find the finest-grained discretization whose FP rate is below the acceptable FP.

Package Level Detection by Bloom Filters

Bloom-filter (BF) Anomaly Detector

- BF is a light-weight data structure; test if an element is a member of a set.
- Insert all the normal packet signatures into the BF detector
- Hash functions map each element to the corresponding positions of a bit vector.



- Lookup an element by hashing it by the same functions, and check:

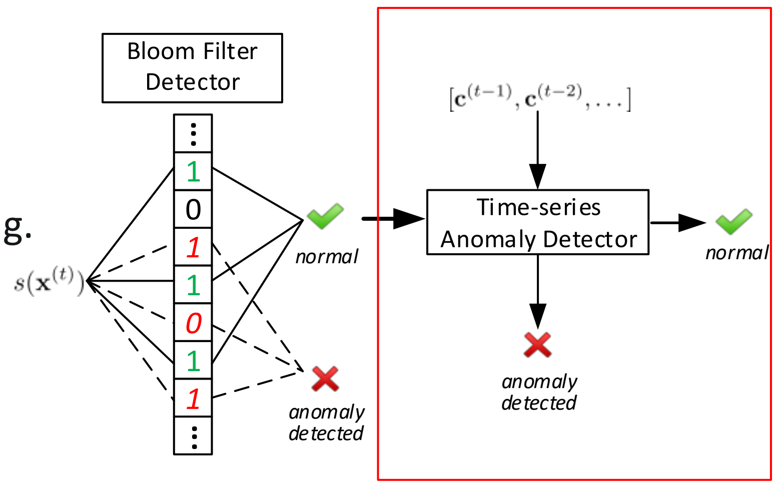
$$F_p(\mathbf{x}^{(t)}) = \begin{cases} 1 & \text{if } s(\mathbf{x}^{(t)}) \notin \mathcal{B} \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{x}^{(t)}$ is classified as **anomaly** if $F_p(\mathbf{x}^{(t)}) = 1$,
- otherwise the package passed our package level anomaly detector.

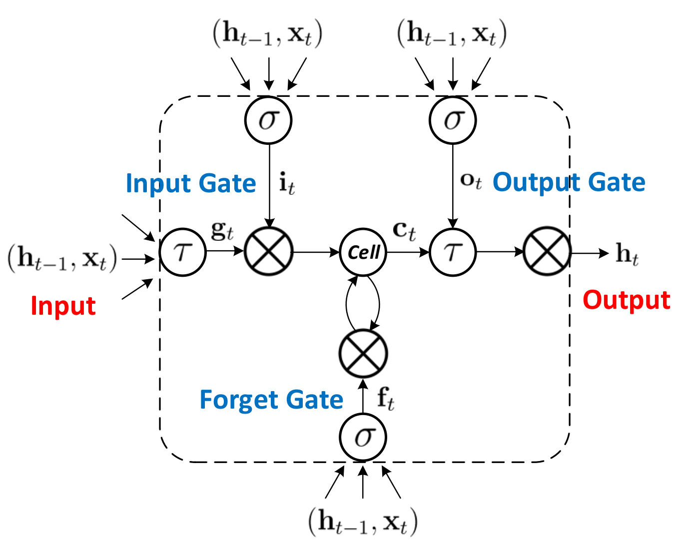
Time-series Level Detection by LSTM

Long Short-Term Memory (LSTM)

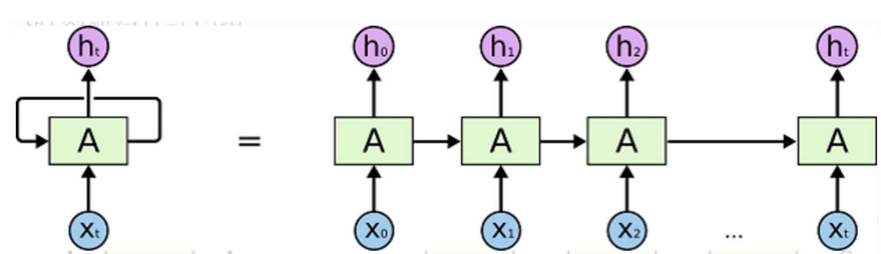
- Advanced anomalies can only be detected by observing preceding packets.
- Recurrent neutral network (RNN) specialise in sequence learning to predict time series.
- Memorize/back-propagate through time for training.
- Identify anomalies with long time lags in between.



A Memory Cell in LSTM



Time Series Learning of LSTM



Time-series Level Detection by LSTM

Stacked LSTM Anomaly Detector

- Store the normal signature database into the LSTM detector
- Input previous network packets (in discretized representation)
- Output the predicted probability of each signature of next packet

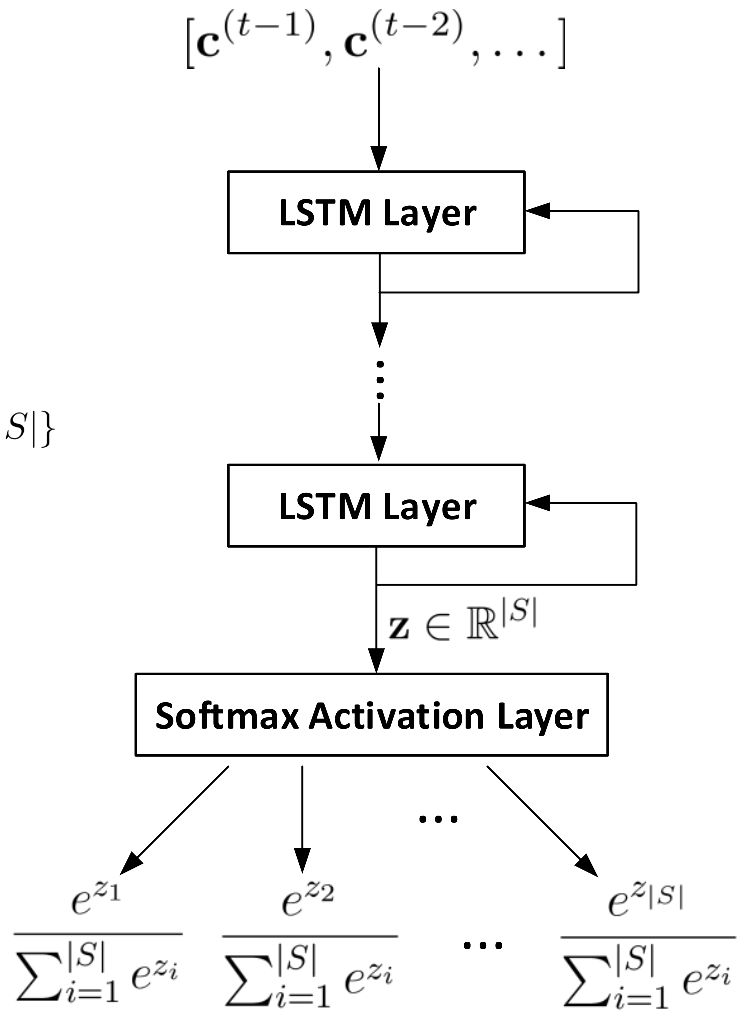
$$\Pr(s_i \mid \mathbf{c}^{(t-1)}, \mathbf{c}^{(t-2)}, \ldots) = \frac{e^{z_i}}{\sum_{k=1}^{|S|} e^{z_k}} \quad \forall i \in \{1, 2, \ldots, |S|\}$$

$$\sum_{i=1}^{|S|} \Pr(s_i \mid \mathbf{c}^{(t-1)}, \mathbf{c}^{(t-2)}, \ldots) = 1.$$

- The top *kth* most probable signatures are used to classify:

$$F_t(\mathbf{x}^{(t)} \mid \mathbf{c}^{(t-1)}, \mathbf{c}^{(t-2)}, \ldots) = \begin{cases} 1 & \text{if } s(\mathbf{x}^{(t)}) \notin S^{(k)} \\ 0 & \text{otherwise.} \end{cases}$$

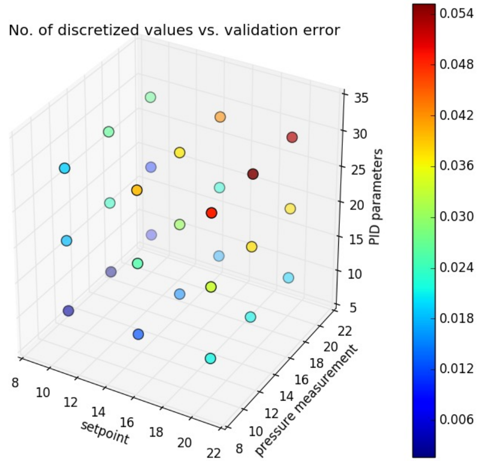
- Optimal choice of *k;* similar as granularity of discretization.
- Add probabilistic noise in training to weaken overfitting.



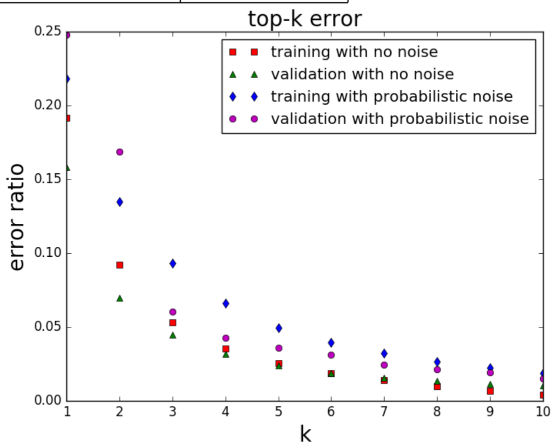
Experiments – Training & Validation

Training & Validation

- Split dataset: 60% training + 20% validation + 20% testing
- Discretization of continuous features by *k-means* and the *validation error.*



Feature	Discretization method	Value No.
time interval	Kmeans clustering	2+1
crc rate	Kmeans clustering	2+1
pressure measurement	Even interval partition	20+1
setpoint	Even interval partition	10+1
PID parameters	Kmeans clustering	32+1



- *A stacked two layer LSTM; each layer has 256 memory units; output 613 signatures.*
- Train the LSTM with/without noise for 50 training epochs.
- Error ratio converges quickly to 0.
- Similar top-k (after k>3) error for models with/without noise.

Precision, Recall, Accuracy and …

- Precision is the fraction of positive elements that are true positives.
- Recall: is the fraction of relevant (false negative) elements that are classified as true positives.
- Accuracy is the fraction of the whole sample that is correctly classified as either a true positive or a true negative.

… F1

- F-measure combines precision and recall. The most common measure is the F1-measure which is computed as:

$$\frac{2 \text{ x precision x recall}}{\text{precision + recall}}$$

- which can be interpreted as a weighted average.

Experiments – Comparison

Comparison with Other Anomaly Detection Methods

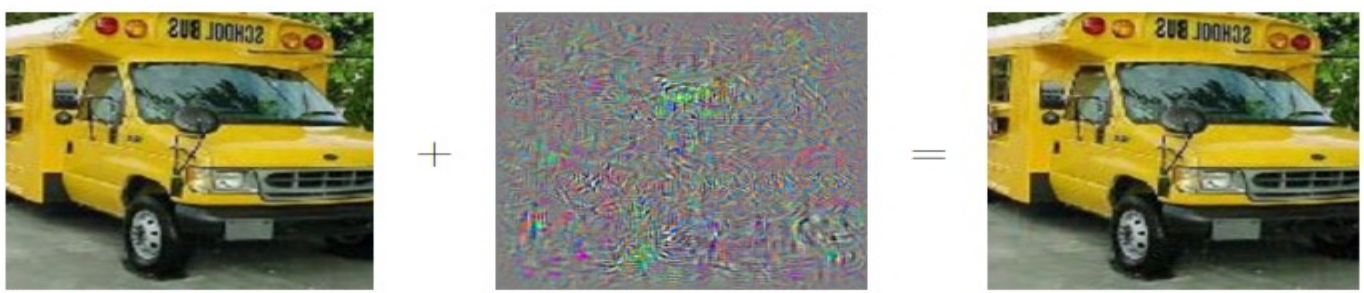
- Evaluation metrics – *Precision, Recall, Accuracy and F-score.*
- Compare with other anomaly detection methods.
- Detected ratio (recall) for seven types of attacks.

Model	Precision	Recall	Accuracy	F-score
Our model	0.94	**0.78**	**0.92**	**0.85**
BF	**0.97**	0.59	0.87	0.73
BN	**0.97**	0.59	0.87	0.73
SVDD	0.95	0.21	0.76	0.34
IF	0.51	0.13	0.70	0.20
GMM	0.79	0.44	0.45	0.59
PCA-SVD	0.65	0.28	0.17	0.27

Attack Type	Model	Detected Ratio
NMRI	Our model	**0.88**
	BF	0.77
	BN	0.77
	SVDD	0.01
	IF	0.13
	GMM	0.31
	PCA-SVD	0.45
CMRI	Our model	**0.67**
	BF	0.53
	BN	0.53
	SVDD	0.02
	IF	0.08
	GMM	0.33
	PCA-SVD	0.19
MSCI	Our model	0.62
	BF	0.18
	BN	0.53
	SVDD	0.19
	IF	0.46
	GMM	**0.66**
	PCA-SVD	0.62
MPCI	Our model	**0.80**
	BF	0.49
	BN	0.34
	SVDD	0.26
	IF	0.08
	GMM	0.64
	PCA-SVD	0.66
MFCI	Our model	**1.00**
	BF	**1.00**
	BN	**1.00**
	SVDD	**1.00**
	IF	0.00
	GMM	0.32
	PCA-SVD	0.54
DOS	Our model	**0.94**
	BF	0.93
	BN	0.93
	SVDD	0.40
	IF	0.12
	GMM	0.15
	PCA-SVD	0.58
Recon.	Our model	**1.00**
	BF	**1.00**
	BN	**1.00**
	SVDD	**1.00**
	IF	0.12
	GMM	0.72
	PCA-SVD	0.54

Evasion Attacks

Originally discovered by researchers when trying to better interpret neural networks.



Schoolbus Perturbation Ostrich

Szegedy, Christian, et al. "Intriguing properties of neural networks." (2013).

Stealthy Attacks (Cheng, Li and Chana)

Objectives

- A framework for conducting stealthy attacks with minimal knowledge of the target ICS

- Better understanding of the limitations of current detection mechanisms, and the real threat posed by stealthy attacks to ICS.

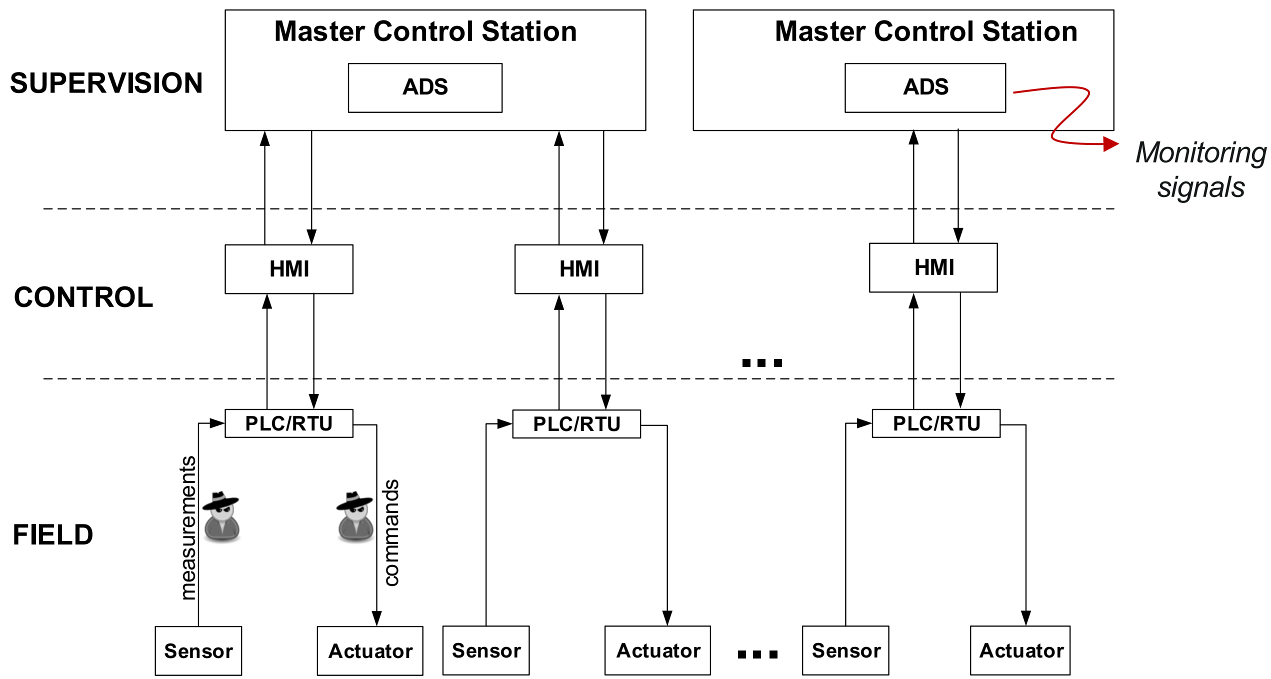
Main Contributions

- Demonstrated attacks can be automatically achieved by intercepting the sensor/control signals for a period of time using a particularly designed real-time learning method.

- Used adversarial training technique – Wasserstein GAN to generate false data that can successfully bypass the IDS and still deliver specific attack goals.

- Two real-world datasets are used to validate the effectiveness of our framework.
    - A gas pipeline SCADA system.
    - Secure Water treatment testbed from iTrust@SUTD.

Stealthy Attacks against ICS



- Intercept the expected behaviours of the system via compromised channels.
- *Injected malicious sensor reading* at each time step to achieve certain attack goals.
- Attackers attempt to hide their manipulation; *remain undetected by ADS.*

Anomaly Detection for Industrial Processes

Anomaly Detection Mechanism for Securing Industrial Processes

- Protect from physical faults and cyber attacks by monitoring the sensor reading & control commands.
- Rely on a Predictive Model which predicts the next sensor measurements based on previous signals.

$$\textit{Observed:}\quad \mathbf{x}^{(t)} = \{\mathbf{y}^{(t)}, \mathbf{u}^{(t)}\} = \{y_1^{(t)}, y_2^{(t)}, \ldots, y_m^{(t)},\ u_1^{(t)}, u_2^{(t)}, \ldots, u_n^{(t)}\}$$

$$\textit{Predicted:}\quad \hat{\mathbf{y}}^{(t)} = \{\hat{y}_1^{(t)}, \hat{y}_2^{(t)}, \ldots, \hat{y}_m^{(t)}\}$$

Existing Predictive Models

- Auto-Regressive (AR) model
  - Fit a linear regression model for each reading based on its *p* previous readings.

$$\boxed{\hat{y}_i^{(t)} = \sum_{j=1}^{p} \alpha_j y^{(t-j)} + \alpha_0 \quad \forall i \in \{1, 2, \ldots, m\}.}$$

- Linear Dynamic State-space (LDS) model
  - A vector *w* for physical states
  - Matrices for system dynamics.
  - Noise vectors.
  - Known as the *State Estimator*

$$\begin{aligned} \mathbf{w}^{(t)} &= A\,\mathbf{w}^{(t-1)} + B\,\mathbf{u}^{(t-1)} + \epsilon^{(t-1)} \\ \hat{\mathbf{y}}^{(t)} &= C\,\mathbf{w}^{(t)} + D\,\mathbf{u}^{(t)} + \varepsilon^{(t)} \end{aligned}$$

- Long Short-Term Memory (LSTM) *[HASE'17]*
  - State-of-the-art prediction accuracy
  - hidden vector computed iteratively
  - Weight matrix and bias vector

$$\begin{aligned} \mathbf{h}^{(t)} &= f(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}) \\ \hat{\mathbf{y}}^{(t)} &= W_y \mathbf{h}^{(t-1)} + \mathbf{b}_y \end{aligned}$$

Anomaly Detection for Industrial Processes

Detection Methods

- An anomalous signal is detected when the residual error between the predicted and observed:

$$|\hat{y}_i^{(t)} - y_i^{(t)}| > \tau_i \quad \forall i \in \{1, 2, \ldots, m\}$$
$$\|\hat{\mathbf{y}}^{(t)} - \mathbf{y}^{(t)}\| > \tau$$

- Use the history of residual errors to detect collective anomalies, by Cumulative Sum (CUSUM) based on an accumulated statistic:

$$H^{(t)} \quad = \quad \max(0, H^{(t-1)} + r^{(t)} - \mu - \omega) > \tau$$

Formally Define Stealthy Attacks

- A general anomaly detector as a function:

$$\mathcal{F}(\mathbf{y}^{(t)} \mid \mathbf{X}^{t-1}) = \begin{cases} 1 & \text{if } \mathbf{y}^{(t)} \text{ triggers an alarm} \\ 0 & \text{otherwise} \end{cases}$$

- Consider the ADS is a black-box for attackers.
- A certain number of PLC-sensor and PLC-actuator channels $\mathbf{y}_c^{(t)}$ are compromised.
- Define a set of stealthy attack goals -- inject malicious values that are deviant with the true values

$$\tilde{y}_g^{(t)} ⓖ v_g \quad \text{where } ⓖ \in \{>, <, \leq, \geq, =\} \wedge \tilde{y}_g^{(t)} \in \tilde{\mathbf{y}}_c^{(t)}$$

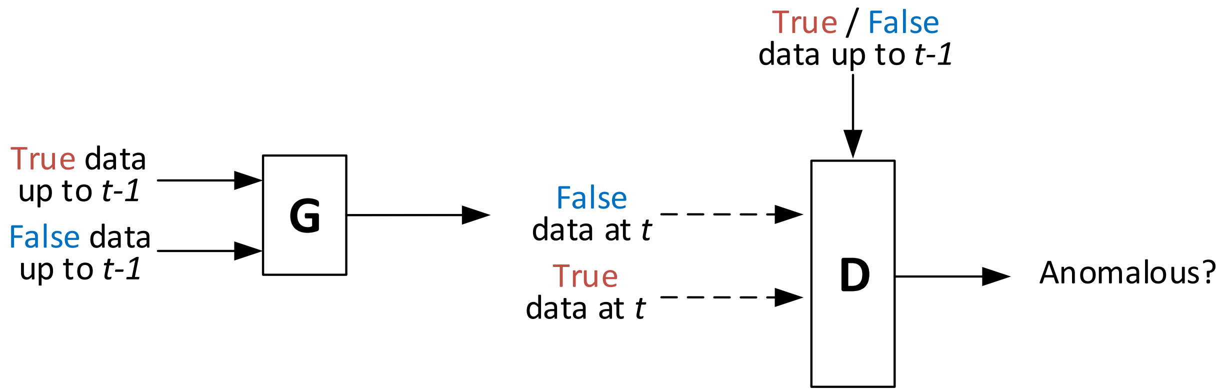
Deep Learning Framework for Stealthy Attacks

WGAN-based Training Model for Stealthy Attacks

- Reconnaissance phase + Attacking phase

- A Wasserstein Generative Adversarial Net (W-GAN) is constructed for training

- W-GAN: an iterative game between two players (Generator and Discriminator)
  - **G**enerator: generates data with the same distribution as the training data
  - **D**iscriminator: distinguish generated data from training data
  - At each step, either **G** or **D** is trained to optimize its objective function.
  - Until **D**iscriminator fails…



Deep Learning Framework for Stealthy Attacks

Malicious Measurement Generator

- Generate malicious data achieving attack goals.
- As a sequence learning problem, solved by LSTM-FNN.
- Two sliding windows:

$$\mathbf{S}_c^t = \{\mathbf{x}_c^{(t-l)}, \mathbf{x}_c^{(t-l+1)}, \ldots \mathbf{x}_c^{(t-1)}\} \quad \tilde{\mathbf{S}}_c^t = \{\tilde{\mathbf{x}}_c^{(t-l)}, \tilde{\mathbf{x}}_c^{(t-l+1)}, \ldots \tilde{\mathbf{x}}_c^{(t-1)}\}$$

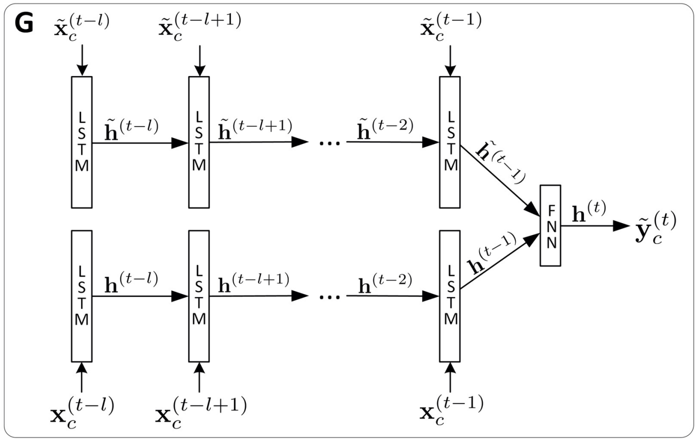
- Generator as an overall function:

$$\tilde{\mathbf{y}}_c^{(t)} = G(\mathbf{S}_c^t, \tilde{\mathbf{S}}_c^t; \Theta_G)$$

- Minimize the chance being detected and deliver the goal:

$$\arg\min_{\Theta_G} \frac{1}{|T|} \sum_{t \in T} \mathcal{F}(G(\mathbf{S}_c^t, \tilde{\mathbf{S}}_c^t; \Theta_G) \mid \mathbf{X}^{t-1})$$

$$\text{subject to} \quad \tilde{y}_g^{(t)} \textcircled{g} v_g \quad \forall g \in \mathcal{G}, t \in T$$

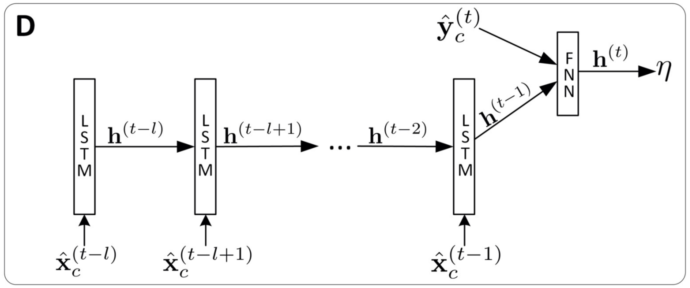


Substitute Anomaly Detector

- Approximate the black-box anomaly detector.
- Input the window of previous data and the current data;
  Output the classification. $\eta = D(\hat{\mathbf{y}}_c^{(t)}, \hat{\mathbf{S}}_c^t; \Theta_D)$
- Detector as an overall function:

$$\arg\min_{\Theta_D} \frac{1}{|T_1|} \sum_{t \in T_1} D\big(\mathbf{y}_c^{(t)}, \mathbf{S}_c^t; \Theta_D\big) - \frac{1}{|T_2|} \sum_{t \in T_2} D\big(\tilde{\mathbf{y}}_c^{(t)}, \tilde{\mathbf{S}}_c^t; \Theta_D\big)$$

- Larger values for malicious; smaller values for true data.



GAN: Generator + Detector

- Generating malicious data which makes the detector output smallest possible values whilst achieving the goals.

GAS Pipeline Case Study

Mississippi Dataset of a gas pipeline SCADA

- Controls the air pressure in a pipeline; contains a PLC, a sensor and several actuators.
- Pressure measurements at every 2s, 68,803 time series signals are collected.

Features	Description
Setpoint	The pressure set point
Gain	PID gain
Reset rate	PID reset rate
Deadband	PID dead band
Cycle time	PID cycle time
Rate	PID rate
System mode	Automatic(2), manual (1) or off (0)
Control scheme	Pump (0) or valve (1)
Pump	Open(1) or off (0) – for manual mode
Valve	Open(1) or off (0) – for manual mode
Pressure measurement	Pressure measurement

Experiment Setup

- Baseline Anomaly Detector uses LSTM model.
- Four Attack Scenarios: being 4 or 8 units smaller than real values; different compromised channels

		Attack Goal	
		$\tilde{y}_g^{(t)} = max(y_g^{(t)} - 4, 0)$	$\tilde{y}_g^{(t)} = max(y_g^{(t)} - 8, 0)$
Attacker's Abilities	PLC-Sensor channel Compromised	Attack Scenario 1	Attack Scenario 2
	All channels Compromised	Attack Scenario 3	Attack Scenario 4

GAS Pipeline Case Study

Results and Evaluation

- Generated malicious measurements successfully capture the trend of the real trace.

- Generated malicious measurements mostly can bypass the anomaly detector
  - Most malicious values have similar or less residual error than the true values.
  - Outliers are caused by HMI human input at manual mode.

- Ratio of attack goal achieved the detection ratio of malicious measurements
  - Ignored the outliers (residual error > 0.05)
  - Less detection ratio for attack scenario 3 and 4.
  - Only compromising PLC-sensor channel still generates high-quality attacks.

Attack Scenario	Ratio of goal achieved	Detected ratio	
		by residual error	by CUSUM
1	88.1%	2.6%	0.2%
2	86.0%	2.4%	0.1%
3	85.9%	1.1%	0.01%
4	90.5%	1.2%	0.01%

UK/Singapore cyber security research

Security by Design for
Interconnected Critical Infrastructures

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Water Treatment System Case Study

Experiment Setup

- A water treatment plant (SWaT from iTrust@SUTD) maintains the water quality within acceptable limits.
- 51 sensors extracted every second, in total 496,800 signals for normal operation are collected.

Features	Description
AIT201	Measures NaCl level
AIT202	Measures HCl level
AIT203	Measures NaOCl level
FIT201	Flow transmitter for dosing pumps
P101	Raw water tank pump state
MV201	Motorized vale state
P201	NaCl dosing pump state
P203	HCl dosing pump state
P205	NaOCl dosing pump state

- Focus on generating malicious HCl and NaOCl measurements, still within normal range.

$$\tilde{y}_{g_1}^{(t)} \geq \min(y_{g_1}^{(t)} + 0.1, 1) \qquad \tilde{y}_{g_2}^{(t)} \leq \max(\tilde{y}_{g_2}^{(t)} - 0.1, 0)$$

Simulation and Evaluation

- A successful attack -- either the HCl (>0.99) or the NaOCl (<0.01) dosing pump is turned on unexpectedly by the injected malicious measurements + bypassed the detector.

Compromised Channels	Successful Ratio	
	by residual error	by CUSUM
Only PLC-AIT202, PLC-AIT203	90.1%	93.8%
all channels	92.4%	94.6%

Future work

- Proposed a novel GAN based stealthy attack framework, required a much lower *a-priori* knowledge of the targeted ICS.
- Developed a real-time adversarial learning method allowing attackers to inject malicious data to automatically conduct stealthy attacks without being detected.
- Indicated that with recent development in deep learning, the widely recognized effectiveness of existing anomaly detection techniques might be overestimated.
- More advanced anomaly detection frameworks are needed.

Thank you

ritics.org

c.hankin@imperial.ac.uk