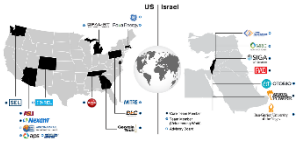
Task 15: Self-healing and auto-remediation

Prof. Asaf Shabtai

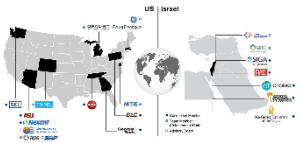
BGU

Problem statement



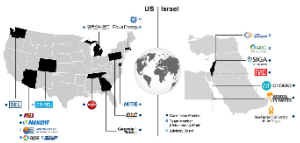
- Faults and cyber attacks on ICS environments may result in physical damage and life threating situations
- Some elements/devices are deployed in distant/remote locations that are difficult to reach and to handle/repair
- ICS environments are complex and understanding which remediation/sequence of remediation to apply may be difficult
- There is a need to recovery of the system, allowing it to return to its normal state with minimal human intervention needed as fast as possible

Challenges



- Many potential remediation options available
- Large number of possible system states
- A seemingly endless number of attacks or malfunctions
- Cases of attacks or malfunctions do not occur frequently
- Automatic remediation should be applied carefully without harming the environment

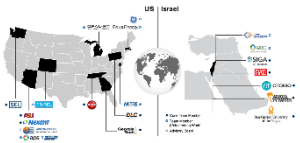
Proposed approach



- Generating remediation “playbooks” for ICS using **attack graphs**

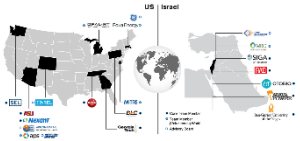
- Allows intelligently defining automatic remediation
  - Selecting remediation and self healing actions that mitigates/recover from the attack/incident
  - Minimal/no impact/risk on the system
  - As close as possible to the incident (but not mandatory)
  - Not necessarily physical location, but also logical one

Attack graphs



- Common vulnerability scanners can identify vulnerabilities present in a host
- Exploiting a single vulnerability is likely to cause significantly less damage than a combination of multiple vulnerabilities, which cannot be realized by these tools
- Analyzing the relationship between vulnerabilities is a complex and expensive task – especially in large-scale network
- Such methods cannot consider multi-host/multi-stage attacks

Attack graphs



- Multihost, multistage vulnerability analysis framework for generating logical attack graph
- How the different (existing) vulnerabilities interconnect to form complete attack paths
- Models the interaction of software vulnerabilities with network configuration
- Based on automatically gathered hosts' information and existing vulnerabilities as well as network connectivity
- General vulnerability information is extracted from publicly available repositories, e.g., NVD

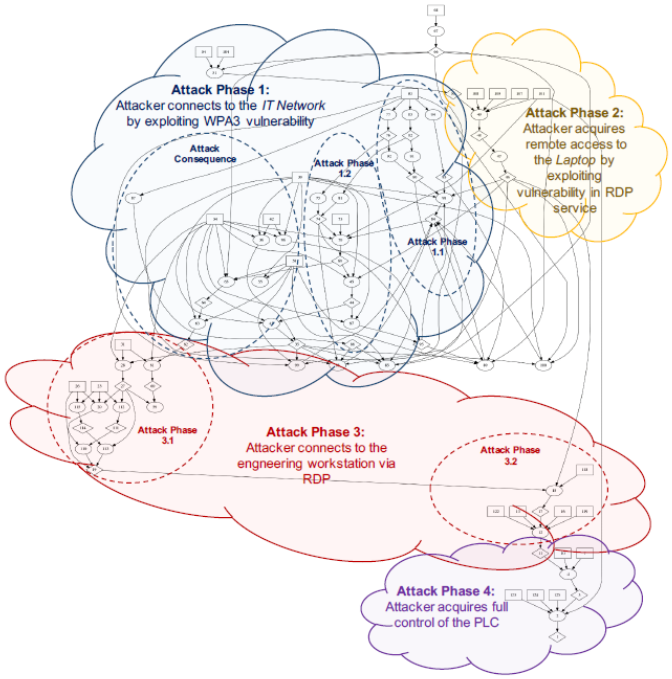


Fig. 12: A multi-stage multi-host attack scenario.

Illustration – logical attack graph

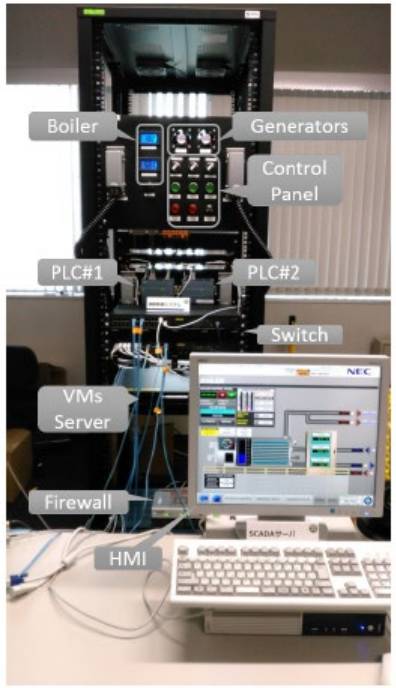


Fig. 6: Operational testbed.

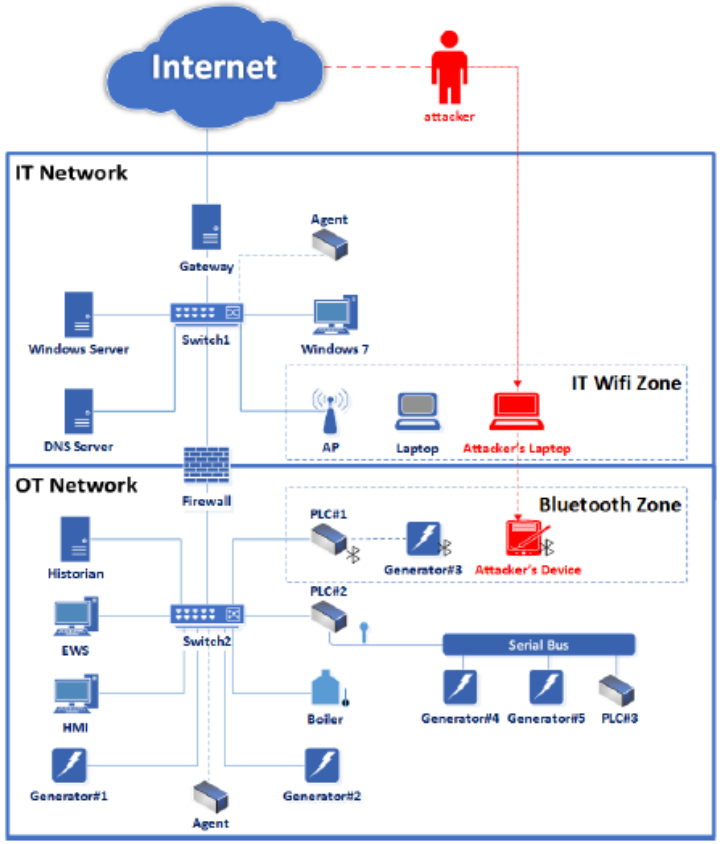
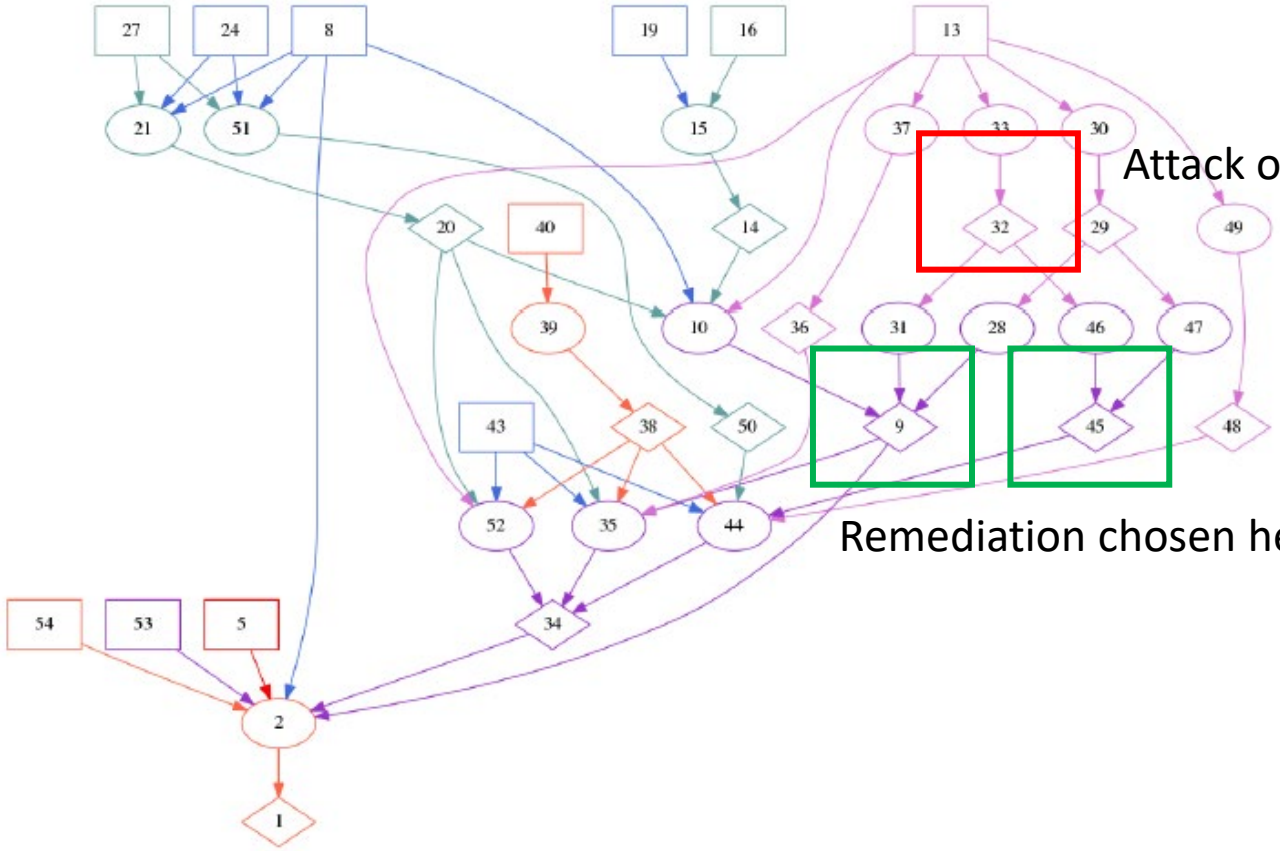


Fig. 5: Testbed network topology.

Illustration – logical attack graph



Attack occurred here

Remediation chosen here

1-2	crackAPEncKey(attacker, 'AP')
5	vulLinkProtocol('IT Wifi Zone', 'VU871675', wpa3_trans, remoteExploit, keyExtraction)
8	isAP('AP', 'IT Wifi Zone', 'IT Network', wpa3_trans, secured)
9-10 9-28 9-31	relay('AP', dragonHandShake)
13	dataFlow('Laptop', 'AP', dragonHandShake, twoWay)
14-15	l2Connection('AP', 'AP', 'IT Network', arp, ipSubnet)
16	existingProtocol('IT Network', arp)
19	located('AP', 'IT Network', ipSubnet)
20-21	l2Connection('Laptop', 'AP', 'IT Wifi Zone', wpa3_trans, wireless)
24	located('Laptop', 'IT Wifi Zone', physical)
27	isAuthenticated(employee, 'Laptop', 'AP')
29-30	dataFlow('AP', 'Laptop', dragonHandShake)
32-33	dataFlow('Laptop', 'AP', dragonHandShake)
34-35 34-44 34-52	accessDataFlow(attacker, dragonHandShake, view)
36-37	dataFlow('Laptop', dragonHandShake)
38-39	localAccess(attacker, 'Attacker Laptop', admin)
40	attackerLocated('Attacker Laptop')
43	located('Attacker Laptop', 'IT Wifi Zone', physical)
45-46 45-47	relay('Laptop', dragonHandShake)
48-49	dataFlow('AP', dragonHandShake)
50-51	l2Connection('AP', 'Laptop', 'IT Wifi Zone', wpa3_trans, wireless)
53	isCredential(dragonHandShake, 'AP', _)
54	malicious(attacker)

grade attack graph.