Under New Management: Practical Attacks on SNMPv3

Nigel Lawrence and Patrick Traynor
Georgia Tech Information Security Center (GTISC)
Georgia Institute of Technology
{*nlawrence@, traynor@cc.*} *gatech.edu*

Abstract

Network monitoring is a necessity for both reducing downtime and ensuring rapid response in the case of software or hardware failure. Unfortunately, one of the most widely used protocols for monitoring networks, the Simple Network Management Protocol (SNMPv3), does not offer an acceptable level of confidentiality or integrity for these services. In this paper, we demonstrate two attacks against the most current and secure version of the protocol with authentication and encryption enabled. In particular, we demonstrate that under reasonable conditions, we can read encrypted requests and forge messages between the network monitor and the hosts it observes. Such attacks are made possible by an insecure discovery mechanism, which allows an adversary capable of compromising a single network host to set the keys used by the security functions. Our attacks show that SNMPv3 places too much trust on the underlying network, and that this misplaced trust introduces vulnerabilities that can be exploited.

1 Introduction

Managing large networks can be a daunting task. Such systems regularly contain thousands of devices, ranging from traditional desktop computers and servers to switches, printers and IP-enabled appliances. Ensuring that all such devices remain responsive and that they perform their assigned duties requires significant resources from the network operator. Fortunately, tools and protocols such as the Simple Network Management Protocol (SNMP) exist to assist in this process.

While a number of features associated with SNMP have changed since its initial standardization[12], the most important revisions in the current release of this protocol (SNMPv3) focus on security. Requests to view status and change settings can now be both authenticated and made confidential, reducing the attack surface within a network. While the individual constructions used to provide these security guarantees are well understood (e.g., HMAC), the overall security of the protocol itself has not been evaluated. Accordingly, we are left with the following question: *Does SNMPv3 achieve the confidentiality and authenticity guarantees that it aims to provide?*

In this paper, we demonstrate that SNMPv3 fails to provide its advertised security guarantees. First, we demonstrate that the contents of encrypted messages to *any* host in the network can be recovered through the compromise of only a single machine. Second, we then demonstrate that spoofed messages that pass all authentication checks can be injected for *any* host in the network using the same compromised platform. In some cases checks can also be redirected to other hosts *without compromising a host*. The vulnerabilities we demonstrate are implementation-agnostic, and demonstrate a fundamental flaw in the current protocol. This flaw occurs in the discovery mechanism used in the User-based Security Model for SNMPv3. Discovery is primarily used to exchange identifiers and timing information between agents. Unfortunately, it also partially determines the encryption and authentication keys used for SNMP `GetRequests` and `SetRequests`. As discovery messages are sent unencrypted and unauthenticated, this allows a MITM to manipulate the keys used to protect the integrity and confidentiality of the SNMP messages. Because the discovery mechanism is itself vulnerable, it can be manipulated to allow an attacker to select the encryption and authentication keys used by the protocol. Successfully executed, such attacks could potentially allow an adversary to reveal information about devices within the network, as well as to potentially modify device behavior. For instance, on a UPS it may be possible to disable the audible alarms, modify the nominal input/output voltages and frequencies, or shut it down remotely [11]. Other devices such as switches may allow modification of security settings which include: disabling protection

from unicast flooding, disabling port security, or changing the list of secure MAC addresses [2].

We implement and demonstrate both of our attacks in a network using Nagios [3] and Net-SNMP[4], one of the most widely used implementations of SNMPv3. We then discuss considerations to make such attacks successful and to avoid detection. Finally, we discuss potential mitigation for this threat including changes to the protocol itself.

2 SNMPv3

Networks today are often large and complex, and maintaining the devices on those networks is a considerable challenge. Network adminstrators are often tasked with monitoring and maintaining a wide variety of devices on their network (e.g., servers, routers), and an increasing of these devices have extremely limited or entirely lack onboard user-interfaces (e.g., HVAC controls, PDUs, sensors, etc). The SNMP protocol solves several problems for administrators. For instance, it allows them to configure and monitor devices that may otherwise be difficult to access. For many devices, configuration must be done via SNMP, the serial port, or a web interface. Of these options, only SNMP allows for scalable configuration management accross a diverse group of devices. For example, a managed LAN switch can be configured with features such as port specific Quality of Service (QoS) and lists of authorized MAC addresses through SNMP requests. An adminstrator can then verify or modify the configuration of *all* of their managed switches through a single application. Accordingly, SNMP is found in virtually every large network as a matter of necessity.

In this section, we give a brief overview of the technical details required to understand the weaknesses in SNMPv3.

SNMP Messages: SNMP is a protocol used to monitor networked devices. These devices often include printers, routers, switches, servers, air conditioners, power distribution units (PDUs), temperature sensors, and many other devices. Monitored devices run an SNMP agent which typically communicates with a manager.

There are two primary types of requests, `GetRequests` and `SetRequests`. `GetRequests` can be used by a manager to poll agents. `SetRequests` are used by the manager to change the values of Object Identifiers (OIDs) on managed devices.

Discovery: Discovery is the process by which SNMPv3 agents learn the `snmpEngineID` of another agent and synchronize their clocks. The `snmpEngineID` is a unique identifier for SNMP agents. Because it is required to perform authentication and encryption, discovery occurs before the sending of authenticated requests. Discovery has two parts, both of which occur without authentication or encryption. In the first, a request is sent to an SNMP agent to request the agent's `snmpEngineID`. Upon receipt of the request, the a response is sent containing that agent's `snmpEngineID`. Because the discovery process is completely unprotected *the received* `snmpEngineID` *can not be trusted*.

Security: In order to provide integrity and confidentiality, SNMPv3's User-based Security Model (USM) allows for several different security levels depending on the user's needs. We focus specifically on the `authPriv` security level which requires the use of both authentication and encryption [9].

SNMPv3 provides message integrity/authentication by using an MD5 or SHA-1 HMAC of the `snmpEngineID` using the password as the key. The resulting HMAC is then used as a localized key for both authentication and encryption[9]. SNMPv3 localized keys allow each host to use different encryption/authentication keys even if they are configured with the same password. The localized key is then used to create a keyed hash of the whole packet, which is verified upon receipt of the message.

SNMPv3 packets are encrypted using either the Data Encryption Standard (DES) or the Advanced Encryption Standard (AES) with the aforementioned HMAC used as the key[9, 8]. The SNMPv3 request/response field containing the request ID, request type, and requested OID is encrypted in each message; both usernames and `snmpEngineIDs` are left as plaintext.

Key Management: Because every agent has its own localized key, agents must decide which key to use when sending messages. Localized keys are generated based on the combination of password and `snmpEngineID`. Because most SNMP agents only respond to queries, the responding agents' keys are almost always used for secure communication. This will be the case in all of our examples. This allows most agents to communicate without knowing the password used to generate their key. The small number of agents that generate requests are typically configured to send a request to a given IP address, and to use a specific password for authentication and encryption. The requesting agent uses discovery to retreive the `snmpEngineID` associated with a given IP address, and then generates the keys it will use to communicate securely.

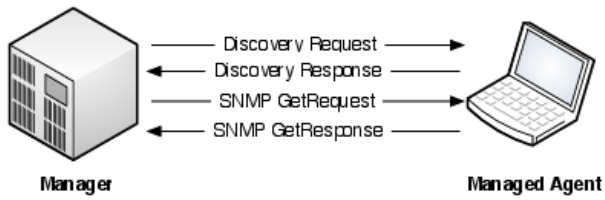

Figure 1: An example of an SNMP `GetRequest`. A manager sends a discovery message to the intended device and receives a response indicating the `snmpEngineID` associated with that device. A `GetRequest` is then sent and the requested value is returned.

3 Vulnerabilities

The attacks we demonstrate highlight two main issues. The first is that the discovery messages used to negotiate the authentication and encryption keys are neither authenticated nor encrypted. This means that they can be modified without detection, and an adversary can choose which localized keys are used. The second issue is that communication between agents does not use strong authentication. Therefore a manager can never be sure that the managed agent it is communicating with resides on a given host. These two issues give rise to a variety of different attacks. The attacks we explore in this paper are not problems with individual implementations of SNMPv3, but rather *with the protocol itself*.

3.1 Reading Encrypted Requests for Hosts

Encrypted requests for other hosts can be read using a single compromised localized key. There are many ways in which a localized key may become compromised. An adversary can obtain a key by compromising a host running an SNMP agent, because these keys are typically stored in plaintext. Alternately, if DES is being used, a brute-force attack could be used to compromise a key given sufficient time[16]. Even if a single key is compromised, the key does not provide an easy way of inferring other keys or the password used to generate those keys.

Once an adversary possesses a compromised key, they will force the manager's requests to use it. This is achieved by modifying or forging discovery messages between the manager and a managed agent. Because these messages are completely unprotected, an adversary can modify them at will. Messages are modified by replacing the `snmpEngineId` in the discovery response with the `snmpEngineId` associated with the compromised key. Upon receiving the forged discovery response the manager does not perform verification of the `snmpEngineId` but instead simply accepts it as correct. The manager then associates the forged `snmpEngineId` with the managed agent.

Because the `snmpEngineId` is directly tied to a localized key, the forged `snmpEngineId` will force the manager to use a specific key. This allows an adversary to choose which key is used by the manager. By inserting the `snmpEngineId` corresponding to a known key, *an adversary can read the contents of the manager's requests*. Responses can not be read in such a manner because managed agents will reject requests that do not use their localized key.

3.2 Spoofing Requests with a Helper

Spoofing responses requires compromising neither a host nor a key. It instead relies on the weak authentication and breaking a fundamental assumption of the protocol. These vulnerabilities allow communication to be redirected such that the manager will believe it is communicating with one managed agent when in fact it is communicating with another.

As mentioned previously, the reliance of a manager on the discovery protocol can be problematic. Because the manager does not otherwise know which `snmpEngineId` is associated with an agent, an adversary can choose which keys are used for communication. To spoof requests, an adversary would choose the localized key corresponding to the host they are using as a "helper". The helper is a host the adversary will use to spoof responses to the manager.

Another weakness is in the authentication of packets by the end hosts. Any time a request or response is received, the only authentication that takes place is verifying that the message was encrypted and authenticated with the right pair of keys. If secure communication relies solely on using the correct keys, and an adversary can choose which keys are used to create a message, then an adversary can manipulate a helper to forge responses for another host.

The final vulnerability is the assumption that hosts are tied to a network address. Because an adversary can force messages to be encoded for a particular host, all they need to do is to find a way to get the helper to accept the messages. One way to do this is to find a host using the Dynamic Host Control Protocol (DHCP) to act as the helper. Since DHCP can be spoofed, a host can be made to take on an IP address of the adversary's choosing. By modifying a host's IP address and spoofing discovery messages, *an adversary can force a host to masquerade as any other host* (provided they are both configured with the same user name).

Figure 2: A `GetRequest` is forced to use a compromised key. This allows an attacker to successfully read the request

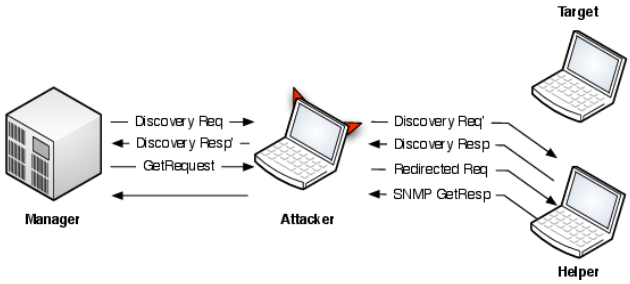

Figure 3: A `GetRequest` intended for the target is redirected to the helper.

4 Exploiting SNMPv3

4.1 Hosts

We used four virtual machines (VMs) set up on a virtual network to demonstrate the attacks. Of these four machines, one is the *manager* which runs Nagios and a DHCP server, one is the *adversary* (the MITM), and two are managed hosts. Both managed hosts are configured using DHCP. Of these managed hosts, we designate one as the *target* of our attack, and one as the *helper*. The helper will unknowingly aid the attacker in spoofing responses to the manager.

4.2 Checks

To demonstrate the exploit we schedule two checks on Nagios. These checks get the hostname of both the target and the helper using the "check_snmp" plugin. The target's hostname check is the one we subvert. Our goal is to show that an adversary can cause checks intended for the target to be encrypted with the helper's key. We then show that we can spoof the result of the target's hostname check by using the helper. This causes Nagios to return the helper's hostname for both checks at the same time.

4.3 Reading SNMP Requests

An adversary who can read encrypted SNMP requests poses a threat because the requests contain sensitive information. For instance, these messages could potentially tell an adversary which devices are performing which duties (e.g., IDS). This will help them avoid attracting attention while attempting to exploit services or compromise machines. Both `GetRequests` and `SetRequests` also include identifiers that may reveal sensitive information about devices. This is problematic in that a device's purpose, its manufacturer, and potentially its model may all be determined from these requests. In the case of a `SetRequest` an adversary will have access to both the identifiers used in the message and the values that would have been set (assuming the adversary had not tampered with the request).

To capture requests intended for the target, the adversary's VM is set up to act as a network bridge. Packets are then captured and inspected using netfilter. netfilter[5] allows iptables to capture and queue packets for processing in user space. We chose to use netfilter because it allows us to use iptables rules to specify which packets are queued for processing.

After capturing SNMPv3 discovery packets with netfilter we forward them if they are bound for managed hosts. We only modify discovery packets that are en route to the manager. For these packets, we replace the `snmpAuthoritativeEngineId` of the target with the one corresponding to the compromised key, and recalculate the checksums before allowing the packet through. This ensures that the time values in the packet are correct, and that the manager will accept it.

Upon receipt of the modified discovery response, the manager begins using the compromised key to encode `GetRequests`. The packets encoded with the compromised key are not passed on to the managed hosts, but instead are written to a pcap file for later analysis. This prevents the target's SNMP agent from seeing the request and realizing it was encoded with the wrong key. After the packets were written to a file, we analyzed the packets using Wireshark. Unfortunately Wireshark does not currently allow users to directly enter the localized key. Instead they are required to enter the passphrase and `snmpEngineId` in order for it to decrypt the stored packets.

4.4 Spoofing SNMP Agent Responses

An adversary with the ability to spoof SNMP agent responses has the power to falsify messages and conceal malicious activity. By redirecting messages intended for the target to the helper, an attacker can effectively hide activity that would normally attract attention. If an attacker has an exploit to crash a company's webservers, they may want to hide their activity when using the exploit. One way to do this is by redirecting checks from production webservers to other webservers such as those

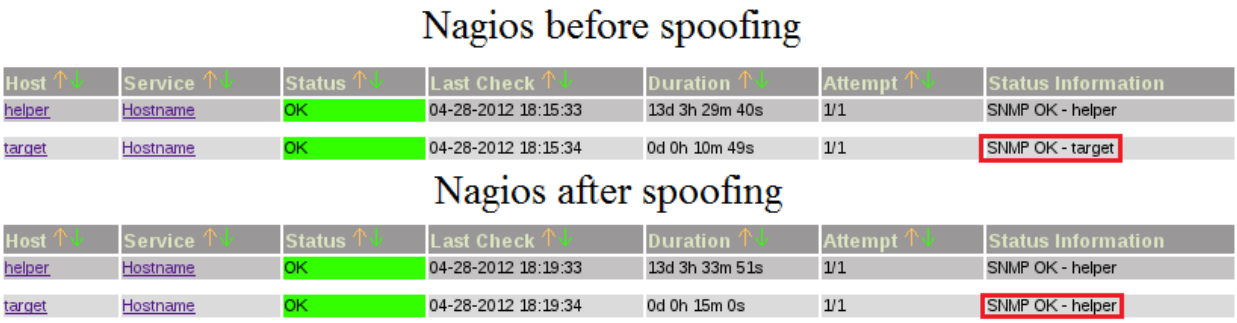

Figure 4: Initially, Nagios correctly reports the hostname for both hosts. We then use the helper to spoof SNMPv3 responses causing the target's hostname to change from "target" to "helper" (See "Status Information" column).

running the company intranet. Redirecting those checks might allow an adversary to take down the production webservers while having them appear normal to a network monitoring application like Nagios.

Spoofing response messages using the helper is done in a similar manner to the first attack. We configure the adversary's VM as a network bridge and use netfilter to modify and capture packets as was done previously. In addition to modifying discovery packets, we will forward the captured encrypted requests in such a way that the helper will respond to them.

To prevent the target from responding to queries, we drop DHCP traffic bound to or from its Media Access Control (MAC) address. This means that when its lease expires it is unable to renew it.

The helper's DHCP requests are modified to make it rotate between IP addresses rapidly. By setting the DHCP lease time to be very short (on the order of ten seconds or less), we can cause the helper to rapidly cycle between the target's IP address and the helper's original IP address. When the helper attempts to renew its lease (usually when half of the lease interval has elapsed), we change its IP address to the one that was not in use.

Because the helper's IP address is being changed so rapidly, in many cases the Address Resolution Protocol (ARP) tables of other hosts will not be up to date. To ensure the consistency of other host's ARP tables, gratuitous ARP messages are issued for both the target's and the helper's IP addresses and the helper's MAC address. This is done every time the helper's DHCP lease is renewed.

Now that the helper cycles between the two IP addresses, the attacker can use it to answer requests encoded with the helper's key. Packets destined for the helper's current address are forwarded normally, where as packets for the other IP address are queued up for later. Whenever the helper's IP address changes, the cache is then flushed by sending the stored packets. As long as the responses occur before the requests time out, they will be indistinguishable to Nagios. In this case, the default timeout value for the Nagios plugin's requests is ten seconds.

SNMP discovery packets have to be handled slightly differently because the target is unreachable and the helper keeps cycling between IP addresses. This time discovery packets going in both directions are modified. The initial packet from the network monitor to the managed host is modified so that its IP address and MAC address match the helper. This simplifies spoofing because the helper does most of the work of responding to discovery messages. All that the adversary needs to do is make the discovery packet appear to have come from the correct host. Because the timeout value for discovery packets in the "check_snmp" plugin is relatively short by default (approximately three seconds), and because discovery packets can be modified at will, there is no reason to cache them.

As shown in Figure 4, the spoofing is difficult to detect. The differences between the spoofed check and the original are that the spoofed check will sometimes take longer to return, and the value it returns may differ from the "real" value. Because the check was performed using the helper's key, the adversary can not read the encrypted messages in transit.

While spoofing, both the helper and the target's hostname checks appear to be normal with the exception of the value they return. In this case, both the target's and the helper's checks return a hostname of "helper" because the helper is generating both responses.

5 Discussion

5.1 Implications of the Attacks

We showed that an attacker can force SNMPv3 to use a particular pair of keys for encryption and authentication. We also showed that the manager could not distinguish between agents with the same username, but with

Table 1: Examples of impact from successful attacks, broken down by device types with models running SNMP, capabilities enabled by SNMP control and the potential consequences of successfully executing the attack.

Device	Capability	Consequences
HVAC	Conceal errors, adjust temperature/humidity, power cycling	Physical damage
Managed Switches	Disable/modify authorization, disable/enable ports	DoS/network access
Power Distribution Unit	Modify voltage/current, low/high power threshold	DoS/physical damage
Perimeter Sensors	Door/motion sensors can be disabled or subverted	Conceal physical access
UPS	Modify voltage/current, power thresholds, power cycling	DoS/physical damage

different `snmpEngineIds`. This means that using a single compromised pair of localized keys, an attacker can forge responses for any request with the same username. Such an attacker is no longer limited to relying on a helper to spoof responses, because *they can spoof responses on their own*. In addition to attacks which subvert localized keys, we have shown that an adversary can deceive a manager *without needing to compromise a host or a key*.

The impact of these attacks depends on what SNMP is being used to manage. For instance, an attacker could use the first attack to potentially map a network by determining the location of the functions executed by devices throughout (e.g., IDS, etc). The second set of attacks are potentially more damaging. For instance, an adversary could reduce the reliability of certain systems by shutting off managed UPS devices[11].

If an adversary knows that a UPS will be shut down via SNMP, they may choose to redirect the shutdown request to another host. If there is another similarly configured UPS that can be used as a helper, then that host can be shut down by forcing it to receive the request as was done in the spoofing attack. Since SNMP is run over UDP, it is often considered unreliable and may send multiple messages if no acknowledgment is received. This would allow an adversary to potentially use a single request to shutdown multiple UPSes provided they acted quickly. Alternatively they could shut down both the target UPS and the helper UPS. After the helper UPS is shut down, an adversary could again launch a spoofing attack and use another host to hide the fact that a UPS is now offline.

The issue with the previous attack is that the adversary has to know about the request in advance to carry it out. As was mentioned previously, an adversary can not read requests while attempting to redirect them to a helper. An adversary may can read requests sent at other points in time if they have a single compromised key. Because SNMP requests are often sent periodically, an adversary may be able to predict when a request will arrive. Even in cases where requests are unpredictable, an adversary may be able to generate requests indirectly. If a device such as a UPS is being managed with SNMP, it can be managed remotely to maintain a particular state. An administrator may want the output voltage OID on a particular UPS to always be a specific value. If an adversary redirects a check for that UPS to a differently configured device, the management system may think the device is misconfigured and attempt to set that device's OID incorrectly. Such an attack would allow an adversary to change the devices configuration.

Many other types of devices can be exploited. An adversary could potentially modify settings on security appliances to [2] or even change the temperature settings in a server room [6]. In short, such an attacker could modify the behavior of any IP-enabled device on the network managed by SNMP.

These attacks may also extend to other SNMPv3 security models through a downgrade attack. Because other SNMPv3 secure transport models are recommended to fall back to USM "in times of network stress"[17], it may be possible to force other security models to use USM.

5.2 Difficulties for Attackers

There are a variety of different conditions which may make our attacks more difficult. We will address a variety of possible obstacles an adversary could face when carrying out an attack.

Uniqueness: Some hosts may have unique checks that are only run on that host. This could cause issues for an attacker attempting to spoof checks because the helper usually must be configured correctly to respond to a query. If it is not configured correctly it may not respond correctly. By having the helper attempt to answer the request, it could in fact return the wrong result and fail the check. This could draw suspicion to the target, something an attacker would likely want to avoid. Unique checks may be possible to detect by looking at and comparing the patterns of checks across devices. A unique check would likely appear to be an outlier. We leave such detection for future research.

Some checks may have different expected results on different hosts. An example of such a check is a temperature sensor. In one area 75 degrees Fahrenheit may be within the acceptable range, but in another area, tem-

peratures exceeding 70 degrees Fahrenheit may be considered abnormal. In such a case, spoofing a response with the warmer value could cause a check to fail for some hosts but not others. Attackers would likely wish to avoid this, but unlike the previous case this check will not be distinguishable in advance based on the pattern of checks. Also, because the *results* of a request are what an attacker would need to be able to read, the first attack would not be useful for guessing the expected value of the check.

An attacker could likely determine if a spoofed check was failing however. In many cases, Nagios and other similar systems are configured to perform multiple checks before determining if a check has failed. This is done to make the systems more robust. Because these retries are often scheduled with shorter intervals of time between them than successful checks, they may be distinguishable from normal checks. An attacker could potentially attempt to spoof checks one at a time and watch for retries. If an unexpected check occurs, they would suspect that their check most likely failed and might then stop spoofing that check. If done quickly enough, this probing might avoid detection by the network monitor. Alternatively, Nagios might only try once before failing. In that case a failed check may have a different retry interval than a successful check. Such a failure would be evident to the monitoring software, but would not likely draw much attention if it resolved itself quickly.

The probing methodology mentioned above could also be used by an attacker to spoof traffic without using a helper. In such a case, the attacker might not know what the expected response to a request is. They could likely try different values and then use Nagios' behavior to confirm their guesses.

Custom Checks: In some cases, there could be requests for unknown OIDs. If the attacker attempts to forge these checks without a helper, they may not know what kind of response is expected. In this case they can likely use the probing methodology previously described. The difficulty of guessing a correct value for such a check is likely to vary based on the type of check.

Other Checks: Even though an attacker may be able to subvert SNMP traffic, checks may be run using other protocols. An adversary can allow this traffic through to the target host if they are not using a helper to perform spoofing. If they are using a helper, it may not be feasible to forward the non-SNMP traffic to the helper. Either way, the adversary will still have to either work around or subvert those other checks. Being able to subvert SNMP does not necessarily mean that all monitoring can be defeated.

Spoofing Multiple Hosts: In some situations an attacker may wish to spoof multiple targets using a single helper. This should be possible, but there will be a limit to the number of hosts that can be spoofed simultaneously for a single helper. An adversary could likely spoof checks for additional hosts by increasing the number of helpers.

Unique Username: In certain, rare cases, there may be a username that is only used on a single host. In this case, the described attacks will be ineffective because there is only a single localized key.

Avoiding Detection: If someone were actively looking for these attacks, they would be relatively easy to detect. For all of these cases, it should likely appear suspicious when more than one IP address appears to be using the same `snmpEngineId`. Someone looking to detect this kind of attack can also attempt to detect when the `snmpEngineId` associated with an IP address changes, because `snmpEngineIds` change relatively rarely (usually only when a device is replaced or the software reinstalled). This kind of detection would applies to all of the attacks mentioned, including spoofing without a helper.

Another way to detect the attack is by looking at the ARP traffic generated. Having multiple IP addresses assigned to the same MAC address is not necessarily unusual, but for many devices with SNMP agents it will be.

For an adversary attempting to read encrypted requests, they can not forward requests sent with their key because the managed agent would be unable to answer the request. As was mentioned previously, dropping such requests prevents the managed agent from notifying the manager of an invalid key. This may also be noticed because it prevents the manager's queries from receiving a response. One solution to this may be to attempt to read messages probabilistically. Because SNMPv3 messages are usually sent via UDP, checks may be considered unreliable or allow multiple retries. By modifying requests infrequently, an adversary's activity may be assumed to be the result of network instability.

Devices are not Configured with DHCP: If no SNMP agents are being run on machines configured with DHCP, or if none of the machines that are using DHCP are suitable as a helper, the second attack will not be possible. It is hard to predict how often this will be the case. This will only hinder an adversary who is dependent on a helper.

Choosing a Helper: A potential drawback to using a helper is that changes to the helper's IP address might

cause problems for other services. An adversary would likely want to choose their helper carefully. A host that is relatively stable and infrequently used would be the best candidate. This host also would likely need to be able to respond to the same checks that the target does. Since an adversary using a helper presumably does not have a compromised key pair, they will have to use other methods to locate a similarly configured helper. It is still possible to determine a great deal of information about a potential helper by looking at its traffic, especially the pattern of SNMP requests being sent to it.

Even without being able to see what checks are being run, an attacker can tell the username that requests are being run as and when they occur. Because systems such as Nagios perform checks at regular intervals, and the interval lengths vary depending on the checks, an attacker may be able to infer which hosts are running similar checks by comparing the timing of the encrypted requests. Contributing to this effect is the fact that hosts are often constructed using templates which include similarly configured checks. By looking at these patterns of checks, an adversary may be able to construct a signature of sorts for each template. This could allow for identification of devices with similar purposes and similar checks. Developing signatures for similarly configured hosts based on the pattern of their checks is a topic left for future work.

5.3 Fixing the Vulnerability

The quickest way for administrators to prevent the spoofing attack with a helper is to ensure that none of the devices running SNMP agents are using DHCP. We understand this may not always be practical or necessary, as there are some cases in which the reliability of checks is not paramount.

Even if no agents are configured to use DHCP, administrators should be wary of allowing usernames to be used on machines with differing levels of security. As we showed earlier, compromise of any machine with that username compromises that username for all machines. In many cases localized keys are simply stored in a file on the device, so obtaining a key on a compromised host is not difficult. The only way to prevent compromised localized keys from being used maliciously is to prevent the network monitor from using the discovery process.

One way to protect the discovery process would be to use IPsec. IPsec protects the transport layer, meaning the discovery process will be protected automatically. Such an approach would make our attacks impossible.

Another solution would be to use the Transport Security Model (TSM) instead of USM to secure SNMP traffic. TSM protects the SNMP protocol at the transport layer through the use of protocols such as Transport Layer Security (TLS). One concern with TSM is that it is not considered to be as reliable as USM in the event of network failures. In order to alleviate this concern the TSM standard recommends falling back to USM in times of network stress[17]. If this recommendation is followed, a downgrade attack may very well be possible. The other concern with simply switching to TSM is that many agents do not support TSM; they only support USM. Because of the nature of the devices running these agents, a significant number of devices may never be updated to support TSM.

Because using IPsec or TSM is not always feasible, there are cases in which USM will continue to be used. In these cases it makes sense to fix the protocol itself. As long as the discovery process is relied upon in its current state, the mentioned vulnerabilities will remain. Discovery can not be removed altogether, because it may still be useful for helping to synchronize time values. However, discovery does not need to be trusted to retrieve `snmpEngineId`. One solution is to keep a list containing each `snmpEngineId` and its associated network address. Discovery can then be used solely for synchronizing clocks. This would remove any ambiguity about which host is being communicated with, and prevents an attacker from being able to choose the key. The list would have to be manually maintained, which would unfortunately put an additional burden on the administrator. There are other potential ways to modify discovery such that it is protected, but almost all such modifications would break compatibility with legacy devices. For legacy devices, this is one potential solution that not only will prevent all of the attacks mentioned, but would require no modification to the agents, and minimal changes to the protocol.

6 Related Work

To date, most research on SNMPv3 has been focused on its performance [14] and potential modifications to the protocol. There have been many studies testing alternate methods of transmitting SNMP traffic[15, 21, 18]. These studies tend to focus on measuring the performance of SNMP over other protocols such as Transport Layer Security (TLS) [15, 21], Datagram Transport Layer Security (DTLS) [21], Secure Shell (SSH) [21], and Internet Protocol Security (IPsec) [18]. The primary motivation is to improve performance without sacrificing security. While these studies are valuable, for the most part they have not been widely adopted. Additionally, none of these studies attempt to address the question, “Is the existing SNMPv3 protocol using USM secure?”.

Other studies have come closer to answering this question. Previous work has been published which examines SNMP vulnerabilities, but this work has mainly fo-

cused on previous versions of SNMP [23, 13, 20, 22]. Additionally, these papers are primarily concerned with implementation-dependent vulnerabilities. For instance, the Oulu University Secure Programming Group tested multiple SNMP implementations for errors when processing requests [22]. Another paper explored common vulnerabilities, many of which were due to configuration errors or are not applicable to SNMPv3 [13]. This research has been valuable, but it has also overlooked some of the inherent weaknesses in the protocol.

Several groups have been critical of the protocol itself, and attempted to improve it. One such early effort was the Application Secure SNMP (APSSNMP). APSSNMP was a protocol that was intended to replace SNMPv3 by being simpler and requiring less overhead [24]. APSSNMP unfortunately was not to be, and was later shown to be insecure [10]. Other schemes have been put forward which use public key cryptography [23, 20] or Diffie-Helman key exchange [19]. Perhaps part of the reason that these alternate schemes have not seen widespread adoption is that they have lacked a strong justification for adoption.

One major justification for replacing or modifying SNMPv3 would be weaknesses in the protocol. Perhaps the most similar research to our own has been the proposition of a theoretical denial of service (DoS) attack on wireless networks [23]. The attack uses a MITM, who de-synchronizes the clocks of both the SNMP agent and the manager to prevent communication [23]. One problem with this attack, is that it has yet to be demonstrated. Another is that SNMPv3 was never intended to prevent DoS attacks [9], and DoS attacks in general have been difficult to prevent traditionally. In our work, we present several attacks that violate the security goals of the protocol [9] and are realistic. Additionally, our attacks can breach the confidentiality and integrity of the SNMP requests.

While our work differs from previous attacks on SNMP, it is not without precedent. Our attack on the discovery process (which defeats key localization) means that all agents with the same username are effectively using a single shared secret similar to Wired Equivalent Privacy (WEP) [1]. Because of this, all agents using the same username are indistinguishable to the manager. This allows any valid host to masquerade as another similarly configured host. Our attack is based on the protocol's inherent trust in the network, and makes use of several known attacks such as DNS and ARP spoofing [7].

7 Conclusion

The current protection afforded by SNMPv3 is not satisfactory. We have demonstrated that under reasonable conditions both the confidentiality and authenticity of messages can be violated. We have explained how, even with the use of strong cryptography, invalid assumptions can cause the protocol to fail. In SNMPv3, the protocol relied on the fact that messages could not be modified to protect the communication against redirection. This failed to take into consideration that an adversary could change an agent's IP address at will in some cases. It also relied on an unprotected mechanism to determine, identity and to choose which key pair to use. We have shown how this can be used by an adversary to force the manager into using a specific key pair.

We have explored how these vulnerabilities can be used by an adversary to hide sabotage done to web servers, backup servers, and other vital services. Because of the ubiquity of this protocol for use with embedded devices, these weaknesses may even have cyber-physical implications. Although these vulnerabilities are problematic, they can be overcome. We have presented a way to fix these issues with minimal changes to the agents or the protocol.

In the future we hope to do research into identifying hosts configured with similar checks based on the patterns of checks being run on them. The ability to do so would allow adversaries to gain knowledge about these systems *without* needing to read the contents of the requests or modify traffic.

Acknowledgments

This work was supported in part by the US National Science Foundation (CAREER CNS-0952959). Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. We would like to thank Chaitrali Amrutkar and Hank Carter for their valuable feedback.

References

[1] IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*, 2003.

[2] Cisco-port-security-mib, May 2012. ftp.cisco.com/pub/mibs/v2/CISCO-PORT-SECURITY-MIB.my.

[3] Nagios: The industry standard in it infrastructure monitoring, May 2012. www.nagios.org.

[4] Net-snmp, May 2012. www.net-snmp.sourceforge.net.

[5] netfilter: firewalling, nat, and packet mangling for linux, May 2012. www.netfilter.org.

[6] Poseidon 3468: IP Thermostat with 230V relays, 2012. http://www.hw-group.com/products/poseidon/poseidon_3468_en.html.

[7] S. Bellovin. A look back at "security problems in the tcp/ip protocol suite. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 229 – 249, dec. 2004.

[8] U. Blumenthal, F. Maino, and K. McCloghrie. The advanced encryption standard (aes) cipher algorithm in the snmp user-based security model. *Internet proposed standard RFC*, 3826, 2004.

[9] U. Blumenthal and B. Wijnen. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). RFC 3414 (Standard), Dec. 2002. Updated by RFC 5590.

[10] R. C.-W. and Phan. Cryptanalysis of the application secure alternative to snmp (apssnmp). *Computer Standards & Interfaces*, 31(1):63 – 65, 2009.

[11] J. Case. Ups management information base. *Internet standard RFC*, 1628, 1994.

[12] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A simple network management protocol. *Internet standard RFC*, 1067, 1988.

[13] P. Chatzimisios. Security issues and vuluerabilities of the snmp protocol. In *Electrical and Electronics Engineering, 2004. (ICEEE). 1st International Conference on*, pages 74 – 77, june 2004.

[14] A. Corrente and L. Tura. Security performance analysis of snmpv3 with respect to snmpv2c. In *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, volume 1, pages 729 –742 Vol.1, april 2004.

[15] X. Du, M. Shayman, and M. Rozenblit. Implementation and performance analysis of snmp on a tls/tcp base. In *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, pages 453 –466, 2001.

[16] T. Guneysu, T. Kasper, M. Novotny, C. Paar, and A. Rupp. Cryptanalysis with copacobana. *Computers, IEEE Transactions on*, 57(11):1498 –1513, nov. 2008.

[17] D. Harrington and J. Schoenwaelder. Transport subsystem for the simple network management protocol (snmp). *Draft standard RFC*, 5590, 2009.

[18] H. Hia and S. Midkiff. Securing snmp across backbone networks. In *Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on*, pages 190 –196, 2001.

[19] H. Oh and S.-H. Jin. A simple snmp authentication method for ad-hoc networks. In *Convergence and Hybrid Information Technology, 2008. ICCIT '08. Third International Conference on*, volume 2, pages 555 –558, nov. 2008.

[20] H. Otrok, A. Mourad, M. Debbabi, and C. Assi. Improving the security of snmp in wireless networks. In *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, volume 1, pages 198 – 202 vol.1, june 2005.

[21] J. Schonwalder and V. Marinov. On the impact of security protocols on the performance of snmp. *Network and Service Management, IEEE Transactions on*, 8(1):52 – 64, march 2011.

[22] University of Oulu Secure Programming Group. Protos test-suite: c06-snmpv1.

[23] T. C. Wan, A. Goh, C. K. Ng, and G. S. Poh. Integrating public key cryptography into the simple network management protocol (snmp) framework. In *TENCON 2000. Proceedings*, volume 3, pages 271 –276 vol.3, 2000.

[24] C. M. Wee, M. Salim Beg, and B. Vaillant. Apssnmp as a protocol for managing network appliances. In *Networked Appliances, 2002. Gaithersburg. Proceedings. 2002 IEEE 4th International Workshop on*, pages 87 –96, 2002.