Systematization of Solutions to Downgrade Attacks

Reshma Vasudevan
Technical University of Munich
Department of Informatics
reshma.vasudevan@tum.de

***Abstract*—Any area of technology employing protocols or systems that have a history of older, insecure versions that still need to be supported are susceptible to downgrade attacks. As long as the legacy systems in place cannot be completely boycotted or there exist delay in switching entirely from an old to a latest version, downgrade attacks are a possibility. Many protocols have employed a variety of solutions to downgrade attacks which end up being very specific to the concerned area. In this paper, we inspect many such attacks and their solutions and try to extract the characteristics of the protocol which makes a particular solution applicable. These features then act as requirements for the applicability of a solution which when inspected are found to be present in other fields as well thus helping us see their cross-cutting nature. This gives us the ability to derive new types of solutions for a given protocol or to recognise them as prospective solutions to other downgrade attacks in different areas than the ones considered in the paper based on the requirements it satisfies.**

bit hard to understand

v long sentences

I. Introduction

Many real-world systems employ protocols to communicate with each other and these go through many version updates for multiple reasons such as fixing bugs, providing more efficiency or being more secure. The roll out of a new version usually does not mean a quick adaptation to it. This may be because of the natural delay in implementing the uptake in thousands of devices or simply the fact that some legacy systems need to be supported using only the old versions that they offer to support. Due to the need to support previous versions which are often less secure than the newer ones, there is an innate vulnerability of a downgrade attack trying to use the old versions without the knowledge or consent of the involved parties and thus compromising security of the whole communication.

Downgrade attacks are found to happen in a wide range of areas. TLS being one of the most important protocols to establish secure connections between a client and server has faced downgrade attacks over multiple versions starting from its predecessor SSL and has had to update to higher versions after each version was found to be vulnerable in a different way to downgrade attacks [1]. The current version TLS 1.3 is found to be the most secure version. A downgrade in itself is not a compromise but it can be used to further exploit a system which is now in a more vulnerable state. HTTPS, in addition to possible TLS downgrade has some more attack surface that can be used to downgrade to HTTP. Mobile networks employing 5G which use encryption to secure user data has to be downgraded to LTE connections when connecting to a different network other than its home network. In this case even though this is an expected behaviour of the system, we may want to still augment additional security even in the downgraded state. Hardware security also faces the threat of downgrade attacks when its multiple components may be downgraded and the system still proceeds to work without any error even though the compromised components now pose a risk to the security of the overall system.

Even though solutions have been proposed for the attack on the different areas mentioned above, these were found to be very specific to the protocol or system they are dealing with. But all the solutions have a principle idea behind them and we intend to bring clarity to which aspects a particular solution tries to tackle and its general principles and when it would be a good solution. Since downgrade attacks are a cross-cutting problem, it is only likely that the solutions to it can be so too. We show that it is possible to derive new types of solutions for existing downgrade attacks based on the requirement guidelines that we identify for that solution.

There exists a systematization of the various downgrade attacks that are found based on the TLS protocol [2] but no resource was found that categorizes the solutions to these attacks. Since it is obvious that a downgrade is possible in any area that involves multiple versions, it might be important to have a place of reference to provide possible solutions that can be employed based on the characteristics of that field.

Thus, we intend to make the following contributions:

- Understand downgrade attacks in multiple fields employing communication protocols or other security systems.
- Analyse the proposed solution to these downgrade attacks.
- Formulate generic guidelines or requirements for employment of this particular solution.
- Find cross-cutting solutions as examples based on the set up guidelines.

This paper is structured as follows: the background knowledge necessary to understand some terms used in the paper is mentioned in section II. In section III, multiple areas are inspected for the downgrade attacks and based on these, some generic solutions are framed. In section IV we lay out the requirements identified which would act as the decision makers for when a particular solution should be applied. In section V we try to draw some examples where the generic guidelines can be used to derive new solutions for existing attacks and also limitations of this paper are discussed. In section VI we provide a conclusion.

II. Background

An attacker model commonly used throughout the paper is the Man-in-the-middle attacker, abbreviated as MITM. A MITM is usually placed in between two communicating parties and is able to eavesdrop on the information being sent. An active MITM may also drop packets, or inject new ones. A MITM is a really difficult adversary to beat in a secure system due to its powerful capabilities. Many attacks mentioned in this paper assume involvement of a MITM adversary who could downgrade the system and later exploit the vulnerabilities of the older versions.

SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are protocols used to send information securely over the internet. TLS is a more secure version by using authenticity and higher encryption strengths. HTTP is used to transfer text-based messages across the internet and HTTPS is the secure version of HTTP as it uses SSL/TLS to encrypt the messages being sent [3].

A common vulnerability in SSL and TLS upto version 1.2 was the support of export-grade ciphers. These are the cipher suites which were weaker due to smaller key length, say 512-bit keys, which allowed them to be easily breakable. Instead of downgrading the protocol version in the downgrade attack, the ciphersuites can also be targeted and forcing the communicating parties to use the export-grade ciphers compromises security of the communication [1].

Another area of downgrade attacks we venture into is on the hardware front. ARM Trustzone is a technology to create a Trusted Execution Environment(TEE) that divides operations into two worlds: a secure world to run security-critical applications and a normal world separated using hardware-enforced mechanisms. Each has a separate OS and applications and an authentication mechanism is performed before an application is loaded into the secure world.

III. Areas of Downgrade attacks and their solutions

A. *Different modes of operation: strict and flexible*

DNS is a mechanism used to translate domain names which are in human readable format into their corresponding IP addresses. It has a hierarchical structure and employs a decentralized administration using delegations to go up the hierarchy when needed. The DNS queries are sent from stub resolvers acting on the client side to recursive resolver servers which may contact other root servers and finally authoritative resolver in case the information is not already present in the cache and then the response is sent back. DNS primarily operates in plaintext format which makes them vulnerable to a MITM attack.

DNS-over-HTTPS (DoH) is used to provide a more secure way of transmitting DNS packets to prevent their manipulation and ensuring integrity and authenticity by encrypting them using HTTPS [4]. This occurs in two primary steps: first, the address of a DoH resolver is obtained by contacting a DNS server which happens in plaintext and second, a connection is established to this DoH resolver using TLS and further DNS request and response are sent over this secure channel. But there are ways to downgrade DoH to plaintext DNS thus nullifying the security that they provide. For the Google public DNS, a DoH query on the browser side would have a format like *https://dns.google/dns-query{?dns}*

An adversary could use DNS poisoning to store a fake IP address and send it as a response intead of a genuine DoH resolver, thus the communication has automatically been downgraded to DNS. Another attack is targeted at the second stage of communication which has an encrypted channel. TCP reset injection can be done by a MITM who can eavesdrop to get the sequence and acknowledgement numbers and then forge a TCP reset packet to cut off the secure channel.

Most browsers support DoH communication and have two modes of operation which can be set by the user: Strict Privacy Profile and Opportunistic Privacy Profile. In strict privacy, only DoH communication is allowed and a 'hard-fail' occurs in case it is unable to establish DoH. Whereas in Opportunistic privacy, a fallback to plaintext DNS is allowed in this case.

Thus, by using the strict mode of operation, the user is able to thwart all attempts of downgrade that may occur and ensure a secure way of operation always. User may still be given the option to switch to plaintext mode according to preference and hence the flexible mode may be used in that case.

It is found that even though these modes have been provided in browsers, turning on strict mode for DoH was not very user-friendly, hence along with providing the two modes, it is important to facilitate easy switching between them.

The addition of DoH does not require major changes in existing infrastructure as DoH proxy can be used on existing DNS servers on the local network to redirect communication to servers supporting DoH (like google public DNS and cloudflare). This eliminates the need to set up completely new servers supporting DoH. The initial message although is still sent in plaintext, this can be overcome by installing a proxy on the machine itself thus ensuring that messages are encrypted as soon as they leave the user.

Another area where a similar approach is followed is for HTTPS communication. Most websites nowadays use HTTPS for their connections to ensure confidentiality, authenticity and integrity when compared to HTTP. But there might be cases when a web application receives a HTTP request even though it supports HTTPS, this may occur, say, due to hardcoded links in webpages. In such a case the website redirects the client to use HTTPS to communicate. But this can be easily defeated by a MITM who can intercept the HTTPS redirect from the server and send an HTTP response instead thus making the client proceed with HTTP and effectively downgrade the connection [5]. This can also be prevented by instructing the browser to always communicate to a website using only HTTPS if this is supported. This can be set using a header, Strict-Transport-Header, which is returned by a website along with a max-age to indicate an expiration time, after a secure access has been made once using HTTPS [6]. This information is recorded by the browser which will only use HTTPS to access this website

even if it is specified otherwise until the max-age, which can be updated by sending a new header message. Thus this can be seen as a setting on an individual level for each website that the client accesses where he adapts a strict mode if provided with the header and a flexible mode if no such header provided yet.

This strategy has a potential of being applied universally to all kinds of downgrade attacks by providing the user with a platform to decide for himself whether he wants complete protection against downgrade or he is flexible with facing downgrade if that happens. Since most areas would possess an interface with the user, this can be used to provide this setting. A compromise between such extreme ends of operation - all hard fails or potential for downgrade attacks is to provide a notification system, which would ask the user on detecting a downgrade whether he still wants to proceed which will be discussed in the next section.

Thus, requirements for this solution are identified as:

- Provision for user to specify whether he wants a strictly secure mode of operation or a fallback is acceptable if required, i.e. a user interface.
- A hard fail is acceptable in terms of not being too disruptive in case of strict mode of operation.
- Easy switching to strict mode should be facilitated to ensure effective usage of this provision.
- Downgrade is detectable.

B. Notify to user on downgrade

A problem identified with DoH is that even though browsers support it and provide settings for its operation in a strict mode, it is not enabled by default. Thus if the user is not protected by default and if he is in an adversarial environment he would not be aware of the attacks because the default 'opportunistic privacy profile' does not provide notifications to user when a downgrade occurs.

Even though this was advised by [4], the browsers weigh the usability more in this case as users may find too many notifications annoying and hence are discouraged to implement this, saying that it is a feature and not a bug. There exists the tradeoff between usability of software, which may be reduced on multiple notifications, and its security but some compromise can be found such as notifying only once for a given website.

This approach is also used in the HTTPS connections for websites in browsers. This acts as an alternative way of solving the HTTPS to HTTP downgrade by providing a setting such as that in Chrome to 'always use secure connections'. By default, on connecting using HTTP to a website, Chrome already displays 'Not Secure' in the address bar. By additionally setting to use secure connections, a silent downgrade is prevented and the user is made well aware of it by providing a certificate warning and asking if user still wants to proceed and only if user asks to continue the website is loaded. Such notification systems are effective when flexible settings are used and downgrade occurs as it makes sense to notify the user in this case so that further actions can be taken by the user. Since realistically, most cases would prefer to use a flexible approach instead of a hard fail, it would be important then to incorporate the mechanism to notify users.

The approach to notify on downgrade seems to be the most fundamental reaction and solution, in a way, to such an attack. This would make sure that the user gets to know about the downgrade and he can take actions accordingly depending on deciding whether he deems it safe to proceed or it was a critical information to get and he would rather abort the connection. Notification systems can also act as a compromise between the extreme modes of operation - strict where hard fails can happen and flexible - where there is a potential for a downgrade attack. By providing notifications as a warning in the strict mode and still giving a possibility to continue with the downgrade we add more flexibility as in the case of the browser setting to allow only secure HTTPS connections but the user may still proceed if he wants to after seeing the warning. Whereas the DoH team wants to go for a stricter approach by providing the notifications in the flexible mode before downgrade by still notifying the user about it. Thus depending on the severity and required security level, one may choose where to incorporate the notifications in case of different modes of operations. But this solution is independent of whether such modes of operation exist and one can still notify the user on downgrade and ask if he wants to proceed.

Requirements:

- Facility or platform to notify user
- User should be able to take action based on the notification
- Downgrade should be detectable

C. Global storage utilisation

DNS uses different types of records to store different kinds of information. The IP address is stored in type **A** records and a record type **TXT** is provided to store arbitrary human-readable information. The TXT records allow storage of textual information in the form of multiple key-value pairs with a maximum character limit of 255 characters. As suggested by [7], we can use this as a global storage space to store information required to prevent downgrade of TLS protocol used to establish secure client-server connections.

Downgrade in TLS can happen on two fronts: either the ciphersuites or the TLS protocol version itself. The initial stage called TLS handshake is used to negotiate both of these. Since a MITM could intercept these messages and downgrade to an older, less secure version of TLS or cause the involved parties to use export grade ciphersuites which are easily breakable, a good solution would be to have prior knowledge of the highest version that one party supports and already use that to establish the connection. For servers which have not specified this information, a default mode will be applied where a fallback to older versions is allowed in case of an unsuccessful connection attempt.

The TXT records can be used to store information advertised by the server by adding a new field with name, 'DSTC' (DNS-based Strict TLS Configurations) and attribute tlsLevel

```
tls12    IN    TXT
"name:DSTC;validFrom:01-10-2020;validTo:01-10-2021;tlsLevel:strict-config;
 revoke:0; ..."
```

Fig. 1. Sample DSTC record

set to strict-config if it supports both the latest TLS version and the latest ciphersuites. When a client wants to establish connection to a given domain, upon contacting the DNS server to resolve its IP address, the client can also collect information regarding the TLS config. This prior knowledge will remove the need for negotiation of TLS version and ciphersuites between the client and the server thus removing an important attack surface for downgrade.

A time period where the strict config is valid for the server can be provided by using two attributes, validFrom and validTo. Also a server can opt out of the strict config by using a revoke flag set to true in which case the record information is invalidated. A sample DSTC record including just the relevant fields for this paper would thus look like as shown in figure 1.

An important concern arising with a global storage is its authenticity. To ensure this a different version of DNS, called DNSSEC (DNS Security Extensions) is proposed to be used where cryptographic signatures are used on the records to ensure authenticity. A Zone-signing key (ZSK) is used by each DNS Zone to sign the records which is in turn signed by a Key Signing Key (KSK) and then an upper-level ZSK to form a chain of trust. Thus attacks like DNS poisoning to alter information stored in DNS servers are prevented by making use of the integrity provided by signatures.

Since DNS is already a part of the process of connecting to a domain, only additional requirement is the storage for the information which is also provided by DNS in form of TXT records, thus adding minimal overhead to implement this solution. This is also a property of this type of solution to not incur additional overhead.

Another area of application of this style of solution is for the HTTPS to HTTP downgrade. Here, a downgrade may still happen if the very first connection is manipulated and a MITM could intercept an HTTPS response given by the server and send an HTTP response instead, thus forcing the browser to remain on HTTP. The user can use prior knowledge about a server and set a header, Strict-Transport-Security, to connect only using HTTPS to a specific website. The servers can advertise their preference to use just HTTPS using a global list, called HSTS Preload list, which is maintained by Google and integrated into browsers.

A global storage based solution to downgrade attacks is beneficial when there exists an active version negotiation phase. By moving this away to a different step that is more trustable in the form of static information with some kind of trust added to it, we are able to transform the attack area into a less severe one depending on the trust mechanism. So for DNS storage, using public key cryptography a well-formed trust is available and for the HSTS Preload, say the Chrome browser can trust the list maintained by Google. For this solution, the problems associated with having a global storage arise, say updation, deletion and syncing. DNS mechanism obtains the information in realtime before connecting whereas HSTS list is integrated into browsers. This might also lead to centralization and thus a single point of failure issue. Since there might be some overhead introduced with this method, it is beneficial if the global storage is introduced in some step that already acts as a precursor to the connection establishment such as how the DNS resolving is an existing initial step required before connecting to a website.

Requirements:

- An active version negotiation phase exists.
- There should exist a global storage which can be utilised to store some information. If such a system already exists, it is more efficient to use it.
- One of the parties advertise their support for the latest version using the global storage, this would be used as prior knowledge by the other party.
- There should be minimal overhead introduced to incorporate this mechanism.
- The source of prior knowledge should be trusted and should have some mechanism to authenticate it.

D. Different identifiers for different versions

Hardware security involves using a different space for executing secure operations by segregating hardware into a normal world and a secure world [8]. Each have their own OS and their own applications. To install an application in the secure world, it must be 'trusted'. This trust is obtained through a verification system which verifies the cryptographic signature used to sign the hash of the application. The booting up follows a chain of trust which would first verify the bootloader, then the OS and then the applications thus having every component existing in the secure world to be verified. The applications in the secure world, also called trustlets, should ideally not work when replaced by an older version. But since there are not multiple keys for multiple versions, even if the trustlet got replaced by an older, vulnerable version the system would still proceed normally.

As described in [8], this attack was conducted on mobile devices which were using ARM Trustzone, a Trusted Execution Environment, by obtaining root privileges by exploiting other vulnerabilities and then replacing an application in the secure world by an older version and the system was found to work as normal. This happened because of a lack of checking for the version of the application being used which allowed an easy downgrade. Many smartphones were found to be vulnerable to this attack and similar approach can also be used to downgrade the bootloader and the OS.

A solution to this problem is to use different keys for different versions of the application thus ensuring the verification would also take the version into consideration. A problem with this approach is maintaining and distributing so many

keys which might be difficult based on the number of versions that an application may have over its course and the mode of distribution.

A version of an entity should be able to identify for itself. In other words, on the entity's side it is set and written in stone what version it runs and thus its security, the user only need to verify this. This approach may make it difficult to maintain different such identifiers which tell themselves apart to a user, thus multiple such identifiers should exist corresponding to the version. This also means that once the verification is done there is no other surface where meddling can happen. This is why this works best in case of hardware security where the application is in the local vicinity and it is hardware-enforced that no other factors can affect the operation. Whereas other applications that operate on-the-air such as TLS can still be affected by changing other parameters such as ciphersuites.

This specific solution can still be generalized to have the following key points or requirements: bit unclear

- Identification process represents the absolute truth, i.e can be completely trusted.
- The authentication process is able to verify the version number.
- The authentication process precedes the secure operation, in this case loading of target into secure world.
- Downgrade goes undetected.

E. Local storage or version control

Another solution for the hardware downgrade attack on TrustZone as mentioned in [8], is to use version control to store locally the valid versions of trustlets. This helps in checking the version before loading an application into the secure world. When a vulnerability is found in a specific outdated version, the version is increased and the latest version as stored would only be allowed to be executed.

In contrast to the previous approach, the local storage pulls the information to the user side and maintains it there. Although this gives complete control and authenticity the the user, there is the overhead of maintaining it by syncing properly. Still this appears to be a universally applicable solution if there is a way to know in advance the version to be used. The local storage also only makes sense if you have a not very dynamic target space.

The requirements would be:

- A storage space locally to maintain the list of valid versions.
- Multiple versions exist.
- Must communicate early on which version is to be used from a trusted source.
- A fixed number of targets whose versions need to be stored
- A mechanism to update the version to higher one should be available, say, finding out that an older version is no longer suitable.

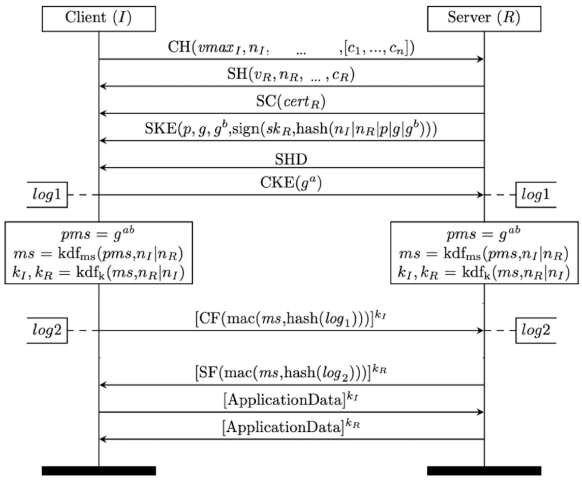

Fig. 2. Sequence diagram of TLS 1.2

F. Verify the agreed on settings

TLS is one of the most important protocols used for establishing secure communication between clients and servers over the internet. It has two phases: first is the handshake protocol where the configuration such as the TLS version and ciphersuites to be used are negotiated between the client and the server and second is the record protocol, where a secure connection is established according to the parameters agreed upon in the previous stage and data is then transmitted in an encrypted manner [2].

Figure 2 shows a sequence diagram of the messages being transmitted in TLS 1.2 as described in [2] with some intricacies removed to focus only on the relevant details for the attack considered in this paper. A client hello message is the first message and it is sent by the client mentioning its maximum supported TLS version, a nonce, a list of ciphersuites that it supports and some additional parameters. A server hello message is sent as response by the server finalising the TLS version considering also the client's offer and selecting a ciphersuite from the list provided by the client, along with a nonce of its own. There is no safety mechanism employed at this stage thus texts are sent in plaintext. Now the server also sends its certificate for verification which also contains the server's public key which is relevant for RSA encryption. In the next step which is Server Key Exchange, the server sends the key parameters for secure connection establishment in case of DHE encryption and also sends a signature over both nonces and the key parameters. A server hello done concludes the communication from server's side for the handshake protocol. The client would then verify all of the above and send its side of key generation parameters in a Client Key Exchange message. Both server and client generate the session and master keys based on the parameters and this concludes the handshake protocol.

After the handshake an integrity check is included by

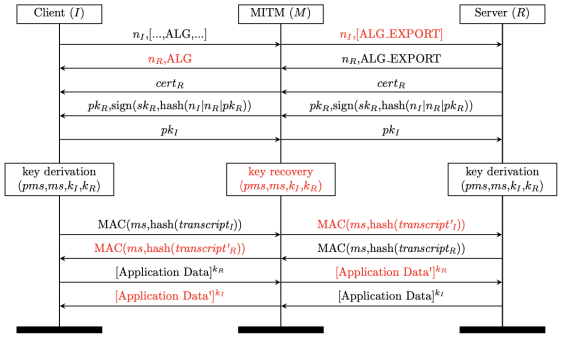

Fig. 3. Attack on TLS 1.2

performing an MAC over the transcript of handshake using the keys that were just negotiated. This is first sent by the client with its view in a Client Finished message and then same process is repeated by the server this time also including the above message in a Server Finished message. From this point the data can be transmitted using the session keys for encryption.

As shown in figure 3, the attack mainly makes use of the fact that the handshake messages are sent in plaintext and the support for export-grade ciphers exist. The attacker acts as a Man-in-the-middle and replaces the ciphersuite list to mention only export-grade ciphers from the client's side. Then the server is forced to choose one of them. To prevent suspicion from the client, he replaces the ciphersuite with a strong one when sending the response to the client. The rest of the messages, namely certificate and key exchange are simply forwarded as they only check integrity of nonces and key parameters and thus the ciphersuite replacement goes undetected. Now, because the keys were generated for an export-grade cipher, they are usually weak and thus breakable. The MITM is thus able to also retrieve the keys and for the MAC calculation he forges it according to the respective views that the server and client have. Since he also knows the session keys he can see the data that is sent encrypted using those keys.

A solution found for this attack and which was employed in the latest version, TLS 1.3, is first to remove the support for export-grade ciphers but more importantly to also verify the integrity of the ciphersuites. Thus the signature now includes the hash of the whole transcript and not just the nonces and key parameters. Also the key exchange is moved to the very first client hello message, thus encrypted communication is begun as soon as possible, namely immediately after server hello.

Thus even though encrypted communication is not possible in the very first steps of connection establishment (due to lack of established keys) in TLS 1.2, this can be compensated by ascertaining the agreed config by re-sending them in an authenticated manner and then proceeding with the communication if this step was successful.

This technique can be used in cases where an active negotiation phase exists thus exposing an attack surface with multiple parameters that can be changed. This can lead to the two parties maintaining different views of the negotiation. If we can re-affirming the concluded settings using some mechanism which can ensure authenticity, for example encryption this can prevent undetected downgrades.

The requirements could thus be mentioned as:

- There is an active negotiation phase involved.
- The involved parties share different views of the agreed upon settings.
- There exists a way to ensure authenticity of a communicated message.
- It is not possible to include authentication in the initial stage of protocol version establishment.
- Downgrade goes undetected.

G. Confirm the downgrade: TLS SCSV

TLS has gone through many version updates due to vulnerabilities found in each of them and applying patches for them. Recent versions of TLS, up to TLS 1.2, were found to be vulnerable to a downgrade attack called a 'downgrade dance'. Since there exist multiple buggy legacy servers that do not support latest versions of TLS, it might happen that some such buggy servers would crash when provided with a higher TLS version during handshake than what they support. In this case the client responds by attempting to establish the connection again this time with a lower TLS version which is acceptable by the server and thus leads to a successful connection. A MITM can exploit this by intentionally dropping packets to imitate a buggy server crashing. Thus the client would resort to lower versions of TLS and proceed to establish a connection in a less secure way, making it easy for the MITM to exploit the further communication by taking advantage of the vulnerabilities present in older versions of TLS. The server would also then respond thinking the client only supports a lower version and proceed with it.

A fix provided in TLS for this attack [9], is to include a flag, SCSV (Signaling Cipher Suite Value) whenever a client wants to downgrade to a lower version even though it supports a higher one. This is provided as a dummy or fake ciphersuite in the ciphersuite list supported by the client. The servers that dont support this fix can simply ignore it. The presence of this dummy ciphersuite indicates that the client downgraded to an older version only because it thinks that the server does not support the higher version that the client had offered [10]. If that is true, the server proceeds with the connection. If that is false and server finds that the mentioned version is lower than what it is capable of supporting it will immediately abort the connection as there clearly has been some meddling with the initial packets. Thus, this acts as a confirmation before downgrading, in a way, asking the server to be sure that it required the downgrade.

Until TLS 1.2 though the issue of lack of integrity check on ciphersuites exists and this solution is still vulnerable to MITM attacks that can alter the ciphersuite. This does require enforcing the use of an export-grade cipher by the MITM using

which he can break the keys and manufacture a new handshake transcript to bypass the check done by the client and server. In TLS 1.3, the downgrade dance attack was solved in a different way by using the last eight bytes of the server's nonce to signal the TLS version that the client offered. Since this is part of the handshake transcript check and since TLS 1.3 does not support export-grade ciphers, its integrity is protected.

Thus for cases where one party has willingly chosen to downgrade, this approach additionally ensures that this was actually called for. This helps in ruling out any third party actions that may have been the cause for the downgrade. Important factor would be the requirement of an authenticated way of confirming this downgrade.

Requirements would be:

- There is an active negotiation phase involved.
- There can be occasions where one party is unable to respond correctly to a version negotiation.
- A second step can be used to confirm when a downgrade is required.
- The second step may either be authenticated or may have special flags to indicate that this is a different type of message (confirmation message).

difference to previous sugg?

H. *Add-on security on downgrade*

5G and LTE are used for communications in mobile networks. A mobile may have to move from one location to another and establish a connection in a new network there and when this happens sometimes a downgrade to LTE is required because not all networks support 5G yet.

For a user, a long-term identity is used which requires to be privacy-protected otherwise it can make them susceptible to tracking and monitoring. But the identity of the subscriber needs to be known in order to correctly bill the amount to him/her. A solution in 5G to the privacy problem is that the identity is encrypted while communicating hence preventing eavesdroppers to get hold of it. Since this is not supported by LTE another way of using identities without revealing the original identity needs to be employed.

This is done by the use of pseudonyms [11]. A pseudonym which acts as a temporary identity is used when connecting to an LTE network. A pseudonym is a randomized identity and cannot be assigned to more than one user at a time. For as long as a user requires to be connected in a LTE network the pseudonym is used as its identity and after a certain time of it not being in use, the pseudonym is deallocated so that it can be reused for a different user. The log of the pseudonym allocation is maintained until billing is performed.

Also a thing that is observed is the additional overhead upon connecting to a lower version, as pseudonyms need to be maintained and deallocated and reused and these are additional information needed to be stored along with the actual identities. This additional overhead helps in adding more security to the lower version even when we cannot completely eradicate the lower version from the system.

Thus for cases where a downgrade cannot be avoided, it is advisable to add some additional security to the existing techniques if possible. This would require considerable overhead which in turn can act as an urge for the lower versions to upgrade themselves.

Requirements:

- It is difficult to completely remove the lower versions
- Connecting to a lower version cannot be avoided.
- There is some method to add-on security to the lower version.
- Introduced additional overhead is bearable.

I. *Use some information/authenticity from the higher version*

Although the use of pseudonyms in the 5G to LTE downgrade solution is very specific to the problem of privacy in order to maintain unlinkability by using a temporary identity, we can still find a general approach that has been used to effectively manage the downgrade. Using some information or validity obtained upon connecting to a higher version, we can employ this when we connect to a lower version to have a more secure communication.

The creation and updation of pseudonyms used in LTE is usually performed by a secure source i.e. on connecting to an authenticated 5G network, a user gets allocated the pseudonyms which he can later use when he connects to the less secure LTE network. Thus the pseudonyms can be trusted as they come from the higher protocol version. In this case there is also a dependency of LTE connections on 5G connections, as a user needs to connect to 5G before connecting to LTE to obtain fresh pseudonyms if its allocated ones have expired.

The approach of using HSTS header in HTTPS to HTTP downgrade can also be thought of to fall under this category because once a browser connects to a website using https, it receives a header indicating the browser that the server would like to only be connected using https. This information can only be obtained after connecting to the website using https and thus once we obtain this header, we can use this information to connect only using https from now on.

For cases where it is possible for the party to connect at least once to the higher version, this opportunity can be used to obtain some information which can be used to manage his further actions either in a vulnerable environment or to make decisions in the future. This information may be required to be refreshed which can be done in the connections that may follow. This can be critical information as the secure version can act as a trusted source.

Requirements:

- At least one successful connection to the secure version should precede.
- Some information exists that can help user make critical decisions or apply in an untrusted environment.
- The information needs to come from a trusted source.
- The higher version is able to generate or maintain this information.

TABLE I
CATEGORIZATION BASED ON IDENTIFIED REQUIREMENTS

Field of attack	**Requirements**										
	Active negotiation phase exists	***Downgrade detectable***	***Global storage available***	***Local storage available***	***User interface available***	***Auth. mechanisms/trusted source***	***Definite no. of targets***	***Higher version contains info.***	***Multiple versions***	***Lower version unavoidable***	***Add-on security methods***
DNS-over-HTTPS	✓	✓	✓	✓	✓	✓	×	×	×	×	×
HTTPS	✓	✓	✓	✓	✓	✓	×	✓	×	×	×
Hardware security	×	×	✓	✓	✓	✓	✓	×	✓	×	×
TLS	✓	✓/×	✓	✓	✓	✓	×	✓	✓	✓	×
5G/LTE	×	✓	✓	✓	✓	✓	×	✓	×	✓	✓

IV. REQUIREMENTS IDENTIFIED

After analysing various attacks and their solutions, we can broadly summarize the requirements identified as below:

- **Active negotiation phase** There exists a negotiation phase in real-time which is used to decide over some of the settings that will be used for further communication. This can be version or other parameters that have the capacity to be downgraded.
- **Downgrade detectable** Whether a downgrade can be detected or not. Detection helps in using some solution types such as notification on downgrade. Some areas may have both detectable and undetectable downgrades based on the attack type such as in TLS where a MITM downgrade may be undetectable to both sides whereas a downgrade dance leads to an intentional downgrade from the client side.
- **Global storage** A global storage should be available which can be used to store some information regarding the version, this acts as a pre-existing knowledge and can help move the attack surface from active negotiation.
- **Local storage** A local storage should be available at client/user side to store version information that it requires. This could be done on a priority basis.
- **User interface availability** Whether the user can interact with the application through a user interface. This is for example, a browser for HTTPS protocol.
- **Authentication mechanisms/ Trusted source** Whether some mechanisms to ensure authenticity of information exists. This may also be in the form of a trusted source, for example, in case of 5G, the Home Network supporting 5G is trusted by the user and is used to maintain information about user's pseudonyms.
- **Definite number of targets** This means that the targets that we are interested in engaging with are definite in number. This is relevant for hardware solutions where you want to maintain a local version control for the applications you want to load into the secure region.
- **Higher version contains information** The more secure version holds some information which can help the user decide before taking certain actions such as version information.
- **Multiple versions** This requires multiple versions of the protocol to exist and is relevant to solutions such as version control to keep track of these versions.
- **Lower version unavoidable** It is unavoidable to completely remove the possibility of connecting to a lower insecure version, thus intentional downgrades are allowed. This is the case with TLS and 5G/LTE.
- **Add-on security methods** It is possible to use some methods to enhance security of lower version in the form of add-on techniques. This might be useful to improve security even if you are forced to use a lower version.

Thus to reiterate, notify III-B requires a user interface and downgrade to be detectable. Confirm downgrade III-G can be applied to protocols having an active negotiation phase where downgrade is unavoidable in certain cases and some authentication mechanisms exist that can be used to verify information. The rest of the solutions also can be referred to using the above requirements which are also mentioned in the specific sections.

V. DISCUSSION

After categorizing each of the solution using its general principles, it is easier to identify certain areas where each of the solution type can be used. It can be seen that some solutions are already employed in more than one area and some act as fundamental solutions to downgrade attack. For example, different modes of operation (strict and flexible) III-A and also notify on downgrade III-B are specifically mentioned as a solution for the DoH (DNS-over-HTTPS) downgrade to its plaintext version when communicating in browsers. But we identify that the solution to HTTPS to HTTP downgrade also falls under the same category. We can also see this as a derived solution for TLS attacks. As TLS has a user

TABLE II
CROSS-CUTTING SOLUTIONS

Field of attack	Diff. modes	Notify	Global storage	Diff. identifiers	Local storage	Verify settings	Confirm downgrade	Add-on security	Info. from higher version
DNS-over-HTTPS	✓	✓							
HTTPS	✓	✓	✓						✓
Hardware security			●	✓	✓				
TLS	●	●	✓	●		✓	✓		●
5G/LTE	●	●						✓	✓

interface in the form of a browser and because downgrades due to attacks such as downgrade dance are detectable, we can employ the different modes of operation and notify in downgrade for TLS. We find that such measures already exist in the browser for setting desired TLS versions and that just proves the success of this method as it was not an examined but a derived solution of this paper. And same applies for LTE where the user mobile device would act as the user interface.

Another solution type to use global storage at first looks very specific to TLS by using DNS servers to store version information. But the approach to use HSTS Preload List to prevent HTTPS downgrade also falls under the same category as both of them try to remove an attack surface occurring due to active version negotiation to a precursor step by using a global storage to save this information beforehand.

Similarly the solution to use different identifiers appears very specific to hardware security to identify which version of an application is running by verifying its signature which is specific to the version. This could suggest a solution to TLS attacks by using different certificates for different versions for the same server. This would mean that verifying the certificate already proves, in a trusted way, the version that would be used for the communication. The feasibility of this solution is not discussed in the paper as creating new certificates for each version would be a hassle, but this is a possible way identified as a solution.

We can also observe that this solution of using a strict-transport-header follows the same principles as the solution type to use some information form the higher version III-I. Although this type was very specific to the problem of maintaining privacy during 5G to LTE downgrade, the principle that the higher version can provide some sort of trust or validity of information is something that other fields can also use.

The solution type to use different identifiers for different versions is found to be very specific to the downgrade in hardware security III-D. Formalizing the solution into guidelines helps us see that this solution is not suitable for the DoH downgrade to plaintext DNS. This is because as mentioned, the solution is effective if we have multiple versions of the target such that each version can be identified differently. But since DoH downgrade involves only two possible modes, it would not be suitable to use a versioning system for this. A better solution there is to use the notification based system.

Also we see that the local versioning system suggested as a solution to hardware attacks can also be switched to a global storage which may come from the corresponding hardware manufacturer.

The solution to obtain some information from the secure version is done in the form of obtaining pseudonyms when a user connects to 5G and using these as identifiers when downgrade to LTE happens so that privacy of the user is preserved as the original ID need not be communicated. This approach is found to be similar to what HTTPS does by sending back a Strict-Policy-Header after the first successful connection. This information from the trusted HTTPS connection is then used by the client to make a decision to only connect using HTTPS to this website. We can also derive a similar solution for TLS which would allow the client to save TLS version information of a browser after a successful connect happens thus the client can use this version to connect the next time.

Also, the solution to hardware downgrade can be augmented by adding on the notification based system to it before loading the applications when it is found that a lower version has been loaded. This is usable as sometimes a user may want to manually use a lower version and in this case he can ignore the notification.

Since we do not consider all attacks that have taken place in a specific protocol, for example, TLS, we cannot say that the requirements apply generally to the protocol but here we intend to consider only the attacks looked at in the paper.

Of course, this paper is not an exhaustive list of the downgrade attacks and their solutions but we want to present this possibility of using existing solutions to act as general guidelines so that in future they can be easily adopted to other areas where a downgrade attack is identified to be a possibility.

VI. CONCLUSION

In this paper, we analysed the various downgrade attacks that exist as a threat in multiple fields employing communication protocols or other security techniques. We went through the solutions suggested to these attacks and even though these were found to be very specific to the concerned field, we could extract some general principles that each solution has. This helps us to formulate each solution using some general guidelines as to when they should be applied. Also we found out how this was useful to identify some of these techniques to be cross-cutting into multiple fields and having the potential of being employed in new fields other than from

where they originated. The guidelines also help us to rule out its application when some criteria is not followed and thus may not be a good solution to a given field. This categorisation also enables us to extend these solutions to new areas after identifying the requirements that are met for a particular solution to be applied. As this is not an exhaustive list, the further work could involve adding more types of solutions that may be inspired from even different fields of technology.

REFERENCES

[1] Downgrade in TLS, "What is a downgrade attack and how to prevent it." [Online]. Available: https://crashtest-security.com/downgrade-attack/

[2] E. S. Alashwali and K. Rasmussen, "What's in a downgrade? a taxonomy of downgrade attacks in the tls protocol and application protocols using tls," 2018. [Online]. Available: https://arxiv.org/abs/1809.05681

[3] TLS, SSL, HTTP HTTPS, "What is tls, ssl, http https? how do they work together?" [Online]. Available: https://www.websecurity.digicert.com/en/in/security-topics/what-is-ssl-tls-https

[4] Q. Huang, D. Chang, and Z. Li, "A comprehensive study of {DNS-over-HTTPS} downgrade attack," in *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*, 2020.

[5] Andrea Chiarelli, "Preventing https downgrade attacks." [Online]. Available: https://auth0.com/blog/preventing-https-downgrade-attacks/

[6] HTTPS downgrade, "Strict-transport-security to prevent https downgrade." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security

[7] E. S. Alashwali and P. Szalachowski, "Dstc: Dns-based strict tls configurations," 2018. [Online]. Available: https://arxiv.org/abs/1809.05674

[8] Y. Chen, Y. Zhang, Z. Wang, and T. Wei, "Downgrade attack on trustzone," 2017. [Online]. Available: https://arxiv.org/abs/1707.05082

[9] TLS RFC for SCSV, "https://www.rfc-editor.org/rfc/rfc7507.html." [Online]. Available: https://www.rfc-editor.org/rfc/rfc7507.html

[10] How TLS SCSV helps, "https://crashtest-security.com/enable-tls-fallback-scsv/." [Online]. Available: https://crashtest-security.com/enable-tls-fallback-scsv/

[11] M. Khan, P. Ginzboorg, K. Järvinen, and V. Niemi, "Defeating the downgrade attack on identity privacy in 5g," 2018. [Online]. Available: https://arxiv.org/abs/1811.02293