Chapter

7

Security in a SIP Network

Security in the IP network is most certainly the most important part of any implementation for traditional service providers. This is because they have been operating in a highly secure network environment for decades. Today's legacy networks are very mature and highly secure networks, mostly due to the implementations by their operators.

Certainly if you ask operators of the world's leading telephone companies what keeps them up at night, you will quickly find network security at the top of that list. In fact it is security that has kept many companies from quickly embracing IP as a network transport and adopting IP for all their service delivery.

It is very ironic, then, that with security such a concern, many operators have such lackluster security measures. This includes established VoIP providers who have been victimized more than once by skillful and ingenious hackers and fraudsters. It is not that IP cannot be secured as much as the implementations are weak from a security perspective.

One of the reasons may be that securing a network can be difficult and requires a lot of specialized expertise. Most service providers do not want to accept the expense of hardening the security around their networks, and many more simply do not understand the measures that must be taken to prevent security breaches.

Should operators be concerned about security? According to statistics from both the GSM Association and from the Communications Fraud Control Association (CFCA), absolutely. Both of these organizations show that fraud is on the rise, mostly due to the deployment of IP technologies in the network.

In fact, fraud was on the decline until recently. As operators began deploying more and more VoIP networks, they began to see these networks come under attack by highly organized and well-funded organizations looking for ways to earn profits either by pumping their own traffic over other operators' networks, or by bypassing operators altogether and building their own peer-to-peer networks connecting through the operators' data networks.

These organizations are also highly educated and possess the resources necessary to breach even secure networks, although they would rather focus their efforts on those networks that are just too easy to breach and require little effort or resources.

When you look at the types of breaches being realized, you quickly learn that the attacks could have been prevented through some very simple measures. For example, a young man on the West Coast was sentenced just this year for hacking into thousands of computer systems looking for platforms that still had their default passwords intact.

He would then scan the systems looking for open ports that could be used for pumping his own traffic, or he would open up ports himself. Armed with the IP addresses of the systems he hacked and reconfigured, he worked with his partner (who actually perpetrated the whole business) to sell routes to legitimate telephone companies at cheap rates.

They routed millions of telephone calls through application servers sitting in businesses and universities across the U.S., charging telephone companies for access to their "network" and pocketing millions in profits. The main perpetrator has fled the country and cannot be found.

I bring this incident up as an example of how telephone companies (and in this case even well-established VoIP operators) can be defrauded of millions of dollars by even small operations. The hacker was later quoted as saying how easy it was to break into the computers and systems and change their configurations.

What was the most surprising aspect of this? That established service providers had deployed VoIP servers without changing the factory default passwords! The most rudimentary security measure had been skipped in these networks. In the many lectures I have done on the topic, I am always amazed at how many people do not change their passwords on a regular basis.

But passwords are only one measure. Changing passwords regularly is certainly one way of slowing down hackers and fraudsters, but there are many more measures that should be taken to further protect any SIP network.

Why is it then that we know security is important, and yet no one seems to be concerned about it (or at least not concerned enough to implement robust security measures)? Security can be very inconvenient not only for the operator, but for the subscriber as well.

Types of Network Attacks

There are many different forms of network breaches, but here we will talk about those that occur the most often. In fact, these are the types of attacks that are seen repeatedly not only in VoIP networks, but the Internet as well.

The Internet can serve as an excellent model of what could go wrong given its wide-open nature. Nothing can be trusted in today's Internet, and fraud and Denial of Service (DoS) attacks are rampant. Anyone who spends any time following security alerts and law enforcement activities in this area quickly understands the Internet is a risky place to do business unless you are prepared to lose a large portion of profits to fraudsters.

That is not to say that the Internet is a losing proposition; certainly there are many businesses thriving on the Internet. But they are also losing a lot of money from fraud, and they could be making a lot more if they didn't have to deal with security issues.

A case in point is the current concerns over Web sites and e-commerce. Certainly there have been numerous reports of sites being compromised, sensitive information being stolen from online sites, and consumers attacked with spam and phishing attacks on a regular basis.

Since SIP is a derivative of HTTP and SMTP, both Internet protocols that are exploited on a daily basis, it only makes sense to understand the vulnerabilities these protocols experience today to better understand the potential threats within a SIP implementation. The 3GPP and the IETF are also actively adding new procedures and enhancements to the protocol to further harden the protocol against network attacks.

Let's look at the various types of network issues, how they work, and how they impact the network operator and the subscriber. Then we will look at ways in a SIP network to prevent these attacks from happening (or at least some good measures to decrease the probability of their occurring).

Understanding how hackers approach a network is important to understanding how attacks begin, and what one should look for. Hackers approach attacks in stages, beginning with probing of the network.

Hackers first look for networks with easy access. This means a network with open gateways and platforms with default passwords still in use. They then begin looking for vulnerabilities in those platforms by scanning the platforms. Once a vulnerability is found, it is recorded for later use.

Many security breaches can be found at the scanning stage. In fact, you could use this approach on your personal computer to find possible viruses and bots that are scanning ports for open access (this is done through software utilities that provide information regarding the activities on your computer platform). This is the same technique used for network elements.

There are distinct characteristics to probing activities that you can look for to determine if there is pre-attack activity going on in the network—for example, multiple requests to the same server from the same UA, or maybe many requests from one UA to multiple servers and proxies.

Once a vulnerability has been identified and validated, it is used later for a larger attack. It is a combination of vulnerabilities that allows hackers to reach through many different networks, traverse through all of these networks unnoticed, and launch large-scale DoS attacks on unsuspecting networks.

Here are some common methods used when attacking a SIP-based network.

Registration Hijacking

SIP uses clear text messages, meaning anyone with a computer and some programming knowledge can "tap" into a network and capture SIP messages. This is different from bit-oriented protocols that simply transport "frames" of bits that when grouped into a

defined format can be decoded to specific messages and parameters. ISDN and SS7 are good examples of bit-oriented protocols.

Since SIP uses clear text, if a hacker can capture these messages, that hacker is able to read subscribers' sensitive information such as their public and private identities. This information can then be used to "spoof" a subscriber. In other words, the hacker can use this information to gain access into the operator's network for his or her own use.

Let's say a subscriber has registered with the network, and the subscriber's location is recorded by the registrar. All calls and e-mails, instant messages, and any other session traffic are sent to this location.

The hacker then accesses the same network and uses the same subscriber information captured when "sniffing" or "tapping" the network to register with the network. Since the hacker is using the identity of the legitimate subscriber, her or she is granted access. However, the registration from the legitimate subscriber is not changed.

To the network it appears as if the subscriber has changed locations in the network and sent a new registration. The hacker has changed this through his or her own registration with a new location that now gets stored in the registrar. This means that all session traffic for the legitimate subscriber will now be sent to the hacker's destination instead of the subscriber's device.

Now eventually the subscriber will change the registration again while changing locations (provided the subscriber is mobile), but the hacker already has the subscriber's identities and network permissions and is therefore able to use this information to gain network access anytime he or she wishes.

Another means of accomplishing registration hijacking is to capture a subscriber's registration message and then "replay" the same message using a new location. This effectively registers the subscriber with the hacker's location. This is one of the more common means for hijacking registrations.

Session Hijacking

Session hijacking works much like registration hijacking, but this attack is used differently. A session hijacking is used to take over a session in progress. Session hijacking began with the World Wide Web.

A Web server is not a stateful server. A session consists of the UAS accessing a page from the Web server. If a subsequent page is accessed, that comprises another session (or at least new authentication from the user and the Web server). To alleviate the need of authenticating continuously when using a Web site, Web developers created the concept of cookies.

A *cookie* is nothing more than a data file, usually consisting of the session ID. The Web server sends the browser the cookie when the site is first accessed. The cookie is then sent by the browser application each time it accesses the Web server for another page to identify itself.

This concept was expanded for use with online shopping sites to maintain the shopping cart. When you visit an online site and wish to review your shopping cart,

the cookie sends your authentication information so that you don't have to keep typing in your password.

If these cookies are intercepted and copied, they allow the interceptor full access to the session already in progress. This means the hacker now has access to all of your transactions and account information (so long as the session is still in progress). A cookie can be expired when the session is over, or it can be set for longer durations.

Many sites generate cookies using an algorithm that uses the timestamp and the IP address of the user to generate a unique identifier. This is an easy identifier for hackers to guess using random generators. Cookies themselves are somewhat controversial for many reasons, but mostly because they are misunderstood (many believe they are executables rather than data files). The usage of cookies is full of vulnerabilities, however, making them very susceptible to session hijacking, so their use within a SIP network is not highly recommended.

This is especially true if WiFi is going to be the access method into the network. WiFi networks are still very susceptible to eavesdropping, and the use of any clear-text transmissions is risky. Cookies can easily be captured through eavesdropping on access points, which would then compromise network services.

One simple check for hijacking is to check the time and date stamp of an incoming request or response. When a UAS receives a request, it should check the date and time with its own internal clock. If there is a discrepancy (more than 30 minutes, for example), then it is very likely that the request was intercepted and has been replayed with a changed destination address.

This happens when a session is hijacked and the message is captured for replay. The hackers will change the origination address of a message to their own and insert the message back into the network. If they do not update the *DATE* header, then the message will appear as if it has been traversing the network for some time, which is not normal—a session request that has been traversing the network for a long period of time (30 minutes is a long time) will most likely be deleted, as the interval specified in its *MAX-FORWARDS* header will have been exceeded.

Impersonating a Server

If you spend any amount of time on the Internet, you will most likely run into this form of security breach. There are many Web sites that look exactly like their official sites but are in fact hacker sites used for stealing information from unsuspecting consumers.

For example, there have been many documented cases where hackers have created sites to look like bank or credit card Web sites, using official logos and modeling the Web site to look exactly like the real Web site. When the subscriber types in the URL for these sites, they get redirected by a redirect server.

The redirect server is compromised by the hacker to send consumers to their Web site rather than the real Web site. Once the consumer has reached this site, they are asked for password information and other sensitive account information. This information is then stored on the server by the hacker and sold on the Internet to other hackers.

Sometimes the hacker will send out e-mails prompting consumers to go to the hacker's Web site to update their accounts or to claim refunds. Again once they reach these sites, they are asked for sensitive account information that can then be used to access the consumers' accounts. There are many breaches of this nature, from text messaging as well as e-mails.

Then of course there is always the possibility of compromising the DNS. This is known as DNS poisoning, where the DNS server is hacked and the IP address for specific servers is changed to a spoofed Web site or address. This is very likely in SIP networks where DNS is used to identify the IP address for domains and their applications.

The damage would be the same in a SIP network as anywhere else. For subscribers to be redirected to rogue application servers could have serious impacts on both the subscriber and the operator.

Tampering with Message Bodies

Since SIP is forwarded in clear text, it is not necessary for someone to have a decoder. Simply capturing the message is good enough, and this is easy to accomplish if one has access to any number of sniffing utilities and is connected to the same network. Once the hacker has captured a message, the message body and the headers in the SIP message can be modified.

For example, a hacker may capture an *INVITE* from a subscriber and change the *FROM* header to reflect his or her own address. This would provide the hacker access to a network he or she is not authorized to use, and would allow him or her to initiate sessions with other subscribers while pretending to be someone else.

There are other concerns with message tampering. Since text messaging is sent in a SIP message body and is also in clear text, a hacker could easily intercept SMS messages and change their content. This could be especially troublesome if messages were being sent by government agencies during catastrophic events.

Message tampering can be easily prevented, as can many of the scenarios we talk about in this section through encryption. Encryption prevents the hacker from being able to decipher the text, and therefore the hacker would be unable to change it and route it back to the network.

Tearing Down Sessions

Tearing down sessions is a concern but not as troublesome as a denial of service attack. Hackers could intercept requests from various subscribers and simply send a *BYE* message as a response (as if it came from a proxy or other network element). This would then terminate the session and cause it to be torn down.

This is more of a nuisance attack and is most likely to be launched against individuals, since it is far easier to launch a DoS attack against the network or network elements, with much more devastating and far-reaching effects.

Denial of Service and Amplification

Denial of service (DoS) attacks can take many different forms and can be launched using many different techniques. The easiest form is simply flooding the network with specific traffic types. For example, using a call generator, a hacker can send millions of *INVITE*s into the network attempting to flood the network with call requests.

We see these types of attacks take place many times, and they have even occurred in the PSTN. The use of SIP and IP provides much easier access to hackers, enabling these types of attacks much more frequently.

Another form of DoS attack involves application servers. By launching a flood of requests to an application server, the network element is immediately flooded and congested, taking it out of service. This can also happen with the DNS through flooding with DNS queries (similar attacks of this nature have occurred many times already). When the DNS is attacked, the entire network can be impacted, depending on where the server sits within the DNS hierarchy and whether or not redundancy has been implemented.

When redirect servers are used, the potential for amplification occurs. One message is forked into many different messages, which will also result in many different responses. An attack relying on amplification will send many requests toward targets known to be redirected, knowing that those targets will result in the request being multiplied.

As the request is multiplied, it is sent to several different destinations. Each one of those destinations will then send an appropriate response back to the forking proxy. It is not necessary for the responses to be successful responses, since the object is to create a large volume of SIP traffic in the network with the hope that this causes enough congestion to result in a DoS.

Since congestion is the ultimate goal, one request is obviously not enough. Nor is one target sufficient for such an attack. Therefore the attacker will use many targets and millions of requests, and will continue to send these requests until the congestion occurs.

The registrar can also be the target of such an attack. A hacker can register a subscriber listing many different user identities for the same subscription. This then provides the registrar with a list of multiple destinations for a request. The hacker then launches requests toward the public identity, which the registrar and proxies then send to multiple destinations based on the registration made by the hacker.

This is also considered amplification, as the registrar is "amplifying" the effects of the attack by sending to multiple destinations. The impacts are the same as a forking proxy, bearing the same results.

A similar kind of attack toward the registrar involves registering many different identities. Each identity consumes memory within the registrar, and therefore if a large number of registrations take place, the registrar runs out of memory.

This only works in open networks with little to no security where anyone can register and use the services of the network to route requests. Hopefully most networks will prevent this from happening just through simple authentication, preventing unauthorized subscribers from accessing the registrar.

The forking proxy in turn will continue to fork the requests into many requests and will continue to manage many responses from the various targets until it becomes congested (or some upstream network element becomes congested). At any rate, the end result is that the network becomes too congested to handle any further traffic and begins denying service to other subscribers. Some of the network elements may even crash due to the levels of traffic.

Bots and DDoS Attacks

Bots are simple scripts that are carried to a subscriber's device through Web sites, or other viruses transported using text messaging or e-mail. Even Bluetooth is highly susceptible to viruses, and cell phones have fallen victim to these as well.

A bot sits on a device and first looks for any open ports or connections that it can use to access the Internet. The bot then looks for other computers or devices connected to the same network and begins exploiting these systems. This is what makes bots especially dangerous, since they have the ability to self-propagate whenever the device is connected to a network. Even firewalls cannot prevent bots from infecting other computers when an infected computer connects behind the firewall.

But the most troublesome aspect of a bot is the ability to control the script from a remote server. Bots use the Internet Relay Chat (IRC) protocol to communicate with a URL programmed into the script. They then receive commands from a server connected to the URL that basically launches the script. The server is the command and control server.

Many bots are used for accessing Web sites and specific links on those Web sites to increase the number of "hits" to a Web site fraudulently. There are some businesses that make money based on the number of hits to a Web site or Web site link, and therefore the bots inflate the revenues at will.

Bots have now moved on to more menacing and threatening uses, causing DoS attacks in many networks. When one hacker successfully launches bots, they create their own little network of bots known as *botnets*. All of the scripts are now under the control of one person, which if launched against a target could be devastating for that target.

For example, if the bots were instructed to access the same URL at the same time, millions of machines could be accessing the same Web site simultaneously. This would certainly cause the server to crash because it would not be able to handle such a huge demand. The Web site would then be out of service.

This has happened more than once, causing businesses to close their Web sites for days. The losses quickly ramp up when your Web site, your sole source of revenue, has been attacked and shut down. Imagine now if that Web site was a bank, or a credit card company.

The largest botnet to date was recorded in 2007. Millions of computers were under the command of one botnet. If that botnet were directed to any URL, it would have a devastating effect. Of course, bots affect more than just computers.

Imagine if cell phones were infected with these bots. Already we are seeing cell phones become infected with experimental viruses, propagating through text messages and Bluetooth. If a botnet were to be amassed, the hacker would have millions of cell phones at his or her disposal. Imagine now if all of those cell phones placed a call at the same time to the same destination. That portion of the network would be out of service for an undetermined amount of time.

Even if the bots were instructed to do something as simple as send a text message to a cell phone, the end result would be a lengthy denial of service for that portion of the network. It has already been demonstrated that just a few text messages could be sent to a single cell site resulting in a lengthy outage. Millions of phones sending messages to many phones in a sustained attack could easily cause an entire network to quickly crash.

The effects of such an attack would be crippling. Not only would the operator lose substantial revenues, they would also lose the faith of the subscribers. Of course, if the attack were held in conjunction with a physical attack, the results would be especially devastating.

Fortunately law enforcement agencies have been diligently searching for botnets and have successfully stopped many of them, but they continue to proliferate. It has become more important than ever to prevent unauthorized attacks on your networks and protect the resources from being compromised.

Security Measures

Security remains one of the biggest issues in packet communications today. When one looks at the many security breaches and network attacks to date, security was seriously lacking. We already talked about one specific case where the perpetrator was able to hack into thousands of computers where even the simplest of security measures (changing the password) had not been practiced.

Security can be very robust and sophisticated, and it can be very simple. Don't be fooled into thinking that highly complex and sophisticated security measures will prevent attacks on your network. Some of the world's most secure networks have been breached.

However, this is no excuse for not implementing any security at all. When you look at the makeup of network attacks, there are basically two types of attackers: one that is looking for the easiest networks to breach (little to no security) and one who seeks out those with very high security (the challenge). Why, you might ask?

Think about those networks with very high security. There is usually something there worth stealing; otherwise, there wouldn't be so much security. If there is something of value to the hacker, then the hacker will attempt to break in. There is also the concept of a challenge. There are many who just want to be able to say they were able to do it as a personal challenge.

So what does this mean for the average network operator? This means you certainly have something worth stealing, so you definitely need to protect it. But you need not

spend a fortune doing so, since we already know that the best security available is still breachable. You should take some basic measures at the very least to prevent being attacked just because your network is so easy.

Know that the more security you put into place, the least likely a hacker is going to choose your network unless there is something specific they want from your network. It's the same concept used in physical security.

For example, when putting an alarm system in a home or business, simply placing signs and stickers at every entry point identifying your home as alarmed is usually enough to keep most burglars away. Why? They know your home is going to be more difficult (not impossible) and they can find another house next door that will be easier to break into. This is true as long as there is nothing inside they are looking for specifically (again, if you have something they want, they are going to try and get it).

Thieves look for the easiest way that takes the least effort, so keep this in mind when securing the network. Making it difficult for the thief but somewhat non-intrusive to the subscriber is the difficult balance. There should be as much transparent security as possible so that the users of your services are not impacted (there is nothing worse than having to enter a password three or four times to access a service).

There are many ways to implement security. Many operators have implemented session border controllers as firewalls to their network. While this is probably a good practice, thinking this will stop network attacks is dangerous. Many recent reports have proven that the majority of attacks come from within the network itself, as well as outside the network. This means that robust internal security must be implemented as well as border security to prevent the network from being compromised.

Also bear in mind that attackers are now well-funded and highly educated individuals with the backing of highly sophisticated organizations. Organized crime and terrorist cells are the most common attackers today, and it then becomes necessary to build a security plan with this in mind. We are no longer fighting the teenager with nothing to do but hack computers. We are now talking about older individuals who pursue this for a living and do nothing more than break into networks for pay.

The most basic concept in network security is maintaining a trusted domain. This means that all interconnections are made with entities that are known and can be trusted to send legitimate traffic. In security terms, the trusted domain is often referred to as the "realm."

A realm can be considered analogous to the trusted domain. It is identified in security parameters within SIP using the domain name of the trusted domain. Everything within the realm is known by the network, meaning it knows the subscribers within its own control and has authenticated all users.

Anything outside of the realm is not to be trusted and should be authenticated prior to authorizing services. This is similar to the concept used in GSM networks, where a cell phone device must first register with the network and exchange credentials. Without proper credentials, the device is denied access to the network.

This example shows how the realm is identified within a security context in a SIP *INVITE*:

```
INVITE  sip:travis.russell@tekelec.com  SIP/2.0
AUTHORIZATION: Digest realm="tekelec.com"
```

When a subscriber is in their home network, and are establishing sessions within their home domain, they are still authenticated by their home network. However, the home network has all of the subscriber's credentials and can easily authenticate the subscriber without issue.

When the same subscriber is roaming in another network, the visited network does not have the credentials to authenticate the subscriber. Therefore, an agreement must be established between the home network and the visited network. This agreement allows for both network operators to exchange subscriber details necessary to authorize services, and to bill one another for services rendered when each operator's subscribers are roaming in the other operator's network.

The reason many phones do not work in other networks when you are roaming is quite simple. The operators do not have an agreement between themselves, and therefore your device is not granted authorization to access any services. This situation is getting a little better as operators partner with more and more networks, allowing their subscribers to roam in many different countries, but there is still a long way to go.

Take the cable industry, for example. The cable industry has established a federation among all of the cable operators. All members of the federation are considered as trusted domains and can exchange subscriber and billing information with one another. This means that the cable operators have effectively created a massive network of networks where their subscribers can get services as if they were in their home network without restriction.

Of course, cable operators have not gone wireless yet. Many have offered wireless services through established wireless carriers but have not launched their own networks. This is all rapidly changing as the cable industry quietly adds more and more services to its portfolio. When cable companies do build out their own wireless networks, they will be part of a huge network composed of many different trusted domains.

There are basically six aspects to securing a SIP network:

- Authentication
- Authorization
- Confidentiality
- Integrity
- Privacy
- Non-repudiation

Authentication requires the use of passwords and the exchange of credentials. We talked a little about this already. Whenever a subscriber registers his or her location with the network, the registrar should always challenge the initial registration. This challenge is explained in more detail in the course of this chapter.

It is unfortunate that many networks simply do not challenge registrations. Instead they verify the user identity and trust that the identity is true. This is one of the reasons there are so many attacks on SIP networks today. Simply challenging the registrations could eliminate many security breaches.

Authorization requires querying a database containing the basic account information for a subscriber. This account information provides the public as well as private identities for the subscription, and all the services the subscriber is authorized to access. This can be part of the authentication process to be most effective.

Confidentiality protects the subscriber and the subscriber's identity. It ensures that conversations cannot be snooped on, and that the subscriber can exchange information freely without the information being captured by someone else. This remains one of the big challenges for network operators, especially given the many tactics being used today to capture sensitive data from subscribers.

At the same time, it is equally important that the integrity of any data sent by a subscriber be sent intact without alteration. This includes any Web sites that may have been accessed as well. It is far too easy for hackers to access SIP messages and change the contents in an effort to change the service and where it is being delivered. It is also very easy to capture a SIP message containing text and alter the text message before it is delivered to its final destination.

Privacy can sometimes be an issue when it is openly provided to anyone. Today on the Internet there are anonymous services where e-mails and other messages can be directed in an effort to hide the address of the originator, and make it appear that the message came from someplace else. At the same time, privacy can also be offered as a feature to some clients who have a need for such a feature.

One example of this is law enforcement agents. Today when they make a call, the call does not give away their identity or the number they are calling from. This is an important service to law enforcement agencies and government alike, and it should be maintained even in the SIP domain.

Finally, non-repudiation prevents subscribers from accessing services and later denying they used those services. If the operator implements the right tools and audit systems, you should have total visibility to every network transaction that takes place. This includes any downloads that the subscriber may have made.

Today this is not the case, and the reason many operators are losing money on music and ringtone downloads. While it is true they are making money on these services, it is also a documented fact that they are losing even more money. One industry analyst firm reports error rates in the reporting of downloads to be as high as 80 percent. The reason for this is simple: no monitoring systems are capable of supporting all of the protocols used within a SIP network.

We have listed the various aspects of a security deployment. Now let's talk about the specifics behind security implementations in a SIP network. These are simple

suggestions and recommendations and by no means an exhaustive description of all that can be done. Nor is this meant to be a very detailed, highly technical discussion on how these security mechanisms work. Always refer to the specific RFCs for the specifications.

Password and Access Controls

I put passwords first because this is the easiest and the most overlooked security measure there is. Passwords are painful for everyone, the consumer included. No one likes to have to deal with passwords every time they access their phone or every time they try to make a call.

What's ironic is that while we hate passwords, we would never think of leaving the house without at least locking the door, and many of us go one step further and set an alarm. Yet isn't it funny that typing in a simple password is too much of a bother?

The trouble with passwords is managing the passwords so that they cannot be compromised. This means implementing password aging, which would require everyone to change their passwords periodically. Some change their passwords every month, while others change passwords every three months.

There are several levels of password control within a network. Certainly the devices accessing the network should be protected themselves, but the network is the most critical asset to the operator, and every entity within the network should be protected as well.

This may seem obvious, yet most network operators do not change the passwords on their voicemail platforms, their gateways, or even their routers. Instead, they deploy these entities using the factory default passwords sent by the vendor. These passwords are well documented and advertised all over the Internet, which means the hackers know your passwords as well.

So the first, most rudimentary step to security in a network is to ensure that all entities within the network have their passwords changed from the factory defaults. But don't stop there, because passwords can be determined through programs developed just for this purpose. The passwords should be changed as often as can be tolerated.

This then requires an entire password management initiative. Passwords should be as long as possible (the password length makes it more difficult to crack) and incorporate as many character types as possible (numbers, letters, and symbols). This extends out to the terminals that are used to access and configure the network elements.

One of the best methods of password control (at least for terminal access) is the use of token IDs. A token ID is a completely randomized password that changes every 30 seconds or so. The password is based on an algorithm that only the software application (which resides on the host) and the password generator know. The password generator (my term) is really a little key fob–like device with a display of numbers that change every 30 seconds or more.

The numbers displayed represent the newest password, and when combined with a user's access password, they become a very difficult mechanism to hack. I strongly recommend the use of these for any terminal access, and for use on any PCs used to

access any network element. A compromised PC that later connects to a network element could put the entire network at risk, given the nature of many of today's bots and viruses.

Encryption

Encryption solves a lot of problems. The best means of ensuring privacy while preventing message tampering is to encrypt the SIP message prior to sending it through the network. This does present some problems, though.

If forwarding a SIP message that is encrypted, the various routers and proxies in the network will not be able to read the addresses contained within the various headers. Therefore the entire SIP message cannot be encrypted, unless every network element is going to be provisioned with the proper decryption keys (not a very likely scenario).

The other issue is forwarding the SIP message outside of the trusted domain. Not everything can be encrypted, as this would make it impossible for the other networks to read the addresses necessary for routing. This is why there are headers that are not encrypted, while the remaining SIP message is encrypted.

Encryption does require all receiving entities to have the encryption keys so that they will be able to remove the encryption (decrypt) and return the message to its original state. Without the encryption keys, the network elements cannot do anything with the message, so care must be given in deciding how to encrypt and when to encrypt. There are numerous ways to implement encryption.

One approach is to encrypt only messages leaving the network. This means there are no encrypted messages internally, since all of the elements are within a trusted domain. The problem with this approach is that most attacks take place from within the network. Therefore encryption for outbound SIP messages does not solve anything. There should be encryption internally as well as externally.

When a message is encrypted, some headers will remain in "clear text," which means they are not encrypted. The headers that are encrypted include:

- Subject
- Accept
- Alert-info
- Expires
- Supported
- User-agent
- Reply-to
- Accept-encoding
- Error-info
- In-reply-to
- Unsupported

- Server
- Organization
- Accept-language
- Authentication-info
- Require
- Retry-after
- Warning

The message body itself is also encrypted, which could include the Session Description Protocol. Since the SDP is encrypted, any proxies that need to read the SDP portion of the message will need to be able to decrypt the message. This is another consideration when implementing encryption, since these elements will have to know the encryption keys. This may include firewall proxies as well.

And don't forget the network receiving the SIP message. The network elements receiving the SIP message will also have to know the encryption keys in order to process the SIP requests. This is another reason that network agreements have to be made to ensure that the interconnecting networks can be trusted.

Transited networks do not have to know the decryption keys, as they do not need to know anything more than where to route the message. The headers used for routing will contain enough information for the SIP message to reach its final destination. Transport Layer Security (TLS) is a good means of providing encryption between networks, while SIP Secure (SIPS) is designed for use within a trusted domain. TLS works at the TCP layer and is best when used between two networks where two network elements do not know each other. TLS cannot be used end to end.

SIPS is used in the request-URI to indicate that TLS should be used to transport the request/response to the designated domain. Once the domain is reached, local policy determines the treatment to be used within that domain. TLS can also be used within a network, although it is best suited for interconnections with other networks.

RFC 3261 specifies that TLS is to be used at proxies, redirect proxies, and registrars when interconnecting with other networks. They should also possess a site certificate for authentication. These proxies also must have the ability to validate certificates from other trusted sites, by storing the certificates from these sites (usually elements from within its own trusted domain).

IPsec is best within a network between trusted elements. IPsec works within an "enclave" or trusted network, implemented by each of the network elements at the operating system level. Security gateways can also be used to create virtual private networks (VPNs) to make the network more robust.

Another approach to encryption is tunneling a SIP message within another SIP message. The original SIP message is encrypted and then encapsulated within another SIP message for routing. The routing information from the original SIP message is used to populate the routing headers in the outside message, but nothing else is given in the outside message.

A proxy then will only read the request-URI, *VIA, RECORD-ROUTE, ROUTE, MAX-FORWARDS*, and *PROXY-AUTHORIZATION*. These are the only headers that can be modified as the message is routed through the network. When the UAS receives a message that has been tunneled in this fashion, it compares the encrypted tunneled message with the outer message. If there are any other changes to the contents, the message is considered compromised and the original message is rejected.

To allow the sender to remain anonymous, the *FROM* header in the encapsulated message can remain as is, but the outer message *FROM* header can be set to anonymous. This helps prevent the identity of the sender from being detected while the message is transiting other networks while still providing the identity once the message is received. Of course, this header should never be trusted to begin with, since it is too easily spoofed. The network should always rely and trust only the *ASSERTED IDENTITY* header.

Tunneling messages prevents messages from being hijacked, modified, and then replayed into the network. As the original message has been encapsulated and sent within an S/MIME body, it should not have been altered. When the UAS compares the encapsulated message with the outer message, it will identify whether or not there have been any other alterations to the original message. Both the encapsulated message and the outer message are duplicated.

If any of the headers in the outer message are different than the headers within the encapsulated message, the encapsulated message of course takes precedence. The headers in the outer message are discarded.

Everything we have talked about so far, encryption and tunneling, applies only to the SIP messaging and not the bearer traffic itself. The bearer traffic should also be encrypted to prevent unauthorized access. Obviously the bearer traffic contains much more value to the hacker than just the SIP signaling, so it should be adequately protected as well.

The exception to this is in the case of the *MESSAGE* method. Text messaging uses the *MESSAGE* method to deliver the "bearer traffic," which in this case is a text message. The text message itself is carried in clear text within the message body of the SIP request.

Since the text is in clear text, it becomes even more important to ensure that the message body is encrypted. S/MIME should also be used within the body text to further protect the message when it is transmitted across multiple networks.

The SIP protocol does support communicating between network elements regarding the security mechanism to be used. The security mechanism is encryption, and the communications between entities are necessary to communicate the encryption schemes supported by the various nodes.

There are three headers defined as extensions specifically to support the SIP security mechanism; *SECURITY-CLIENT, SECURITY-SERVER,* and *SECURITY-VERIFY*. These are used to communicate to either the UAC, UAS, or upstream proxies the method to be used.

A UAC can query an upstream node to determine the encryption methods it supports prior to sending a request. It does this by sending the *OPTIONS* header. The receiving

entity then returns a response containing a list of methods it supports. If the responding entity is the UAS, it would return the *SECURITY-SERVER* header contained in its response.

The *SECURITY-SERVER* header would then list the methods supported. Possible values include *TLS, DIGEST, IPSEC-IKE,* and *IPSEC-MAN.* The receiving node then returns a request along with the *SECURITY-VERIFY* header containing the relevant security keys. A UAC uses the *SECURITY-CLIENT* header to send a list of support encryption methods to upstream nodes.

Authentication and Authorization

One method of authentication is through the use of certificates. This is implemented in many Web sites today. When you set your browser to check for certificates, and you access a network resource (such as a Microsoft Web site to purchase software), your browser will ask the Web site for its certificate.

These are public certificates that are advertised to "trusted" communities so that they are able to access the sites. This works by providing the browsers with the key for the "trusted" Web site. The key is used to determine if the certificate is valid or not. Only the key holder and the application accessing the site know the key.

The key can also be provided when a subscriber signs up for a new service, such as online banking. The online banking institution would exchange the key with the subscriber upon signup for the service. The subscriber and the online banking service then are part of a trusted domain.

The certificate is associated with a specific URI using a cryptic identity. Only the receivers with the proper key can decrypt the identity and therefore authenticate the site. This concept is not perfect by any means, but it is one way of at least making it more difficult to steal services from networks.

Anytime a service is being provided, and a subscriber sends a request that is of questionable origin (has not been authenticated before), the application server should reply with a `401 Unauthorized` response, forcing the UAC to send the proper credentials prior to accessing the service.

When the UAC receives this response, it should resend the *INVITE* again but include the *AUTHENTICATION* header in the message. The *CALL-ID* should be the same as the original *INVITE* so that the server knows that this is a second attempt to access services and it is in response to a previous challenge.

The *AUTHENTICATION* header provides the application server with the necessary credentials so that the server can provide the subscriber with the requested services. This should be practiced in all networks to prevent unauthorized access to services.

Authentication can be applied to application servers as well. As when using a certificate, each application server would be configured with a unique encryption key. Only authorized UAs would receive the proper key. When they receive a request from an application server, they would then be able to challenge the application server (the same process used against the UAC in reverse).

This would help prevent rogue application servers from sending requests to unsuspecting subscribers, and it would allow devices to challenge the servers they access. Since it is encrypted, it should be more secure than certificates.

When the *MESSAGE* method is used for forwarding text messaging, application server and proxy authentication should be enforced to prevent spoofing and spamming. As mentioned in the earlier section "Encryption," the *MESSAGE* method is used to deliver a text message. The text message is delivered in clear text within the SIP message body. This makes it vulnerable to attacks unless encryption and authentication are enforced.

A *DATE* header should also be enforced to ensure that there is a timestamp for every message. Any time a *MESSAGE* is received with a timestamp indicating it is more than several minutes old, the message should be rejected by sending the response `400 INCORRECT DATA OR TIME`.

The exception is when a store and forward function is used to deliver these messages. When a store and forward server is being used, the message will of course always be more than a few minutes old, since it was delivered to be stored and forwarded when the subscriber became available.

Strict Routing

This is a concept that the 3GPP introduced to be used within IMS networks. In a SIP network, a proxy is able to use any available route. The routing decisions are made in real time by each of the routers in the path, based on current availability and traffic conditions. This means that a request may take one path, while its responses may take another path. This, of course, makes it impossible to trace any of the messages or to capture all of the messages related to one session without capturing everything in the network and using a correlation engine to associate each of the requests/responses.

The exception to this rule is when stateful proxies are used, since a stateful proxy must see every response for a request in order for the proxy to be able to monitor the state of the session. If the proxy does not see all of the responses, it will not be able to determine if the request was successful, and it will not be able to determine whether or not a session was terminated (unless it sees the relevant messages).

Because of this type of deterministic routing, hackers are able to use man-in-the-middle attacks to intercept sessions and have them rerouted to another address. They are able to use replay to send copied messages back into the network without detection, even if they suddenly route responses to a different address than what was registered.

With strict routing, the routing path is recorded as the subscriber registers in the network. The *REGISTER* message contains the *RECORD-ROUTE* headers, used to collect the addresses of each of the proxies in the path. These addresses are saved in the order they were added, so that a route list can be established for the subscriber.

The route list then becomes part of the subscriber's registration. Stateful proxies in the path then also store the route list for the subscriber so that they know how to route

messages (both requests and responses) to the user identity. Even if a hacker attempted a session hijack, for example, the proxies would ignore the routing provided in the message, relying instead on the route list they recorded for the subscriber during the registration.

The hacker then would have to hijack a registration and successfully register his or her own address using the subscriber identity. This is still possible, unless the network is using encryption and authentication keys, in which case the hacker would be unable to read the intercepted messages (thereby being prevented from capturing the user identity) and would be unable to pass through authentication (not having the proper authentication keys).

Strict routing does have its drawbacks, though. This is not a very efficient means of network routing because it forces traffic through specific paths regardless of network utilization at that specific time. This means the network will have load imbalance, and many facilities may sit idle during certain periods of the day.

This, of course, goes against the grain of IP networks, where routing is determined according to the network conditions. It does align more with the circuit-switched network procedures, so switching engineers will be very comfortable with this form of routing, but IT engineers will not be so thrilled about strict routing.

If used with all of the other approaches we have talked about so far, this will certainly make the network a very robust and well-protected network. Nothing is perfect, and there is no such thing as a totally secure network as we have seen demonstrated time and time again. But remember that the object to security is to make it difficult for the hacker to fool the network, while allowing your subscribers to enjoy an easy and feature-rich, yet secure network experience.

Security Solutions

There are many reasons why many networks do not implement any form of security. Many platforms are older systems running operating systems that do not support security patches and must be completely replaced by newer platforms. This of course is cost prohibitive.

Other operators are severely short-handed and lack both the resources and the expertise to implement a strong security policy. They also lack the capital to invest in security implementations. Human error and configuration mistakes add to the problem, especially when expertise is lacking.

There are many different types of solutions available for securing networks. The security industry is probably one of the fastest growing tech sectors today, with products ranging the full gambit. This is both good and bad. With so many different products available, there are also just as many different approaches to security.

One worthy change within security concepts today is the layering of security implementations. Layering means implementing a security solution at all layers of the network. There are many reasons why this approach makes a lot of sense. When looking at the various types of attacks, some attacks are best detected at the lower layers, while others can only be detected at the application layers.

For example, security implemented at the network level will detect network anomalies and changes in the traffic flow but cannot detect buffer overruns in the application servers. Systems that operate at the network level have no visibility to the applications themselves and therefore are unable to detect problems within the applications themselves.

A layered approach provides a lot more flexibility to the operator as well. It allows operators to scale their security implementation rather than invest everything at the transport layer. By implementing just what is needed at the various layers, operators can save on their overall investments significantly.

Here is why. I could launch a distributed denial of service (DDoS) attack on a network that would be very difficult to detect at the transport layer. This is because I am sending large volumes of *INVITEs* from many different origination points within the network using bots. I would need to collect data from all over the network to determine that there was an attack of this nature in progress, or rely on an alarm at the destination point when it went into congestion.

However, if I am monitoring the application layer, I will see an increase in traffic in real time. Furthermore, I will see that the traffic is coming from multiple origination points and looks to be a DDoS attack. I could invest more money to implement sophisticated analysis applications at the transport layer combined with deep packet inspection, or I could invest in detection software at the application level (and most likely not have to invest as much).

The concept of layering your implementations allows you to make the right amount of investment at each layer, without having to purchase very expensive solutions at one layer that does all. It also provides a much more robust platform, allowing you to provide various layers of security for different segments of the network and various applications and services.

There is no single-point security solution that will protect the entire network, so a layered approach is the best means of ensuring your network assets are protected. Also remember that a single security solution is purpose built, designed to protect against specific types of security breaches. All security plans should incorporate several different platforms and methods based on the type of network and the services being provided.

If the services consist mostly of content, then that content is what must be protected. Digital rights management (DRM) will be required to deter copyright violations, while security measures such as those discussed will be necessary to prevent unauthorized access to the network and the content.

The following sections discuss the various solutions available to protect a network from attack. These are not exclusive; there are numerous different solutions and approaches, but these should be considered as a bare minimum.

Intrusion Detection

Monitoring systems have been used for several years to monitor the health of the network. Now they bring additional value as intrusion detection systems (IDSs). Of course,

traditional network monitoring only has visibility to the call control layers within your network. You will need a platform to provide additional visibility into the transport layers as well.

This is especially true of data-intensive networks where the primary service is data services rather than just voice. In a voice network, traditional monitoring systems can easily be implemented to detect specific attack scenarios such as ISUP flooding (in an SS7 network) or SIP *INVITE* flooding (in a SIP network).

An IDS can operate in real time (or near real time) or historical mode. A real-time system is needed for detecting attacks while they are in progress. However, these systems should also have some capacity of storage to allow for the investigation of network events at a later time. The amount of storage depends on the amount of traffic to be stored, and the duration of time you need to review.

For example, if you want to be able to pull traffic for a six-month period, you will obviously need storage capacity for six months of traffic (or possibly more). While this is not common today, there are more and more network operators who are implementing long-term storage of their traffic for investigating incidents (as well as traffic modeling, revenue assurance, lawful intercept, and many other applications).

The purpose of these intrusion detection systems is to detect and notify of certain network anomalies at the call control layer. They should have the ability to see all SIP methods, and they should have the ability to provide basic key performance indicators (KPIs). These KPIs later become important for identifying flooding scenarios and other attack methods.

The IDS identifies the source of data (the network, application, or host). It performs analysis on the traffic based on rules (policy) and could also have the ability to establish its own policy through neural technology. The IDS then sends notification of the event to a console or other reporting system.

For example, in a flooding case, the monitoring system should detect an increase in the number of SIP requests across the entire network, as well as specific network segments. This could indicate (if it is a sharp rise in the number of requests) that there is a DoS attack underway.

If the system also supports the configuration of thresholds, then the system can be set to alert the users anytime these thresholds are exceeded. This is important for identifying DoS attacks.

Another advantage to monitoring systems is the ability to measure the performance and set thresholds for the entire network, or critical network segments. Because the monitoring system has visibility to all network elements, and all network facilities (if implemented network wide), the system can therefore see traffic levels across the entire network rather than within just one entity.

If you are relying on alarms from the various network entities, you will most likely not see a distributed DoS attack until it's too late. This is because a DDoS attack comes from many different sources, originating from many UAs across the network. Without a view of the entire network, you will not be able to see all of the traffic.

The individual network elements will only be able to detect and provide data on what they see, which is only a fraction of the traffic. For example, security implemented

within proxies is great for identifying situations at the proxy, but they will miss anything that happens within other proxies, or occurs at any other point in the network.

For this reason it is best to leave network element security implementations focused on transport security, and rely on monitoring systems for security at the call control layers. This is the best way to ensure you will see all situations that could take place at that layer.

Monitoring systems can be thought of as network-based IDSs. They use probes to capture the network traffic, based on rules (or filters) defining the type of traffic. Usually the filters support defining traffic specifics such as origination points, type of traffic, and so on.

The probes then report back to a central server where the final analysis is completed. This server will also provide some form of console for reporting incidents and generating alarms. The probes themselves are passive, although they can be active in some cases.

Because they are passive, they do not require a network address. This makes them transparent in the network, which adds another distinct advantage. Because they cannot be detected, they are invisible to hackers.

The network-based IDS is best deployed both at the network edges (monitoring the access points into the network) and at aggregation points within the network (where it will be able to detect man-in-the-middle and other internal attacks).

One disadvantage to passive probes is that they do not work in networks where encryption is used. Because they are passive, they are unable to actively exchange encryption keys and negotiate these keys with other network elements. This can be a major setback, since encryption is key to any network security strategy.

There are a number of integrated solutions for detecting network intrusions at the transport layer. Host-based IDSs are typically implemented within the proxies and the gateways in the network.

The IDS should be separate from the elements it is protecting. In other words, it is not desirable to integrate the IDS function within the various network elements, since an attack on the network element would also compromise the IDS. The exception to this rule is when you are using a combination of IDS implementations. When combined with network IDSs, host-based IDSs can provide an additional advantage.

A host-based IDS monitors the various ports as well as processes within each of the network elements. This provides another set of eyes at the network element itself that a probe would never be able to detect. This can be important especially when looking for probing events where hackers are scanning platforms looking for vulnerabilities.

Another advantage of a host-based IDS is that it works within an encrypted network. Since it resides within the various hosts, the host-based IDS does not need to know the encryption keys. It is dealing with internal processes within the host itself, and any traffic data it receives will already be decrypted by the host it resides in.

Host-based IDSs are also capable of detecting viruses and Trojan horses residing on the host. Only a host-based IDS would have this visibility, since it requires visibility to internal processes and ports on the host. This is also a major advantage and key reason for implementing host-based IDSs in conjunction with network-based IDSs.

With all of the advantages also come drawbacks. A host-based IDS still needs to report back to a central authority events going on within each of the hosts. A central console is needed to collect data from all the IDSs deployed and provide reports and alarms. Otherwise, operators would have to access each host individually and extract this information on a host-by-host basis.

Backhauling all of the data from multiple hosts can be a challenge and in some cases would require a lot of bandwidth. For this reason, not every host should be covered. Certainly the critical hosts within the network should be protected.

Host-based IDSs are also susceptible to attack, since they reside within the targets themselves. This can add to the challenge and, of course, if compromised will become worthless. A good example of this would be virus protection implemented on your computer being compromised so that it receives automatic updates from a hacker site rather than the software vendor's own site. This is a tactic used today by many hackers.

Since they reside in the hosts themselves, host-based IDSs obviously do not have visibility to the rest of the network. For this reason they should not be used as the sole security implementation within a network. They should always be used in conjunction network-based IDSs.

An application-based IDS is actually a subset of a host-based IDS. It resides on the host but monitors specific applications on the host rather than the whole host. The purpose of an application-based IDS is to provide visibility to a user's interaction with the application, detecting unauthorized use of an application.

The application-based IDS should always be used in conjunction with network and host-based IDSs, to provide a layered approach to security, as well as a total view of what is truly going on within the network and its resources. When implemented in this fashion, the overall IDS strategy can provide enough visibility for an operator to detect in real-time events in the network before they cause significant disruptions and outages.

An IDS performs analysis on collected data to determine if there is an attack underway, or abnormal events within the network. There are two types of methods that are used: signature analysis and anomaly analysis.

Signature analysis relies on rules that are defined within the IDS (also referred to as policy). These signatures are established from previous attacks, so they are signatures of known attacks used as profiles for detecting the same attacks again.

These are very accurate, since they are based on known attack signatures. They are good for networks where expertise may not be abundant or resources are limited, but they should not be the only analysis used. Signature analysis is based on known attacks and therefore is not well suited for finding new attacks that use a different signature.

Anomaly analysis looks for abnormal behavior in the network. This is the best method for finding new attack signatures, but it does require additional expertise, since the analysis is usually a manual process today.

Profiles are built based on a snapshot of captured traffic over a period of time. The longer the duration of time used to collect the traffic, the better the profile. One method

may include setting thresholds in the network, and anytime these thresholds are exceeded, raising an alarm. Anything that deviates from the profile is then considered an anomaly.

Statistics can also be used for establishing profiles. Statistics such as key performance indicators (KPIs) and key quality indicators (KQIs) are commonly used today in many monitoring systems for analyzing traffic data and determining if there are security breaches.

Both methods should be used together for the most effective approach. A signature analysis implementation alone will not be effective in finding new attacks and will leave the network very vulnerable, since attack methods change regularly.

A new approach is to deploy an IDS as a "honeypot," where a host is set as a decoy in the network. The data within the host is of enough interest to draw hackers, but benign to the network operator (no harm can be done with the data). When the attacker accesses the host, information is gathered about the attacker and all aspects of the session.

This information is then used in other IDSs within the network to establish a profile and signature to be used to detect other network breaches. Sometimes the attacker is even transferred to another part of the host where he or she is isolated from the network and its resources, limiting the harm the attacker can render on the network.

In voice networks (especially wireless) there can be individual numbers within number ranges that are left unassigned (dark). These numbers are then closely monitored for activity. If there is an attempt to call one of these numbers, the call data is collected and analyzed, since it could be a hacker attempt to create a hit list within a network to be used as targets at a later time.

An IDS should be deployed using the layered approach discussed in this section, using all forms of IDS rather than just a single approach. If only one approach is feasible, however, the network-based approach is the best direction to take, since this will give you the best visibility to network activity.

Network probes should be deployed behind firewalls, as well as at network edges, on major backbones, and on critical subnets. This will give the best overall view of everything in the network. They can also be converted to include active capability at a later time (or at the time of implementation).

Intrusion Protection

An intrusion protection system (IPS) combines the analysis of an IDS with the added protection of a firewall. The IPS then must be configured with a network address, since it will be an active element within the network. It is able not only to alter received traffic but to generate traffic.

The IPS is capable of inspecting packets and altering packets, making them benign in the network. This allows rogue packets to continue in the network without sending failure responses back to the originators, alerting them that their attempt was not successful.

The IPS can be implemented as part of the IDS, or it can be implemented separately. In this type of implementation the IPS could receive instructions or data from the IDS or from a policy engine.

The IPS can also be integrated on a host with an application. Vendors are continually adding security features within their platforms to further enhance their applications. This adds another level of security, albeit at a slightly higher cost, since it must be implemented at all critical applications.

An IPS, like an IDS, must support the network protocols used within the network. This includes any vendor-proprietary protocols that are implemented on vendor platforms. It makes decisions based on policy, although in some cases intelligence can be added that allows the IPS to operate in a neural fashion, creating new profiles and policy based on historical traffic patterns. This requires storing traffic from many months for the most complete profile.

Many systems begin as passive IDSs and then later evolve into active IPSs. This is done by converting the passive probes to include active interfaces so that only those interfaces that are to be active would require a network address. This way, the passive capability can be maintained along with the active capability.