Edge Computing Security State-of-The-Art and Challenges

Alessandro Carrega

TNT Lab – DITEN
University of Genoa

Information 1/2

- Lecturer: **Dr. Alessandro Carrega**.
  - *Email*: alessandro.carrega@unige.it.
  - *Skype*: alessandro.carrega@gmail.com.
  - *Telegram / Whatsapp*: **3487485497**.
- Duration: **20 hours**.
- Language: **English**.
- Lesson in site and remote (Teams).

Information 2/2

- Dedicated Teams channel:
  - **PhD STIET Cyber security approaches for Cloud/Edge Environments**

    https://teams.microsoft.com/l/team/19%3a-Dtnw_NHUAl1AjZZV4HixIifmU8gywbskeeQwSV--uk1%40thread.tacv2/conversations?groupId=bdafff5c-0ab9-44b2-aef2-5a14e1dd6e15&tenantId=6cd36f83-1a02-442d-972f-2670cb5e9b1a
- GitHub repository:
  - https://github.com/tnt-lab-unige-cnit/phd-stiet-cyber-security-approaches-cloud-edge-environments
- Optional homework.
  - Available in Teams and GitHub.
- Final Exam with 3 options:
  - **Theoretical**: *short survey with 3 papers*.
  - **Pratical**: *2 exercises*.
  - **Quiz**: *100 multiple choice questions (**60%** to pass exam)*.

Introduction

- Edge computing provides more feasible computing technology for smart city applications and beyond.
- Its emergence introduces more security threats since it increases real-world attack surface from following four angles:
  - weak computation power;
  - attack unawareness;
  - OS and protocol heterogeneities;
  - coarse-grained access control.

Weak Computation Power

- Compared to cloud server, computation power of edge server is relatively weaker.
- Edge server is more vulnerable to existing attacks that may no longer be effective against cloud server.
- Similarly, compared to general-purpose computers, edge devices have more fragile defense systems; as consequence, many attacks that may be ineffective against desktop computers can pose serious threats to edge devices.

Attack Unawareness

- Unlike general-purpose computers, majority IoT devices do not have UI interfaces, regardless of fact that some may have crude LED screens
- Therefore, user may have limited knowledge about running status of device,
  - e.g., whether it has been shutdown or compromised.
- Hence, even if attack is taken place in edge device, most users may not be able to discern it.

OS and Protocol Heterogeneities

- Unlike general-purpose computers that tend to use standard OSes and communication protocols such as POSIX, most edge devices have different OSes and protocols without standardized regulation.
- This problem directly leads to difficulties of designing unified protective mechanism for edge computing.

Coarse-Grained Access Control

- Access control models designed for general-purpose computers and cloud computing mainly consist of four types of permissions:
  - No Read & Write, Read Only, Write Only, Read & Write.
- Such model would never be satisfiable in edge computing due to more complicated systems and their enabled applications, which call for fine-grained access control that should handle questions such as “who can access which sensors by doing what at when and how”.
- Unfortunately, current access control models are mostly coarse-grained.

Flooding-Based Attacks Access Control 1/3

- DDoS attacks aiming to shutdown normal service of server based on large amount of flooded malformed/malicious network packets, and are mainly classified as UDP flooding, ICMP flooding, SYN flooding, ping of death (PoD), HTTP flooding, and Slowloris, according to attack techniques.
- In UDP flooding attack, attacker continuously sends large amount of noisy UDP packets to target edge server, causing server incapable of handling benign UDP packets in time and thus interrupting normal UDP services provided by edge server.
- In ICMP flooding attack, attacker exploits ICMP protocol to craft attack by sending large number of ICMP Echo Request packets to target edge server as fast as possible without waiting for replies.
- This type of attack consumes both outgoing and incoming throughputs of victim server since server returns ICMP Echo Reply packet upon each receipt of ping request, resulting in significant system-wide slowdown.

Flooding-Based Attacks Access Control 2/3

- In SYN flooding attack, attacker exploits three-way handshake of TCP protocol by initiating amount of SYN requests with spoofed IP address to target edge server, while server responds with SYN-ACK packet to spoofed IP for each SYN request and waits for confirmation ACK that never comes.
- In PoD attack, attacker creates IP packet with malformed or malicious content that has length greatly larger than maximum frame size for standard IP packet (65,535 bytes), and splits long IP packet into multiple fragments and sends them to target server.
- Upon receipts of fragments, server must reassemble all fragments which end up with IP packet whose size is over 65,535 bytes. If attacker continuously sends large number of such packets to target, computation resources of target server may be completely occupied to reassemble all fragments.

Flooding-Based Attacks Access Control 3/3

- In HTTP flooding attack, attacker simply sends tremendous amount of HTTP GET, POST, PUT, or other legitimate requests to edge server.
- Since edge server is less likely able to handle huge number of legitimate requests in timely manner due to restricted computation power, its normal services can be easily throttled if large amount of HTTP traffics received in short period of time.
- In Slowloris attack, attacker creates numerous partial HTTP connections, which for example can be realized by only sending HTTP headers to target server but never completing one.
- In this case, server keeps all open connections falsely in different concurrent threads/processes until total number hits maximum concurrent connection pool size, causing shutdown of server.

Zero-day DDoS Attacks

- More advanced than flooding-based DDoS mentioned above but it is more difficult to implement.
- Attacker must find unknown vulnerability (i.e., zero-day vulnerability) in piece of code running on target edge server/device, which can cause memory corruption and finally result in service shutdown.
- For instance, vulnerability CVE-2010- 3972 is heap-based overflow that can cause DoS on Internet Information Services (IIS) 7.0 and IIS 7.5.
- This kind of attack is also most difficult one to defend against since it exploits zero-day vulnerability that has not been known to public.

Solutions against Flooding-Based Attacks 1/5

- Detection of flooding-based DDoS attacks can be mainly classified into two categories:
  - per-packet-based detection;
  - statistics-based detection.
- Per-packet based detections aim to detect flooding-based attacks at packet level.
- Intuitively, since flooding-based DDoS attack is launched mainly by sending enormous amount of malicious or malformed network packets, detecting and filtering those packets can have effective defense.
- Integrating packet filtering mechanisms into congestion control frameworks to mitigate attacks.

Solutions against Flooding-Based Attacks 2/5

- When suspicious packet is identified, network can simply drop packet before it arrives at destined edge server.
- Although this may sound trivial, detecting whether packet is DoS-oriented malicious one is never easy.
- Attackers can leverage advanced techniques such as spoofing packets using counterfeit IP/MAC addresses and choosing carefully crafted HTTP headers and agents to make DDoS attacks more stealthy.
- Researchers turned to develop more effective detection schemes.

Solutions against Flooding-Based Attacks 3/5

- Mechanism to spot DDoS on per-packet basis.
- Exploits fact that packets coming from same path have same identifier.
- Hence, if packet has same identifier as DDoS packet previously spotted, it is highly likely that this packet is also DDoS-oriented.
- Detect possible DDoS packets based on packet identifiers.
- More sophisticated attacker can easily circumvent such detection mechanisms by changing identifiers of packets using tools such as hping3.
- Negative selection algorithm to quickly figure out whether IP address of packet is legitimate based on eigenvalue sets to resist this type of DDoS.
- Requires server to maintain list of legitimate IP addresses, which implies that if client changes its IP address, it has to report change to server, making whole process less efficient.

Solutions against Flooding-Based Attacks 4/5

- Statistics-based approaches mainly detect DDoS attacks based on advent of clusters of DDoS traffics.
- Advantage of such methods lies in that they do not require per-packet information such as packet identifier and IP/MAC addresses for attack detection.
- Existing statistics-based detection solutions employ either packet entropy or machine learning tools.
- Researchers developed various entropy-based mechanisms to detect possible DDoS traffics.
- These methods not only more or less require manual efforts, which may face great challenges if DDoS traffics are encrypted, but also need distribution of large amount of traffics to achieve accurate detection.
- To automate detection, researchers switched to seek possible solutions from machine learning and deep learning techniques.

Solutions against Flooding-Based Attacks 5/5

- Basic machine learning methods such as J48, naive Bayes, and Bayesian network classifiers to detect botnet DDoS.
- Deep learning model using auto-encoder to detect encrypted DDoS traffics.
- Neural networks to identify DDoS attacks in software-defined networks.
- Although learning-based mechanisms require little human effort, they are susceptible to overfitting, meaning that they may perform differently on different types of DDoS attacks.
- To sum up, one can see that statistics-based detection needs large amount of DDoS traffics for entropy computation or training purpose, which implies that such detection mechanisms can only begin to function after fair amount of DDoS traffics already damage edge servers.

Solutions against Zero-Day Attacks 1/3

- To defend against this type of attacks, researchers developed mechanisms such as pointer taintedness detection and ECC-memory to spot possible memory leaks in program.
- Such methods require presence of original source codes, which are usually unavailable for edge devices.
- Approaches to perform memory analysis based solely on firmware.
- With help of deep learning models, e.g., recurrent neural networks (RNN), graph neural networks (GNN), and deep NLP, one can identify vulnerabilities in firmware with higher accuracy rates.

Solutions against Zero-Day Attacks 2/3

- Nevertheless, firmware is usually not available nowadays due to encryption and anti-debug fuses.
- On other hand, discovering and fixing memory corruptions when firmware is unavailable is non- trivial.
- Fuzzing mechanisms from IoT-app side to identify possible memory corruptions without need of firmware.
- However, this solution requires constant human interactions, making it unscalable and infeasible in some scenarios.
- Besides mechanisms associated with vulnerability discoveries, researchers also sought ways to actively protect edge devices from zero-day attacks.

Solutions against Zero-Day Attacks 3/3

- In-process memory isolation extension module to binary to defend against possible memory corruption attacks.
- Nevertheless, their method was designed specifically for x86 platforms and might consume extra computing resources, making it less feasible to be adopted in most resource-constrained IoT devices.
- IoT firewall using SDN to reduce attack surface of exposed IoT device.
- Lightweight isolation mechanisms on access routers which serve as guards before IoT botnet virus can access real edge devices.
- Note that even though these mechanisms can provide active protections, they cannot fix zero-day vulnerabilities, and hence may be ineffective if zero-day vulnerability is complex to trigger or has not be discovered before.

Attacks Exploiting Communication Channels 1/6

- In edge computing, exploiting communication signals has high potential to reveal sensitive information of victim due to rich channel information.
- In this case, attacker can be any curious malicious node, which does not have to be edge device or edge server, who continuously sniffs network traces and wishes to extract sensitive information out of them.
- Taxonomize this type of attacks into two sub-classes: those exploiting packet streams and those exploiting wave signals.
- Packet is atomic unit in most communication channels.
- Sequence of packets contain rich information, and hence, are widely exploited by attackers to infer sensitive data.

Attacks Exploiting Communication Channels 2/6

- Difference coding scheme employed by H.264 and MPEG-4 to reduce temporal redundancy in adjacent video frames can cause severe privacy leak in home surveillance even if video stream is encrypted.
- They found that leveraging simple machine learning algorithms such as k-NN and DBSCAN-based Clustering can achieve accuracy as high as 95.8% for inferring four standard human daily activities (dressing, styling hair, moving, and eating) defined by HIPAA.
- Inference model that takes Long Short-Term Memory (LSTM) deep learning network as attack vector to detect existence of wireless cameras and to infer user presence around target house.

Attacks Exploiting Communication Channels 3/6

- They successfully achieved prediction accuracy of 97.2% as result.
- Researchers also demonstrated that inference attacks can be launched by exploiting IoT traffic streams.
- Monitoring encrypted IoT traffics was developed three-step attack by first separating traffic into individual device flows using IP addresses of edge servers, then correlating each flow with its responsible IoT device according to unique identifiers, and finally inferring user activities from traffic rate changes.
- Due to subtle timing channel vulnerability introduced by most wireless routers which respond to different TCP packets with different timing gaps, attacker can easily infer correct TCP packet number and conduct off-path TCP packet injection attacks.

Attacks Exploiting Communication Channels 4/6

- Wave signals are another type of side channels existing in communication process and may have potential to reveal victim's sensitive information.
- One of notable examples is electromagnetic interference (EMI).
- Enev et al. carried out attack to infer video content playing in modern TVs through discernible EMI signatures.
- Selvaraj et al. showed that with help of intentional EMI, attacker can manipulate input and output signals of IoT sensory device from physical layer and bypass traditional integrity checking mechanisms.
- Besides EMI, researchers also found that Wi-Fi waves can be used as side channels to conduct inference attacks.

Attacks Exploiting Communication Channels 5/6

- Li et al. proposed malware-less side channel attack by exploiting channel state information (CSI) to infer victim's sensitive password input such as Alipay code based on finger movements.
- Existing studies also showed that human brain wave data contains rich information which can be used to conduct inference attacks.
- Group of researchers from University of Oxford studied privacy threat from brain-computer interfaces (BCI) and showed that if attacker can successfully capture raw electroencephalography data (EEG, i.e., human brain wave data), combined with machine learning algorithms such as boosted logistic regression, Stepwise Linear Discriminant Analysis (SWLDA), and Fishers Linear Discriminant Analysis (FLDA), attacker can infer victim's banking information, month of birth, face, and geographic location with accuracies of 15% to 40% better than random guessing attack.

Attacks Exploiting Communication Channels 6/6

- Xiao et al. demonstrated that attacker can still infer user's fine-grained activities even with reduced-featured EEG data.

Attacks Exploiting Power Consumption 1/5

- Power consumption is indicator of electric usage of system.
- It carries information related to either device that consumes energy as different devices have different power consumption profiles when operating, or intensity of computations in computing task.
- Hence, it attracts attention from number of researchers to investigate its link to sensitive data.
- Categorize this type of attacks into two sub-classes: attacks exploiting power consumption collected by meters and those exploiting power consumption collected by oscilloscopes.
- Smart meters can accurately measure electric power consumption of household.

Attacks Exploiting Power Consumption 2/5

- Therefore, such data can be exploited to infer sensitive household activities.
- In as early as 1992, Hart et al. proposed side channel inference method named non-intrusive appliance load monitoring (NILM) to monitor simple device states, e.g., on or off, based on energy consumption of individual appliances.
- This inference was benign as it was not employed for malicious attacks.
- Later, Stankovic et al. revised original NILM to conduct inference attack and showed that most household activities such as cooking, washing, laundering, watching TV, Gaming, and so on, can all be inferred from energy data available in smart meter infrastructure.

Attacks Exploiting Power Consumption 3/5

- Clark et al. carried out side channel attack leveraging power outlet of edge device, and managed to infer webpage that device was visiting with approximately 99% of accuracy.
- Later, they extended work to show that using power energy as inference feature can even detect malicious malware in edge device with accuracy of approximately 94%.
- Even though this work is not attack-oriented, it indirectly shows that sensitive behaviours of benign software can also possibly be identified based solely on power consumption.
- Recently, researchers found new physical channel, thermal side channel caused by power consumption, and exploited it to more effectively conduct time power attacks, and hence compromising edge data center availability.

Attacks Exploiting Power Consumption 4/5

- Oscilloscope is instrument measuring electronic information (e.g., voltage and current) of hardware device.
- In modern embedded devices, some chip can perform complicated cryptographic algorithms such as AES-CCM, with hardcoded secret key in chip.
- Such secret key cannot be directly cracked if no software-level vulnerabilities are present.
- However, researchers found that power consumption of hardware may be susceptible to leaking key.
- Ors et al. demonstrated that adopting simple power analysis and differential power analysis can reveal significant amount of information carried by elliptic curve cryptosystem on FPGA chip.

Attacks Exploiting Power Consumption 5/5

- Ronen et al. showed that adopting correlation power analysis can completely reverse AES-CCM master key used to encrypt/decrypt firmware installed in Philips hue smart lights such that they can deliberately create any malicious firmware and install on any Philips hue smart light over-the-air.
- What is even worse is that nearly all cryptographic approaches and their corresponding hardware are vulnerable to power analysis attacks.
- Yet, launching power analysis attacks require attacker to be able to physically access target device or through malicious apps, making this class of attacks difficult to implement in practice.

Attacks Exploiting Smartphone-Based Channels 1/5

- Smart- phones are key edge devices in many applications.
- Different from IoT devices, smartphones have more advanced OSes and possess richer system information.
- Therefore, compared with more dumb IoT devices, smartphones can be exposed to broader attack surface.
- Categorize attacks into two sub-classes: attacks exploiting /proc filesystem and those exploiting smartphone embedded sensors.
- /proc is system-level filesystem created by kernel in Linux.
- It contains system information such as interrupt and network data.
- Even though it is system-level filesystem, it is readable by user-level threads and applications.

Attacks Exploiting Smartphone-Based Channels 2/5

- Hence, accessing /proc filesystem does not require any additional permission.
- As result, /proc has been widely employed to perform side channel attacks.
- Chen et al. proposed UI state inference attack via which attacker can carry out UI phishing to trick victims to make unwanted requests to edge servers by using memory data that is publicly available in /proc.
- Diao et al. exploited interrupt information stored in /proc/interrupts to infer sensitive information of smartphone such as pattern lock and foreground running UI.

Attacks Exploiting Smartphone-Based Channels 3/5

- Zhou *et al.* made use of various side channels such as tcp_snd, tcp_rcv, and BSSID available from Android device's /proc to infer user's sensitive information including health condition, location, and social network identity.
- Xiao *et al.* further strengthened this work by developing approach to correlate victim's social network identity with smartphone device used to access social network in more accurate and practical manner.
- Yet, as matter of fact, accessing /proc requires attacker to trick victim to install malicious app, which is bottleneck of such attacks.
- Nowadays, smartphone is integrated with variety of embedded sensors for handling various tasks.

Attacks Exploiting Smartphone-Based Channels 4/5

- On one hand, these sensors can greatly elevate functionalities of smartphone; on other hand, they impose security concerns of leaking sensitive information.
- Asonov *et al.* and Zhuang et al. independently showed that it is feasible to infer user's keystrokes by analysing acoustic sounds emitted from physical keyboards which existed in early stage smartphones.
- However, most current smartphones eliminate physical keyboards by employing touchscreens.
- Nevertheless, attackers make great effort to keep pace with this development.
- Zhou *et al.* cracked pattern lock of smartphone by leveraging acoustic signals reflected by fingertip captured through microphones.

Attacks Exploiting Smartphone-Based Channels 5/5

- Cui et al. showed that tap keystrokes can be inferred using smartphone accelerometer and gyroscope sensors.
- Recently, Chen et al. proposed novel side channel attack leveraging victim's eye movements from video secretly recorded by smartphone camera, to infer victim's keystrokes on mobile device.
- One can see that embedded sensors within smartphone carry abundant information that can be exploited to perform inference attacks.
- Based on above summary, categorize side channels that can be exploited by attackers into two classes: controllable ones that include packet streams as well as smartphone-based /proc filesystem and embedded sensors, to which access can be restricted, and uncontrollable ones that include wave signals and power consumption, which exist unconditionally due to innate nature and are not modifiable.

Data Perturbation 1/

- Most well-known perturbation algorithm to protect sensitive data from inference attacks is k-anonymity, which modifies identifier information of piece of data before publishing its sensitive attributes, making it indistinguishable from another $k - 1$ pieces of data, with these k pieces of data forming equivalence class.
- Machanavajjhala *et al.* found that k-anonymity suffers from homogeneity attack when values of sensitive attribute within equivalence class are identical.
- To overcome this issue, they proposed l-diversity by ensuring each equivalence class to have at least l distinct values for each sensitive attribute.
- However, Li *et al.* pointed out that l-diversity has two main limitations, i.e., it may be difficult and unnecessary to achieve, and it is insufficient to prevent attribute disclosure when distribution of certain value differs significantly from those of others for same sensitive attribute.

Data Perturbation 2/

- Therefore, they proposed t-closeness to overcome these two limitations by requiring difference between distribution of sensitive attribute value in class and that in whole database is less than threshold. Nevertheless, researchers noticed that Earth Movers Distance metric used in t-closeness may not be able to convincingly measure “closeness” among values.
- On other hand, even though k-anonymity and its successors provide reasonable privacy protection, they do not have sound theoretical foundations to support their privacy preservation capacity.
- Breakthrough was made by Dwork et al. in 2008, who presented concept of ε-differential privacy, which formally defines privacy preservation with solid theoretical proof.

Data Perturbation 2/

- Data randomization algorithm A is said to provide ϵ-differential privacy if

$$\Pr[A(D_1) \in S] \leq e^{\epsilon} \times \Pr[A(D_2) \in S]$$

- where D_1 and D_2 are any two databases that differ by single entry, S is collection of results of A, and ϵ is positive real number, namely, privacy budget.
- In same work, Dwork et al. proposed Laplace mechanism to achieve ϵ-differential privacy for numerical values by adding noises generated from Laplace distribution.
- To deal with entity objects, Talwar et al. presented exponential mechanism to achieve differential privacy.
- Since differential privacy is only privacy norm with strict mathematical proofs, it has been widely deployed to defend against side channel information leaks.

Data Perturbation 3/

- In 2009, McSherry et al. implemented Privacy Integrated Queries (PINQ) platform which provides differentially private data analysis through SQL-like programming languages.
- Later, Roy et al. built Airavat, differentially private platform for data computation over edge servers.
- Xiao et al. injected differentially private noises into procfs to prevent side channel attacks on data storage.
- Cheu *et al.* proposed distributed differential privacy model via mixnet, which fits decentralization feature of edge computing.
- Yet, problem of differential privacy is that it may have limited protection effect if data are correlated.
- More generally speaking, differential privacy may not be applicable when data has high global sensitivity, which requires strong noises to perturb data for privacy guarantees, and thus sacrificing utility of data.

Restricting Accesses to Side Channels

- As mentioned earlier, alternative approach to defend against side channel attacks is to restrict accesses to side channel information.
- Side channel obfuscation on source code level is such scheme to defend from software surface.
- Molnar et al. proposed mechanism to eliminate control-flow side channel attacks from source code.
- Zhang et al. developed side channel detection scheme to monitor abnormal cache behaviours on cloud and edge servers.
- These two methods directly perturb side channels to obstruct accuracy of inference algorithms leveraged by side channel attacks. In recent years, as development of TrustZone technology advances quickly, researchers invented mitigation solutions using TrustZone-empowered hardware, SGX, to prevent side channel attacks.
- This method mainly disallows unauthorized accesses to side channels protected in TrustZone.

Server-Side Injections 1/5

- There are mainly four types of injection attacks targeting edge servers, namely SQL injection, XSS, CSRF and SSRF, and XML signature wrapping.
- SQL injection is code injection technique that destroys backend databases.
- To construct normal SQL query, legitimate user is allowed to manipulate only designated areas (e.g., name and date) to get results from server.
- However, attacker may manage to circumvent this constraint by inputting escape characters (such as quotation marks) along with query string.
- In this case, server may mistakenly execute everything attacker inputs after escape characters.
- This vulnerability usually exists when database management system does not filter escape characters for SQL processing.

Server-Side Injections 2/5

- SQL injection not only is serious threat to data confidentiality and integrity, but also allows attackers to inject malicious scripts
  - e.g., using SELECT ... INTO OUTFILE command.
- Hence, SQL injection is one of main methods for malware injection.
- Cross-Site Scripting (XSS) is client-side attack in which attacker injects malicious codes (usually HTML/JavaScript codes) into data content, which can be accessed and executed automatically by servers.
- Note that modifier “client- side” here does not refer to edge device side, but rather edge server side in which edge server works as “client” to visit or access services provided by other edge servers or cloud server.
- Therefore, contrary to traditional general-purpose computing systems, XSS is type of injection attacks that happen at edge server level in edge computing.
- The attack is caused by fact that edge servers do not filter code from data content.

Server-Side Injections 3/5

- Even though XSS is not novel attack and its mechanism has been well studied, it is still serious threat to edge computing infrastructures.
- XSS vulnerability was reported in Cisco Edge Director framework which can lead to arbitrary code executions.
- Martin et al. created automatic model checker based on Goal-Directed Model Checking to find XSS and SQL vulnerabilities in large amount of codes.
- Cross-Site Request Forgery (CSRF) is attack in which end user (i.e., edge server in this case) is forced to execute unwanted actions through web applications.
- Server Side Request Forgery (SSRF) is attack in which edge servers are abused to read or alter internal resources.
- Root cause of both attacks is coarse-grained design of verification mechanism, such as weak identity verification method that can be easily broken.

Server-Side Injections 4/5

- By exploiting coarse-grained verification, attacker can spoof as "*legitimate*" edge server to send command to other edge servers without being discovered, making other edge servers to be subject to CSRF and SSRF attacks.
- Typically, CSRF and SSRF mainly target traditional Internet infrastructures.
- It was found in 2016 that edge systems are also susceptible to these two attacks.
- XML signature wrapping happens when edge computing infrastructure acquires Simple Object Access Protocol (SOAP) as its communication protocol, which transmits messages using Extensible Markup Language (XML) format.
- In this attack, malicious attacker first intercepts legitimate XML message, creates new tag, and places copy of original message (which may contain verification parameters such as tokens) within new tag (also known as wrapper) to form "giant" tag-value pair.

Server-Side Injections 5/5

- Next, attacker replaces original values with malicious codes in original message, and combines modified original message with "*giant*" tag-value pair by placing new pair before regular tag-value pairs of original message.
- Upon receiving this tampered message, victim edge server would first verify message, which could succeed since attacker did not delete original values (containing still-valid verification information) but rather puts them into new tag (wrapper).
- Once verification succeeds, server would execute malicious code injected by attacker.

Device-Side Injections 1/4

- Various diverse methods for injecting malware into IoT devices exist since IoT devices are highly heterogeneous on both hardware and firmware.
- Most common approach to remotely inject malware is to exploit zero-day vulnerabilities that can lead to remote code execution (RCE) or command injection. One of most infamous examples is “IoT Reaper” virus captured in 2017, which infects millions of IoT devices through Internet protocol and WiFi by exploiting at least 30 RCE vulnerabilities existing in 9 different IoT devices ranging from network router to IP camera.
- In academia, Cui et al. discovered that HP-RFU (Remote Firmware Update) protocol adopted by LaserJet printers allows attacker to modify any pre-deployed firmware of printer due to lack of signature verification check.
- Hernandez *et al*. found that Smart Nest Thermostat lacks proper protection for firmware update, allowing attacker to update arbitrary firmware using USB connection.

Device-Side Injections 2/4

- Maskiewicz *et al.* pointed out that firmware update mechanism used by Logitech G600 mouse is buggy and allows attacker to infect firmware through networking or USB.
- Recently, Ronen *et al.* implemented attack to remotely and contactlessly inject malicious firmware into IoT devices using Zigbee Light Link protocol.
- Note that attacks mentioned above are typically referred to as firmware modification attacks.
- Injecting malware that has cross-accessing capability into mobile devices is not trivial since major mobile OSes such as iOS and Android adopt app-isolation mechanism, i.e., sandbox mechanism, to ensure that every app is isolated virtually on memory and no app can access other apps' resources and contents unless permitted from kernel level.
- Wang *et al.* conducted early study to summarize how legitimate API calls, e.g., intent and scheme which are open to all mobile developers, can possibly allow attacker to inject malicious contents into other third-party benign apps.

Device-Side Injections 3/4

- Ren *et al.* exploited OS-level structure called Android Task Structure (ATM) to passively inject malicious UIs to benign Apps.
- Xiao et al. improved Ren's work by exploring active attacks exploiting ATS, making ATS-based injection attack more feasible and powerful in practice.
- Even though these attacks sound powerful, they may not cause significant damages to edge computing infrastructure as they
- directly exploit Android official APIs and structures.
- To carry out more dangerous attacks, attackers tend to use third- party malicious libraries which are not only more powerful but also less likely to be detected.
- Chen *et al.* found that 6.84% of official Android Apps from Google Play Store and 2.94% of iOS Apps from Apple App Store use harmful libraries such as PhaLibs which opens backdoor for code injection.

Device-Side Injections 4/4

- Note that all attack mechanisms mentioned above require victim to install attacker's malicious app to begin with, since they all require assistance from native Android OS at some level.
- This limitation may diminish practicality for launching such attacks in real world as victim has to be tricked to install malicious app in first place.
- To overcome this limitation, *Li et al.* made breakthrough by discovering serious design vulnerability in Android WebView, which allows attacker to remotely inject malicious apps into legitimate Android device through malicious website.
- This method can achieve most of attack effects (e.g., stealing other app's resources and UI hijacking) app-required attack can have without need of installing malicious app into victim's device - it only requires victim to open webpage using its smartphone.

Defenses Against Server-Side Injections 1/6

- Defenses against server side injections also consider four major types of attacks: SQL injection, XSS, CSRF/SSRF, and XML signature wrapping.
- Since SQL injection attacks were discovered at time when SQL databases were invented, defense and detection mechanisms have been studied for fairly long time.
- Halfond *et al.* categorized early research into detection-focused and prevention-focused.
- Detection-focused techniques basically employ code checking with various schemes such as static analysis, dynamic debugging, blackbox testing, and taint-based analysis; while prevention-focused techniques target to prevent any illegal SQL queries from being executed by means such as setting up proxy filter and employing instruction set randomization.

Defenses Against Server-Side Injections 2/6

- In same work, Halfond *et al.* also pointed out that most of early defense mechanisms were not mature, and that only two of them may have potential to resolve practical SQL injection attacks.
- First one is lattice-based static analysis framework proposed by Huang *et al.*, which not only achieves defense against SQL injection on full surface but also manages to run in real-time.
- Yet, major limitation of this mechanism is that it only protects servers built on PHP, while many servers are built on other back-end languages such as Java and C#.
- Second mechanism was designed by Livshits et al., which is specific to Java programs while SQL injections can happen in other backend languages such as PHP and C#.
- Recently, researchers proposed improved mechanisms by comparing input queries with programmer intended queries to check inconsistencies for possible SQL injection identification, or even simply remove SQL query attribute values for further analysis before query can be executed.

Defenses Against Server-Side Injections 3/6

- Yet, these mechanisms require fair amount of manual effort to begin with.
- Therefore researchers started seek possible solutions from machine learning or deep learning.
- For examples, Jackson *et al.* located SQL injection vulnerabilities in program using natural language processing (NLP); and Ross et al. evaluated multiple machine learning techniques and showed that they can achieve high accuracy for detecting SQL injection attacks.
- Similar to SQL injection, defense mechanisms against XSS have been studied for long time.
- Based on survey conducted by Gupta *et al.* early studies focused on 10 types of defense schemes, including manually implementing hardcoded rules at client side to inhibit XSS malicious codes from being executed and adopting instruction set randomization (ISR) to turn malicious codes into harmless ones.
- Gupta *et al.* also mentioned that these early research cannot completely resolve all types of XSS attacks.

Defenses Against Server-Side Injections 4/6

- Recent research improved early studies by adding context-aware sanitization into XSS detection to make it more robust with reduced false positive rates.
- Rathore *et al.* also leveraged learning techniques to detect XSS vulnerabilities and achieved over 97% accuracy.
- Compared to SQL injection and XSS, CSRF and SSRF have much shorter history.
- Number of defense mechanisms targeting CSRF is still limited.
- Jovanovic *et al.* proposed defense approach based on secret token while Johnson presented CSRF defense scheme based on referrer header checking.
- Later, Barth et al. showed that these two mechanisms can fail if being adapted to new variations of CSRF, e.g., login CSRF; they proposed modified version of referrer header method by forcing client side to send origin header to defend against login CSRF.

Defenses Against Server-Side Injections 5/6

- Yet, this mechanism requires effort from client side, which may increase computational burden of edge devices.
- Czeskis overcame this limitation by offloading detection task to server side to create lightweight protection platform.
- Defense against SSRF is even more limited.
- Fung *et al.* proposed privacy-preserving defense mechanism against SSRF by embedding clients' credentials in requests.
- Srokosz *et al.* revised static WAF approach making it able to defend against SSRF.
- XML signature wrapping became popular when SOAP protocol was introduced in industrial world; hence, it has shortest history among all server-side malware injection attacks.
- Compared to other server-side injection attacks, wrapping attacks are not difficult to defend against due to their limited attack surface and simple attack forms.

Defenses Against Server-Side Injections 6/6

- Jensen *et al.* proposed schema hardening approach on basis of W3C XML Schema for countering wrapping attack.
- Gupta *et al.* developed side-channel based detection mechanism by counting frequency of each node in requested service to spot suspicious wrapping attack.
- Kumar *et al.* developed detection method by introducing positional tokens.

Defenses Against Device-Side Injections 1/6

- For injection attacks targeting IoT devices main threat comes from firmware modification attacks.
- Currently, limited research exists to investigate corresponding defense mechanisms.
- Cui *et al.* was first to propose defense mechanisms to mitigate firmware modification attacks.
- Inspired by idea of Address Space Layout Randomization (ASLR) and Instruction-Set Randomization (ISR), they proposed Autotomic Binary Structure Randomization (ABSR), which takes arbitrary executables or firmware as input and outputs variant of original with reduction of unused codes to minimize attack surface.
- They also proposed software symbiotic method which injects intrusion detection functionality into binary firmware of existing IoT devices to inhibit malicious modifications.

Defenses Against Device-Side Injections 2/6

- Even though ideas of these two mechanisms are reasonable, their realizations are never easy, and Cui *et al.* did not present any specific methodology on implementation of these two mechanisms.
- Lee *et al.* proposed blockchain- based cryptographic approach to securely update firmware for IoT devices.
- However, this design adopted PoW algorithm which is not only computationally prohibitive both on hardware and on time in IoT but also lacks ability to detect malicious codes, making this method impractical in defense.
- Moreover, it is unconvincing that this blockchain based mechanism can fit edge computing infrastructure.
- Weiser *et al.* proposed to isolate sensitive data and codes from other non-sensitive ones using memory protection unit (MPU) to inhibit firmware modification attacks under RISC-V.

Defenses Against Device-Side Injections 3/6

- However, this approach requires that microcontroller (MCU) of IoT device to be equipped with MPU.
- Moreover, it only supports RISC-V instruction set while ARM and MIPS are more popular - in fact, ARM has approximately 95% of IoT market share.
- In summary, defending against malicious firmware modification attacks remains largely under-explored at end device.
- Malware injection attacks targeting mobile devices exploit design flaws of mobile OSes as well as usage of malicious libraries.
- Schmerl *et al.* and Wu *et al.* independently proposed static analysis methods to identify possible malicious uses of dangerous Android APIs.
- Backes *et al.* developed library detection technique which can resist common code obfuscations and detect security vulnerabilities and malicious behaviours in Android libraries.

Defenses Against Device-Side Injections 4/6

- Xiao *et al.* proposed Task Interference Checker, TICK, to eliminate malware injection by making use of Android Task Structure.
- Unfortunately, currently no practical solution exists that can counter remote malware injection attack exploiting WebView unless implementing protective schemes at Android kernel or re-designing Android OS.