ChatGPT

Network-Level Intrusive Logging and Push Notification Manipulation: Technical & Operational Analysis

Introduction

Network-level **intrusive logging** and **push notification manipulation** refer to sophisticated attack techniques where adversaries intercept, monitor, or alter data in transit between users and services. Such operations often involve **man-in-the-middle (MitM) attacks** to eavesdrop or inject content, and can be conducted by state actors, contractors, or rogue insiders. This report provides a technical deep dive into these methods, how attackers gain access to critical messaging infrastructure, tactics for modifying content on the fly, ways to forensically detect such intrusions, and practical defenses for users. Real-world cases are included to illustrate the capabilities and risks.

Man-in-the-Middle Attack Techniques and Network Interception

Attackers use MitM techniques at the network level to **intercept and log communications or to modify data**. Key methods include:

- **DNS Spoofing (DNS Cache Poisoning):** The attacker corrupts the Domain Name System resolution process so that domain lookups return a malicious IP address. This effectively misdirects traffic to a server the attacker controls, enabling them to impersonate legitimate sites or proxy the connection. For example, a malicious actor can intercept a DNS request and return an IP to their own server instead of the real server, then relay traffic to the real site while silently eavesdropping [1]. This can redirect victims to bogus websites or allow the attacker to **steal information** by relaying (MitM) the traffic. DNS spoofing typically requires control over the target's network (e.g. compromised router or ISP). Modern defenses like DNSSEC and DoH/DOT, as well as TLS (HTTPS), can reveal or prevent spoofing – if the attacker tries to spoof an HTTPS site without the proper certificate, the user's browser will display a certificate warning indicating a possible MitM [2].

- **TLS/SSL Interception:** Even if DNS is correct, an attacker may attempt to intercept encrypted HTTPS traffic. In **TLS interception**, a proxy controlled by the attacker sits between the client and server, presenting a fake certificate to the client to fool it into trusting the proxy, while the proxy establishes its own TLS session with the real server. This is usually done by installing a malicious or rogue **certificate authority** certificate on the victim's device (or abusing a compromised/trusted CA). For instance, there have been cases where certificate authorities (CAs) issued subordinate CA certificates to companies or agencies, which allowed those entities to generate valid certificates for any domain and silently decrypt traffic [3]. In one incident, a major CA (Trustwave) admitted it issued a subordinate certificate for an enterprise's data loss prevention device, effectively enabling MITM interception of all SSL/TLS traffic within that organization [3]. If an attacker obtains such a capability (through CA compromise or malware-installed root certificates), they can decrypt and read or modify supposedly secure traffic. Modern systems employ countermeasures like **certificate pinning** (embedding the legitimate server certificate

or CA in the app) to thwart unauthorized certificates. For example, Apple's push notification daemon (apsd) implemented certificate pinning for its servers, only trusting a specific Apple CA and refusing others – an attempted MITM with a self-signed cert will fail the trust check and be logged as an "untrusted peer" [4] [5].

- **Proxy Servers and Traffic Injection:** Attackers may also use **malicious proxies or in-path appliances** to intercept and inject content. At the local network level, techniques like ARP cache poisoning can funnel a victim's traffic through an attacker's machine (e.g. on public Wi-Fi) [6] [7]. On a larger scale, state-level actors have deployed **network injection devices on ISPs/backbones**. These devices can **monitor streams and inject malicious payloads or forged responses** on the fly. A notorious example is China's *Great Cannon*, which is a tool co-located with the Great Firewall. The Great Cannon can hijack traffic to or from specific IP addresses and *arbitrarily replace unencrypted content* as it passes through the network [8]. Unlike a passive firewall, this offensive system intercepts targeted traffic and injects data (e.g. malicious JavaScript) into the connection. In 2015, the Great Cannon was used to inject code into HTTP traffic from Baidu, redirecting foreign browsers to flood GitHub with a DDoS attack [9] [8]. Similarly, Western spyware contractors have provided **FinFisher/FinFly ISP** packages that enable government clients to infect targets via ISP-level injection. In Turkey and Egypt, *Sandvine PacketLogic* devices (deployed at ISPs) were configured to **precisely target certain users**: when a target attempted to download legitimate software (like an app or update), the device would inject an HTTP redirect to a malicious server, resulting in the user downloading a trojanized version of the software [10]. This covert redirection/injection tactic is extremely stealthy – the download URL changes in transit without the user's knowledge, delivering spyware while still installing the expected app. Such attacks illustrate how an attacker in control of the network path can *both surveil (log)* user activity and **manipulate the data** being transferred.

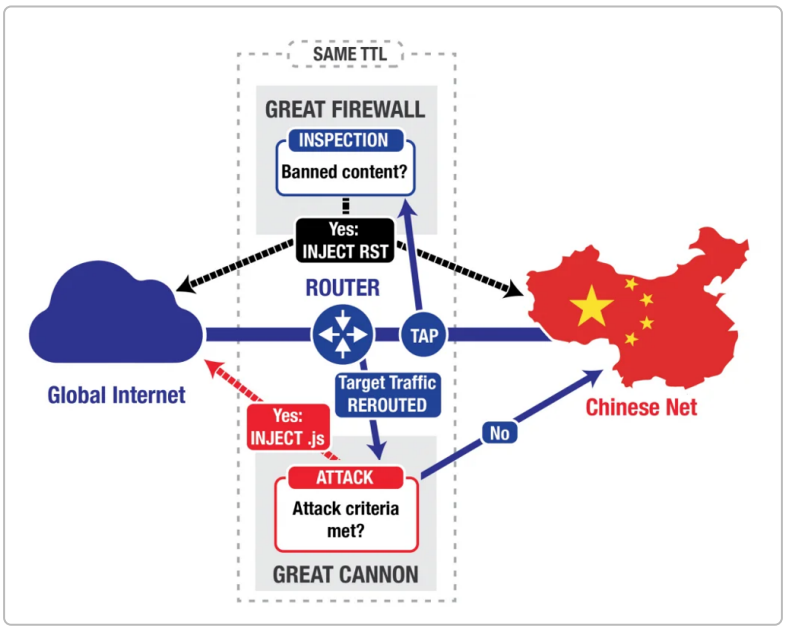

Figure: Simplified topology of a state-level on-path interception system (Great Firewall) versus an in-path attack tool (Great Cannon). The Great Firewall monitors traffic and can inject forged TCP resets when it sees "banned content" to cut connections, whereas the Great Cannon (if attack criteria are met) can reroute or inject malicious payloads (e.g. script) into traffic destined for specific targets [11] [12]. Both systems operate at the network backbone level.

State actors and advanced threat groups may chain these techniques – for example, performing DNS hijacking to route a target's traffic through a controlled proxy, then using a forged certificate to intercept TLS and modify content. In all cases, **intrusive logging** is a by-product: the MitM position allows the actor to log URLs, messages, or any unencrypted data. Intelligence agencies have been known to place fiber-optic taps and network sensors (with cooperation of telecoms or clandestine access) to record vast amounts of internet traffic for analysis. In summary, MitM attackers leverage weaknesses in network trust (DNS and PKI) and control of infrastructure to **surveil and tamper** with data in transit.

Gaining Access to Push Notification Infrastructure (FCM and APNs)

Push notification services like Google's Firebase Cloud Messaging (FCM) for Android and Apple Push Notification service (APNs) for iOS are critical infrastructure for delivering messages from app servers to devices. These services are designed to be secure and are operated by the platform vendors, but determined actors may find ways to exploit or leverage them:

- **Insider Access and Legal Requests:** One straightforward path is through the platform providers themselves. Governments can compel companies like Apple and Google to hand over data related to push notifications. In late 2023, it was revealed that both foreign and U.S. agencies had been requesting push notification records from Apple and Google [13]. Because nearly all mobile app notifications transit via Apple/Google servers, those companies have a *"unique insight"* into which apps are sending notifications, when, and to which user device [14]. By obtaining these logs, an actor can infer if a particular user has a certain app installed or is active on it (for example, correlating a device ID with usage of an encrypted messaging app via its notification timing) [13]. The data handed over is typically *metadata* – e.g. which app's server contacted the push service, timestamps, and the destination user's device token or account identifier – but in some cases **unencrypted notification content** might also be revealed [15]. (Many apps send only a generic alert or use end-to-end encryption for content, but poorly designed ones might include sensitive text in the push payload.) Such data access does not require "hacking" the infrastructure; it is obtained via legal process or secret agreement, effectively making Apple/Google an unwilling participant in surveillance. In response to public disclosure, Apple updated its policies to require a higher legal standard (court order) before releasing push notification metadata and now reports these requests in transparency reports [15] [16].

- **Compromise or Imitation of Push Services:** Another vector is technical subversion of the push services. While neither FCM nor APNs are known to have been *breached* at the server level (they are heavily secured), attackers have explored ways to *impersonate or piggyback* on these systems. For instance, Apple's APNs uses mutual authentication (the device trusts Apple's servers and presents a device certificate). A rogue actor who can perform DNS spoofing or has a fraudulent Apple certificate might try to pose as the APNs server to a victim device. In practice, this is extremely difficult because of TLS and Apple's certificate pinning for APNs (as seen, attempts to fake APNs were rejected as untrusted [4]). However, **exploit frameworks** have found creative ways in. The NSO Group's infamous Pegasus spyware (a private contractor tool used by governments) managed to exploit Apple's push ecosystem via zero-click iMessage vulnerabilities. Citizen Lab reported that Pegasus had a method to *send a malicious payload through Apple's push channel* by impersonating an app service. In a leaked description, a Pegasus exploit could **forge an application's push message** in a way that APNs accepted and delivered it, thereby injecting spyware onto the device without user interaction [17]. In simpler terms, the spyware crafted a

push notification that looked legitimate (coming from an app the target had installed), exploiting an APNs weakness to drop malware onto the phone [17]. This is an example of a *rogue agent abusing the push infrastructure* by finding a protocol vulnerability rather than actually breaking Apple's servers. Similarly, a vulnerability in FCM or improper app server security could be abused – if an attacker obtains the FCM server key for an app (via a breach of the app developer's systems), they could send unauthorized push messages to all users of that app. There have been cases of attackers compromising news or content apps' backends and sending out rogue push notifications (e.g. false news alerts) to sow confusion, essentially hijacking the trust in that channel.

- **Mobile Carrier or Network Access:** Some push notification data might be accessible by actors with lower-level network control. For example, FCM messages and APNs notifications travel over TLS-encrypted channels (to Google's and Apple's servers respectively). A nation-state that controls telecom infrastructure could attempt to intercept those channels, but without breaking TLS they would only see metadata (IP addresses, packet sizes). In theory, an ISP-level attacker could block or delay push traffic (causing notifications to fail), or perform downgrade attacks if any legacy protocols are in use. In practice, to **read or alter push contents** in transit, the attacker would need to compromise the TLS encryption (via a malicious root cert or a vulnerability in the protocol, as discussed in MitM techniques). There are no public reports of FCM or APNs traffic being routinely decrypted by ISPs, but governments have attempted nationwide TLS interception in other contexts (e.g. Kazakhstan's government mandating a root certificate). A *rogue employee* or contractor at Apple/Google could theoretically abuse internal access to push systems, but platform companies have strict controls and audit logs to deter this. The more common scenario is using push infrastructure as a **targeting mechanism**: e.g. an adversary might obtain a list of device tokens for a chat app and use the push service to enumerate active users, or request records to see who received a certain broadcast message. This kind of access turns the push network into a surveillance tool (mapping out a social network of who uses what app).

In summary, actors gain access to push notification infrastructure either through **official channels (legal demands or clandestine partnerships)** or via **technical exploits that impersonate or undermine the push delivery mechanisms**. Both FCM and APNs are high-value targets for surveillance because they sit at a choke point for millions of users' communications. Users generally cannot opt out of these services without losing functionality, which makes securing them and monitoring their use by third parties extremely important.

Dynamic Notification and Content Manipulation Tactics

An attacker with a man-in-the-middle position or access to the message pipeline can go beyond passive spying – they can **dynamically modify or inject content** into notifications and articles on the fly. This section examines how delivered content might be altered, including techniques for content routing, A/B narrative targeting, and symbolic triggers:

- **Content Routing and Injection:** Attackers often **route content through infrastructure they control** (such as proxy servers or malicious content delivery nodes) to modify it in transit. As described earlier, devices like the Great Cannon or FinFly ISP essentially *reroute targeted traffic* and then **inject** new content or alter the original. For example, in Turkey's spyware campaign, when a user requested a software download, the Sandvine/FinFisher system diverted that request to a malicious server which responded with a *tampered file*, all accomplished via on-the-fly HTTP redirection [10]. Similarly, an attacker could intercept an unencrypted news feed or API

response and change the text of an article or the link in a push notification. **Proxy injection** can also be selective: the attacker might only inject the malicious content for certain users or at certain times (to avoid detection), while letting others receive the legitimate content – a form of *content routing* where some traffic takes a different path (through the injector) and some goes direct. In one documented case, China's Great Firewall infrastructure was used to *inject a fake news story* on Qatar's state news agency website in 2017, misattributing quotes to Qatar's leader – this planted article contributed to a diplomatic crisis [18]. While that was done by hacking the site, a powerful MitM could achieve a similar result by altering the content as it is delivered to readers (for instance, swapping out the text or inserting disinformation into the data stream).

- **A/B Narrative Targeting:** Borrowing a term from marketing ("A/B testing"), adversaries can present **different content to different groups** in order to influence opinions or test reactions. At the network level, this means the injector uses some criteria – such as the user's IP range, geolocation, or an identifier – to decide which "version" of content to deliver. For instance, an oppressive regime could intercept a breaking news notification and modify its text for users inside the country (narrative A), while users outside get the unaltered news (narrative B). This was observed in concept with the Sandvine device, which could target **specific IP addresses** for delivering trojanized downloads, while others got the normal software [10]. An attacker could apply the same principle to information: one segment of the population receives a push notification with a false message or propaganda, while others receive a benign or different message. This allows fine-grained *disinformation campaigns*, potentially tailoring narratives to demographic groups. Security researchers warn that such capability could be used to conduct covert influence operations – effectively a personalized propaganda delivery via hijacked channels. A real-world hint of this was Russia's "Quantum Insert" operations in which the NSA/GCHQ delivered tailored exploit pages to individual targets by intercepting their web requests [19] [20]. While the goal there was malware, the technique could just as easily swap legitimate content with doctored information for certain users. **A/B targeting via MITM** might also involve *experimentation*: an actor could inject two variants of a misleading news story to see which incites more reaction (measured via subsequent traffic or social media posts), then push the more effective narrative broadly.

- **Symbolic Trigger Embedding:** Sophisticated attackers often embed **triggers or markers** in content to activate specific behaviors or to track the content's spread. In the context of intrusive logging and manipulation, a *symbolic trigger* could be a code word, pattern, or object in a delivered message that is recognized by either the receiving device or by analytic systems. For example, the Great Firewall of China monitors for certain "sensitive keywords" (like references to Falun Gong) in web traffic; if it detects them, it immediately injects forged TCP reset packets to terminate the connection [12]. Here, the banned keyword is a trigger that causes an automated injection response. Similarly, NSA's QUANTUM system used triggers such as unique web cookies or specific URLs: when their passive sensors sniffed a target's HTTP request with a matching identifier, it signaled the attack server to shoot back a malicious payload packet **before the real server's response arrives** [21] [22]. The trigger in that case could be as simple as "user requested linkedin.com/profile of target X," which cues the injector to send a fake response redirecting to an exploit site. In push notification manipulation, triggers might be **symbols in the content** that only certain devices respond to. For instance, a rogue firmware on a phone might watch notifications for a special keyword or character sequence; when it arrives (embedded in an otherwise normal-looking message), it could activate spyware or send a secret acknowledgement. Alternatively, an attacker could watermark different versions of an article or notification with distinct symbols (like a specific unicode character or whitespace pattern) as a way to **trace leaks** – if version "A" with symbol X appears on Twitter, they know which group saw that version. These symbolic triggers can override normal logic, somewhat analogously to how

psychological triggers might override reason [23] (e.g. using an emotionally charged word in a headline to provoke reaction). In the operational security realm, embedding triggers in content is a stealthy way to either command and control malicious implants or to gauge how information propagates through a population.

In summary, once an attacker can manipulate content delivery at the network level, they gain the ability to **shape reality for the target**: injecting false information, selectively spreading different stories (narratives), and using triggers to orchestrate more complex operations. These tactics have been theorized and in some cases implemented by advanced threat actors, blending the lines between cyber-attack and information warfare.

Forensic Detection of MITM Attacks and Notification Tampering

Detecting that such an intrusion or manipulation has occurred is challenging – skilled attackers aim to leave few traces. However, there are several **forensic strategies and indicators** that can reveal MITM presence or content tampering:

- **Network Traffic Anomalies:** One telltale sign of on-path attacks is anomalies in the packet flows. For example, in the NSA's QUANTUM INSERT attacks, researchers discovered that a victim's browser would receive two TCP response packets with the **same sequence number** – one from the legitimate server and one from the attacker's race-injected server [24] [25]. Under normal conditions, duplicate sequence numbers wouldn't occur; their presence was a giveaway of a "man-on-the-side" injection. Similarly, China's on-path Great Firewall injects reset packets without preventing the real packets from arriving, which creates a detectable pattern: the target might briefly see a connection reset even though some legitimate data was received, or a scan of the traffic shows both genuine and forged packets intermingled [26]. Intrusion detection systems (IDS) can be tuned to flag these conditions. Fox-IT researchers even released Snort/Suricata rules specifically to catch QUANTUM Insert by looking for these network anomalies [27] [24]. From a defender's perspective, **browser payload comparison** is a powerful technique: if you suspect content tampering, you can compare the data retrieved by a client on one network with the same request on a known clean network. For instance, fetch the same article or file via two different ISPs (or through an encrypted VPN vs. direct connection) – if one is altered or contains extra code, that's evidence of MITM manipulation. In practice, organizations like Citizen Lab do this kind of *differential analysis* to catch censorship or injection: any differences in payload indicate something in the middle is modifying content.

- **Certificate and TLS Validation Errors:** Because active attackers often struggle to perfectly impersonate legitimate servers (especially for TLS), their attempts may leave traces in logs. A user or investigator might find **errors in system logs or browser console** indicating certificate issues. For example, on a Mac, if someone attempted to spoof Apple's push server, the apsd daemon log showed messages like "Unrecognized leaf certificate" and SSL handshake failures [28] [4]. This means the client detected a certificate that isn't the one it expected (likely the attacker's). Many applications will simply fail to connect if the cert check fails, but they may still log the event. Security-conscious users can monitor their device logs (via tools like **Console.app on macOS/iOS** or `logcat` on Android) for unusual SSL/TLS warnings. Likewise, if a **browser suddenly pops up a TLS certificate warning** for a popular site or a push service, it could be a sign someone is trying to intercept you (for instance, a warning that `courier.push.apple.com` certificate is untrusted would strongly suggest a MITM, since Apple's real cert should be trusted [4]). Certificate Transparency logs are another resource: one can search CT logs for rogue certificates issued for domains like google.com or apple.com – this

has caught past incidents of misissued certs by malicious actors. In summary, **log monitoring** for any cryptographic errors or unexpected certificate authorities is an important forensic step.

- **Payload and Content Integrity Checks:** To detect if notifications or content were modified in transit, one can compare them against a known authentic reference. Some apps include a signature or hash with content. For example, if a news app digitally signs the articles or a messaging app signs notification contents, a user (or the app itself) could verify that signature. A mismatch would indicate tampering. In practice, most content is not individually signed, but forensic investigators can often reconstruct what *should* have been delivered by retrieving it from the origin server or caches. If the push notification shown on a device differs from the message that the app server intended to send (which might be logged on the server side), that's evidence of manipulation. One technique is **browser-level payload capture** – for instance, using developer tools or a debugging proxy to record what data an app is receiving. If an attacker injected something like a hidden script or image, it might appear in the captured HTML/JSON. Comparison of these captures over time or across devices can expose discrepancies. In one real case, GreatFire activists detected Chinese injection by noticing that accessing an HTTP site from within China returned extra malicious script, while outside China it did not [9] [11].

- **App and Metadata Tracing:** Sometimes the *absence* or alteration of expected metadata is a clue. Push notifications typically carry certain metadata like the sender's service ID, message IDs, etc. An enterprising user could examine the notification logs on Android (via `adb logcat`, which often shows FCM message receipt info) or use Apple's debugging profiles to get push delivery logs. If an attacker were injecting rogue notifications, there might be anomalies such as unknown sender IDs or device tokens that don't match the official app. Additionally, checking the **integrity of the app itself** can be important – if a malicious actor cannot directly hijack push delivery, they might opt to *replace the app* (through a fake update) with one that communicates with their own server. Thus, verifying app signatures and versions (to ensure you're running the authentic developer-signed app) is a basic but vital step; any irregularity might indicate a tampering at supply-chain level rather than network, but the effects would be similar (user sees manipulated content). Another advanced technique involves **tracing network route changes**: using tools like traceroute or VPN endpoints in different regions can reveal if your traffic is being proxied through unusual locations. If your connection to a service normally goes directly but suddenly it's going through an IP that belongs to, say, an unexpected ISP or a known surveillance node, that could indicate malicious routing (BGP hijacking or DNS poisoning upstream).

- **User-Visible Artifacts:** Occasionally, MITM interference creates user-visible glitches. For instance, if a page is modified, it might break something in the content (content not loading fully, or mixed content warnings in browser if HTTP elements are injected into HTTPS pages). An injected fake notification might contain odd characters or poor localization that savvy users notice. One historical example is the *Turkish Telecom ad injection scandal*, where users on that ISP saw additional ads/injected scripts on webpages – the extra content was a clear sign when users compared notes. Thus, encouraging users to report strange notifications or content is part of detection: e.g. if a news app's push says one thing on one device and a different thing on another device (under same conditions), something is wrong.

In essence, **forensic detection of network tampering** relies on catching inconsistencies – in network handshakes, in delivered content vs. expected content, or in logs – that betray the attacker's interference. Using a combination of network monitoring tools (IDS, packet capture), system log analysis, and cross-checking with trustworthy sources, defenders can gather evidence of a MITM or injection attack. It is often through such diligent comparison that campaigns like the FinFisher ISP

injections were uncovered (security companies noticed unusual redirects and malware where there should be none). Once evidence is collected (logs, packet dumps, altered content samples), it's important to preserve it for further analysis and potentially to notify the service providers (Google/Apple) so they can investigate if their channels are being abused.

Defense and Mitigation Strategies for Users

Defending against network-level intrusions and push notification manipulation requires a combination of technical measures and vigilant practices. Here are **practical methods** for users and organizations to maintain integrity and gather evidence:

- **Use End-to-End Encryption and Secure Channels:** Ensure that as much of your communication as possible is end-to-end encrypted or at least served over TLS. Apps like Signal, for instance, encrypt message content such that even if an attacker intercepts the push notification, the actual message content isn't in it (it's just a generic "You have a new message" or a cryptographic blob). This prevents content tampering since the attacker cannot alter or read the actual message without the keys. For web content, prefer HTTPS everywhere – as noted, DNS or HTTP hijacking is trivial to exploit, but with HTTPS an attacker *must* also compromise the certificate to do harm, which will usually either fail or leave signs [2]. Where possible, use services with **certificate pinning** or enable plugins like HTTPS Everywhere to avoid downgrade to plaintext. If you are accessing critical information (bank, news, etc.), consider manually verifying the SSL certificate or using a browser that supports extra integrity checks (for instance, Chrome and Firefox leverage Certificate Transparency to catch illegitimate certs). Some modern browsers and operating systems will outright block connections if a known good certificate is suddenly replaced by an untrusted one (e.g., Chrome detecting a MITM cert).

- **Secure Your DNS and Verify Responses:** Since DNS spoofing is a common entry point, use secure DNS services. Options include **DNS-over-HTTPS (DoH)** or **DNS-over-TLS** resolvers, which encrypt DNS queries and make it harder (though not impossible) for an attacker to inject false records. You can also use DNSSEC-validating resolvers, which check the cryptographic signatures on DNS records to ensure they haven't been tampered with. While DNSSEC isn't widely deployed for all domains, it can protect those that support it. For a user-friendly approach, using a reputable VPN service for all connections will tunnel both your web traffic and DNS queries through an encrypted channel, effectively shielding them from local ISP or Wi-Fi attackers [29]. Be aware that a sophisticated adversary could compromise BGP routing to even intercept VPN traffic, but that raises the bar considerably. In high-risk situations, using the Tor network can anonymize and distribute your traffic, making targeted manipulation more difficult (though not impossible, as exit nodes could tamper with plaintext HTTP – so still use HTTPS on top of Tor).

- **Verify and Limit Push Notification Data:** As a user, you have limited control over how push notifications are delivered, but you can minimize exposure. In your mobile settings, review which apps have permission to send push notifications and disable those you don't need – fewer channels mean fewer avenues for exploitation or surveillance. On Android, consider using **alternate push frameworks** like UnifiedPush (for apps that support it) which allow you to route messages through your own server or a trusted intermediary rather than Google's FCM – this can mitigate metadata collection by big tech or at least give you insight into what goes through the service [30]. For highly sensitive communications, rely on apps that do not reveal content in push (or can work without push by manual refresh). Developers and advanced users can enable logging for push notifications (Android's Developer options or iOS's Apple Configurator can enable Notification logging profiles) to keep a record of what was received. This can later serve

as evidence if an unexpected or suspicious notification appears. Also, keep your device firmware updated – both Apple and Google patch their push messaging systems and OS components regularly (for example, Apple introduced BlastDoor in iOS 14, a sandbox to safely process iMessage content after the Pegasus attacks [31] [32]).

- **Hardening Against TLS Interception:** Check the list of **trusted certificates** on your devices periodically. Remove any that you do not recognize or that were added by third-party apps/profiles you no longer use. For instance, if you participated in a corporate BYOD program, your device might have a MDM profile that installed the company's CA – if you're no longer under that policy, it's wise to remove it to prevent potential abuse. Both Android and iOS allow viewing installed root certificates (on iOS under Settings > General > About > Certificate Trust Settings; on Android under Security > Trusted Credentials). Be cautious about installing any configuration profile or app that asks you to trust a new certificate authority. If you suspect your traffic is being intercepted (e.g., you consistently get certificate warnings or see unusual behavior), try visiting a test site in a browser that will flag interception. Websites like badssl.com have pages with intentionally untrusted certs to see if your connection is being silently MITMed (if you *don't* get a warning on a bad cert site, something is very wrong). Enterprise users can deploy browser extensions or anti-MITM tools that detect changes in certificate fingerprints (some antivirus software also does this, ironically sometimes flagging legitimate corporate proxies).

- **Monitor for Indicators on the Device:** Keep an eye on device behavior: sudden battery drain or network activity could indicate spyware (if an attacker did manage to deliver malware via push or MITM). Use mobile security apps or built-in tools to scan for known spyware. On Android, you can use apps like **NetGuard or TrackerControl** to monitor which servers apps are contacting – if you see an app that normally connects only to `api.whatever.com` suddenly reaching out to an unknown server, that could be a red flag. On iOS, the options are more limited, but the device will alert you if a configuration profile is installed or if the device is supervised (which could indicate an MDM or monitoring). For web, browser extensions that validate content (like HTTPS integrity or content hash checkers) can be employed for critical pages. Organizations might deploy client-side scripts to periodically verify that important web content (say, a downloaded update file) matches the known good hash – this kind of **client-side attestation** can catch tampered downloads quickly.

- **Collect and Preserve Evidence:** If you do detect something suspicious – say you received a push notification that seems fake or a webpage that is altered – **document it immediately**. Take screenshots of the message or page, save the raw HTML if possible, and use packet capture tools to record the network traffic. Tools like Wireshark or tcpdump can be run on a laptop tethered to a phone (or on the router) to capture the raw packets for analysis. Many modern phones support creating an **offline diagnostic packet trace** (for instance, Android's Developer Options include "Take bug report" which can capture some network info, and iOS has a Logging profile for networking). If you suspect high-level interference, consider sharing the evidence with security researchers or the affected service provider. For example, Google has a mechanism (SafetyNet) to detect network-based proxies in some cases; if you feed back evidence, they might block a rogue certificate or warn other users. From a legal standpoint, maintain a chain-of-custody for evidence if you plan to pursue it – keep logs, times, and any relevant system info. For web content differences, you can use tools like `diff` on the HTML or a hash comparison to clearly show modifications.

- **Operational Security (OpSec):** Finally, adapt your behavior knowing these threats exist. When discussing sensitive topics or organizing (e.g. journalists, activists), verify information from multiple channels. If you get a news alert that seems sensational, cross-check the news on

official websites or other sources in case your feed was tampered. Use out-of-band verification for critical messages (for instance, if you get a push notification to reset your password, confirm by going to the app directly). Educate your peers as well – many attacks succeed because users are unaware of them. By raising awareness that, for example, *push notifications can be spied on by governments* [33] [13], people will be more cautious about what info they tie to notifications (maybe disable message previews, etc.). If you're in a high-risk category, consider using devices with **enhanced security OSes** (like GrapheneOS for Android) which have network and logging features to detect anomalies, and avoid relying on infrastructure that is known to be surveilled (for instance, Chinese-made phones in certain regions have been found with backdoors; using trusted hardware/software is part of defense).

In essence, defending against these threats is about *raising the cost* for the attackers. By using strong encryption, verified endpoints, and monitoring, you reduce the chances of being an easy target for network MITM or push manipulation. And by collecting evidence of any suspected attack, you contribute to the ecosystem's ability to identify and shut down such operations.

Conclusion

Network-level attacks that combine **intrusive logging** with **notification/content manipulation** represent a powerful toolkit for surveillance and information control. From the examples above – state censorship and injection systems, ISP-level spyware campaigns, and exploits abusing push services – we see that these techniques have moved from theory to practice in the real world. Defenders are not powerless: thorough network audits, transparency reports, and collaboration between tech companies and researchers have shed light on these covert tactics. Moving forward, it's crucial for platform providers to harden push notification systems (e.g. ensuring that even metadata is protected or anonymized) and for the security community to continue developing detection mechanisms (such as the Snort rules for QUANTUM inserts or browser safeguards against MITM). Users and organizations must stay vigilant, applying the best practices outlined – from using VPNs and pinned certificates to watching for anomalies and promptly installing patches that close exploit pathways. By combining technical defenses with user education and forensic readiness, we can significantly mitigate the risk of man-in-the-middle intrusions and ensure the integrity of the information delivered to our devices. The battle is an ongoing one, but knowledge and preparedness are our best weapons against these complex, evolving threats.

Sources: The analysis above synthesizes information from expert literature and documented cases in network security, including reports by Citizen Lab, security researchers, and news investigations. Key references have been provided inline (in brackets) to allow further reading on specific points [8] [15] [10] [20] [17], among others. These illustrate the state of the art in both attack and defense, as of 2025. By learning from these sources and real-world examples, defenders can better anticipate attackers' moves and protect the communication infrastructure that we all rely on.

---

1 2 What is DNS spoofing Man in The Middle Attack?- Security Wiki
https://doubleoctopus.com/security-wiki/threats-and-tools/dns-spoofing/

3 Trustwave revokes "MitM" certificate, vows never to issue one again - Help Net Security
https://www.helpnetsecurity.com/2012/02/08/trustwave-revokes-mitm-certificate-vows-never-to-issue-one-again/

4 5 28 tls - Is Apple's push notification service implementation vulnerable to a MitM attack - Information Security Stack Exchange
https://security.stackexchange.com/questions/56749/is-apples-push-notification-service-implementation-vulnerable-to-a-mitm-attack

6 7 29 How to Prevent Man-in-the-Middle Attacks I Arctic Wolf
https://arcticwolf.com/resources/blog/how-a-security-operations-approach-can-prevent-man-in-the-middle-attacks/

8 9 11 12 26 China's Great Cannon
https://citizenlab.ca/2015/04/chinas-great-cannon/

10 ISPs inside Turkey and Egypt spread FinFisher spyware in massive espionage campaign | CyberScoop
https://cyberscoop.com/isps-inside-turkey-egypt-spread-finfisher-spyware-massive-espionage-campaign/

13 14 33 Governments spying on Apple, Google users through push notifications - US senator | Reuters
https://www.reuters.com/technology/cybersecurity/governments-spying-apple-google-users-through-push-notifications-us-senator-2023-12-06/

15 16 30 Governments May Spy on You by Requesting Push Notifications from Apple and Google
https://thehackernews.com/2023/12/governments-may-spy-on-you-by.html

17 18 31 32 The Great iPwn: Journalists Hacked with Suspected NSO Group iMessage 'Zero-Click' Exploit - The Citizen Lab
https://citizenlab.ca/2020/12/the-great-ipwn-journalists-hacked-with-suspected-nso-group-imessage-zero-click-exploit/

19 20 21 22 24 25 27 How to Detect Sneaky NSA 'Quantum Insert' Attacks | WIRED
https://www.wired.com/2015/04/researchers-uncover-method-detect-nsa-quantum-insert-hacks/

23 Symbolic Triggers and Fixed Action Patterns in Humans
https://www.psychologytoday.com/us/blog/clinical-and-forensic-dimensions-of-psychiatry/202502/symbolic-triggers-and-fixed-action