# VULNERABILITIES IN SNMPV3

A Thesis
Presented to
The Academic Faculty

by

Nigel R. Lawrence

In Partial Fulfillment
of the Requirements for the Degree
Masters of Computer Science in the
School of Computer Science

Georgia Institute of Technology
August 2012

# VULNERABILITIES IN SNMPV3

Approved by:

Professor Patrick Traynor, Advisor
School of Computer Science
*Georgia Institute of Technology*

Professor Mustaque Ahamad
School of Computer Science
*Georgia Institute of Technology*

Professor Jon Giffin
School of Computer Science
*Georgia Institute of Technology*

Date Approved: 1 July 2010

*To my family,*

*you have always been there for me.*

# PREFACE

I first became interested in the SNMP protocol while working for the Georgia Tech Research Institute. As a co-op student, one of the responsibilities that I took on was monitoring and maintaining the network. In the course of doing so, I spent much of my time configuring Network Management Software (NMS).

While working with our NMS, I quickly realized the flexibility afforded by extending our SNMP agents. By extending agents, I could run almost any check I wanted by using the agent to execute scripts. What dawned on me very quickly, however, was that to use this functionality, I absolutely needed to be using SNMP in a secure manner. This was what initially sparked my interest in the protocol.

In the course of learning to *use* SNMPv3, I found that I was learning a great deal about the protocol. This was partially a result of my curiosity, and also partially due to inadequate documentation.

After returning to the Georgia Institute of Technology as a graduate student, I was fortunate enough to take a Network Security class with Professor Traynor. The project for this class provided me with both the opportunity and the motivation to really learn the details of the protocol. After many hours of reading through RFCs, I found several issues with the protocol which were the subject of my project, and which eventually became this thesis and a paper.

# ACKNOWLEDGEMENTS

I would like to thank my committee members for taking the time to help me prepare my thesis, and in particular Patrick Traynor for helping to guide and edit the accompanying paper.

I would also like to thank GTRI and especially Steven Hurst for supporting me as a GRA. Without their assistance, I would not have been able to produce this thesis.

# TABLE OF CONTENTS

# LIST OF FIGURES

# SUMMARY

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.

# CHAPTER I

# 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 [14].

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.

## *1.1 Background*

### 1.1.1 ARP

The Address Resolution Protocol (ARP) is a protocol commonly used to allow hosts to resolve a network address to a link layer address.A common example of this is converting an IP address into a MAC (Media Access Control) address. The ARP standard is defined in RFC 826 [30].

The ARP protocol is primarily carried out through a request and response mechanism. Before sending a packet, a host checks its ARP table (which contains a mapping from network addresses to link layer addresses) for a corresponding entry. If no such entry is found for the network address, it will issue an ARP request. These queries are sent to the broadcast address of the subnet, and causes any host using that network address to send an ARP response. This response contains the link layer address of the host with the matching network address. Upon receiving this response, the requester will add an entry to its ARP table for that response. Each entry in the ARP table will also have an associated timeout value. This timeout value determines how long a host should wait before deleting an entry in the table. The timer used to time out entries is usually reset to zero when the host observes network activity associated with a given entry.

ARP also has several other functions, but the most relevant to our work are announcements. ARP announcements, often known as gratuitous ARP messages, are messages sent solely to update nearby hosts. They are not intended to garner a response, but instead cause hosts who hear the announcements to update their ARP tables in response to the announcement. This allows hosts who have recently changed their IP address to update the ARP table entries of other hosts, preventing network problems that might otherwise arise.

ARP spoofing is the process of falsifying ARP messages on a network. Because ARP traffic is not authenticated, it can easily forged by an adversary. ARP spoofing is most often used to allow an attacker to redirect traffic intended for another machine to the attacker's machine. This traffic can then be forwarded from the attacker to the legitimate host, allowing the attacker to act as a man-in-the-middle. This type of attack is often referred to as ARP poisoning.

### 1.1.2 DHCP

The Dynamic Host Control Protocol (DHCP) is used to automate the distribution of network configuration details such as: obtaining an IP address, learning the address of the local gateway, and learning the address of DNS servers. DHCP is an extension of the BOOTP protocol, and helps to bootstrap hosts so that they can access networks without manual configuration. The current DHCP standard is defined in RFC 2131[17].

When a device is configured for DHCP and connects to a network, it initially undergo a process known as discovery. During discovery, a host sends `DHCPDISCOVERY` messages to the broadcast address, looking for a DHCP server. When a server or servers see these packets, they respond by sending a `DHCPOFFER` message to the initiating client. The client device may receive multiple offers, but will select one offer, and send a `DHCPREQUEST` message to the corresponding server. This server then allocates the address, and returns a `DHCPACK` to confirm the assignment.

Because network addresses are not allocated indefinitely in DHCP, each allocation is provided for a set period of time. This portion of time for which an address is allocated is known as a lease, and each lease has a maximum duration. The expiration of a lease allows the IP address to be reused by the same client, or other future clients. When a client's lease expires, the client may skip the discovery phase and instead send a request directly to its previous DHCP server. If it receives an acknowledgment, its lease has been renewed and it may continue using its previously allocated address. It is not uncommon for hosts to attempt to renew their address in such a manner once half of the lease duration has expired.

Like ARP, DHCP traffic is unauthenticated and easily forged DHCP spoofing is an attack in which the adversary impersonates a DHCP server. Spoofing allows an attacker to change the IP addresses of hosts on the network, or to prevent hosts from being assigned IP addresses in a denial-of-service attack.

### 1.1.3 Bridging

Bridging is a term that refers to forwarding packets on a packet-switched network. Bridges that exist solely at layer two may be referred to as transparent. Transparent bridging works by examining and forwarding packets based on information stored in a forwarding table. Each entry in the forwarding table contains a link layer address and a physical port. This port is used to send traffic to the associated address.

Forwarding tables are initially empty, but in most bridges are filled as traffic is observed through a process known as learning. Whenever traffic flows through the switch from an unknown source address, an entry is added for the unknown address and the port its traffic arrived from. When traffic is destined for an unknown host, the bridge usually floods the traffic to all of its ports to ensure delivery. Because bridges are layer two devices, they are completely invisible to higher layers.

## *1.2 Overview of SNMP*

SNMPv3 introduces many new security features over previous versions. These features are intended to protect against a variety of threats including message modification, message re-ordering, delaying of messages, replaying of messages, and eavesdropping on messages between agents [9]. There are additional protections to prevent an attacker from impersonating an authorized user, from reading traffic, from replaying traffic or reordering traffic, and to ensure that users have authorization to perform certain actions. These protections can be divided into four primary mechanisms: the Hash-based Message Authentication Code (HMAC), the encryption algorithms, the View-based Access Control Model (VACM), and the mechanisms that are used to ensure timeliness and correct ordering of messages. This section provides information on how the SNMPv3 protocol operates, and explains several flaws in its operation. Additional details on the security mechanisms of the protocol can be found in RFC 3414 [9].

### 1.2.1 SNMP

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.

Another common SNMP mechanism is a trap. Traps are unsolicited SNMP messages which are usually sent in response to an event on the agent. An example would be a printer sending a notification when its toner is depleted.

### 1.2.2 Security Levels

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].

### 1.2.3 Message Integrity

In order to guarantee the authenticity of a message, both the data origin and the integrity of the message must be preserved. In SNMPv3 the way this is done is with an HMAC using either MD5 or SHA-1 as the hashing algorithm [9]. The secret key used with the HMAC is referred to as the `authKey`. Each username should have multiple `authKeys` which are localized for use with different agents, but were all created using the same passphrase[9]. The expectation is that a single username and password may be used on multiple agents residing on different devices. This allows multiple devices to use the same base password, but prevents compromise of that password from affecting other machines configured with the same username. Keys

are localized by hashing the authKey with the snmpEngineID to create a key that is unique to each SNMP engine [9]. The `snmpEngineID` is a unique identifier for each SNMP engine within an administrative domain. Without access to the original `authKey`, an attacker can not construct a localized key, and without a valid localized authKey an attacker can not generate the correct digest for any messages sent to or from an SNMP agent. Additionally, since the entire message is used to generate the HMAC, an attacker cannot modify any field in the message without changing and thus invalidating the digest. Any messages received with an invalid HMAC and a security level of `authNoPriv` or `authPriv` cause an authentication error [9].

### 1.2.4 Message Privacy

Inside SNMPv3 packets, there is a `scopedPDU` which contains the actual information for requests. This `scopedPDU` is the portion of the packet that is encrypted when privacy is desired [9]. The SNMPv3 standard currently supports encryption with the Data Encryption Standard (DES), but there is a proposed standard which adds support for the Advanced Encryption Standard (AES) [9, 8]. The key used for encryption is referred to as the `privKey`, and is localized in a similar manner to the `authKey`. It is worth noting that the only information encrypted is the request or response itself.

### 1.2.5 Authorization

Before a request can go through, it must be authorized to do so by an Access Control Model. In SNMPv3, one such model is the VACM [42]. The VACM determines whether a type of access (read, write, or notify) is allowed to a certain OID. Access is allowed or denied based on factors such as the type of access that is being requested (read,write,notify), the username requesting it, and the security level of the request (`noAuthNoPriv`, `authNoPriv`, `authPriv`).

### 1.2.6 Authoritative and Non-authoritative SNMP Engines

One important distinction is that of authoritative and non-authoritative SNMP engines. For a message that expects a response, the sender is non-authoritative and the receiver is authoritative. For a message that does not expect a response, the sender is authoritative and the receiver is non-authoritative. The authoritative host's `snmpEngineID` is used to determine which pair of localized keys will be used for communication [9].

### 1.2.7 Message Replaying, Reordering, and Delaying

To prevent replay attacks, message reordering, and delaying of messages, each message includes timeliness indicators. These indicators of timeliness are the `snmpEngineBoots` and `snmpEngineTime` values. The `snmpEngineBoots` value indicates the number of times that the agent has re-booted or re-initialized since the `snmpEngineID` was configured. The value of `snmpEngineTime` indicates the number of seconds since `snmpEngineBoots` last changed [9]. The values that are used for communication are the snmpEngineID and snmpEngineTime of the authoritative agent [9]. These values are kept loosely synchronized on the non-authoritative agent, and they are resynchronized every time communication occurs. SNMP agents will reject messages that are outside of their time window (if they are off by plus or minus 150 seconds) if they are authoritative[9]. If they are the non-authoritative agent, then they will only reject older messages and resynchronize upon receipt of newer messages. These mechanisms do not ensure that messages arrive in order within the time window, so there are several additional measures that are taken to prevent SNMP set operations from arriving out of order (most other SNMP operations may arrive out of order if they satisfy the timeliness requirements).

One last method used to protect against replay attacks, is matching incoming responses to previously sent requests. How to do this is not explicitly defined, but

the protocol requires a method to discard responses that are not associated with an outstanding request [9]. The protocol mentions a common way to do this, which involves using a unique `msgID` per request, this `msgID` is included in the packet, and having the request and response share the same `msgID` allows requests to be tied to responses.

### 1.2.8 Discovery

The last feature of SNMPv3 that needs to be mentioned is that of discovery. Discovery is the process by which non-authoritative SNMP engines learn about the `snmpEngineID` of the authoritative engine. For authenticated communication, the discovery process also learns the values of the authoritative engine's `snmpEngineBoots` and `snmpEngineTime` [9]. Of particular importance is the fact that both the request and the response messages used for discovery are sent with the security level `noAuthNoPriv`, meaning *they offer no guarantee of confidentiality or integrity.*

### 1.2.9 Unreliable Transport

SNMP messages are traditionally sent over UDP[31], which may cause messages to arrive out of order or not at all. Get and set requests are confirmed with a response, so even though they are unreliable the manager knows a request has been received when it gets a response back. For unacknowledged messages such as traps, however, there is no response[13]. This means that traps can fail to arrive at the manager, but neither the agent or the manager will know that a message was lost.

# CHAPTER II

# PREVIOUS WORK

The SNMP has seen widespread use since its creation in the late 1980s. SNMPv3, however, is a more recent development. Some research suggests that even though SNMPv3 is the current standard, and SNMPv1 and SNMPv2c have been declared as *historic*, the older and less secure versions may still be the most widely used [35]. While a significant body of research exists for SNMPv3, it tends to be focused in several areas.

## *2.1 Performance Tradeoffs*

Much of the research on the SNMPv3 protocol has been focused on performance trade-offs between SNMPv3 and SNMP version 2c (v2c) [16], or between SNMPv3 using USM and SNMPv3 over some other protocol. For instance, research has compared SNMPv3's performance to SNMPv2c used over other protocols such as Transport Layer Security (TLS) [18, 33], Datagram Transport Layer Security (DTLS) [33], Secure Shell (SSH) [33], and Internet Protocol Security (IPSEC) [23]. While these studies are important, they do not focus on the security of various schemes so much as they focus on network performance and the overhead associated with the chosen encryption schemes.

## *2.2 Network Obstacles*

Other research on SNMP has focused on challenges in the network, such as getting past firewalls [32] by using a form of peer-to-peer network. Another study looked at the challenge of monitoring devices behind NATs [29]. Both of these studies are interesting, but the focus is more on adding features to SNMP than looking at its

security or modifying the existing security scheme.

## 2.3 Alternate Security Schemes

Another common research area has been alternative ways to protect SNMP, either by using a different protocol, or by making modifications to the current scheme, SNMPv3. One such protocol was the Application Secure SNMP (APSSNMP) [41] scheme. Because APSSNMP was later shown to be insecure [10], it has not been widely deployed. Other schemes using public key cryptography [39, 28] or Diffie-Helman key exchange [27] have been put forward, but never widely adopted. Perhaps part of the issue with these schemes is that they do not adequately justify *why* the security of SNMPv3, as it exists currently, is inadequate.

## 2.4 Known Vulnerabilities

There are few known security vulnerabilities in the SNMPv3 protocol. Several papers have mentioned vulnerabilities, but for the most part these are not issues with the protocol itself. In one of the papers on adding public key cryptography to SNMP the authors fail to define specific vulnerabilities with the protocol, but instead state that for the normal SNMPv3 scheme, the User-based Security Model (USM), "it is not sufficient in a distributed Internet environment where malicious users can perform a variety of attacks (e.g. via username and password interception, dictionary attacks, session hijacking, etc) to gain unauthorized access to SNMP enabled resources" [39]. They later make the claim that "Modification of Information, in the form of intercepted messages via "Man in the Middle" attacks, is an inherent weakness in USM due to the use of shared symmetric authentication and privacy keys." This is incorrect, because no information can be modified in an SNMPv3 packet due to the fact that the authentication mechanism (which ensures integrity) takes in the entire packet "as received on the wire"[9]. Because USM uses a secure HMAC scheme, the entire message is protected from changes by an adversary unless they know the symmetric

key. This would protect the scheme against a traditional MITM attack.

In another paper on adding public key cryptography to SNMP, the authors mention two specific threats[28]. The first of these threats is a proposed MITM attack that works by desynchronizing the clocks between the agent and manager so that the manager and the agent can no longer communicate [28]. The reason we do not consider this attack to be significant is that a MITM can almost always choose to drop traffic intended for a host because the traffic going to that host must pass through the MITM first. This would achieve a similar effect, is simpler, and is very difficult to protect against. The other attack involves preventing the targeted host from sending traps for failed authentication [28]. This too can be done by a MITM by simply dropping the traffic. The argument presented is that, with notifications disabled, an attacker can spend time attempting to crack the password without being noticed [28]. While it is true that such an attack could work, if strong passwords are chosen it should be infeasible.

## *2.5 Underlying Algorithms*

Another important thing to look at is whether any weaknesses have been found in the encryption and authentication algorithms being used. SNMPv3 can use either MD5, or SHA-1 in a standard HMAC construction to provide authentication [9]. Several papers have shown weaknesses in MD5 [37, 40, 25], and to a lesser extent SHA-1 [26], but it seems unlikely that these issues will affect SNMPv3 because they all involve finding collisions in the hashing algorithm. This means that an adversary still cannot generate the HMAC without the secret key. For encryption, SNMPv3 uses the Data Encryption Standard with Cipher Block Chaining (DES-CBC) to provide privacy [9]. Recent work has shown that due to the small key size, DES is vulnerable to brute force attacks [20, 21]. This is a serious threat to the privacy goal of SNMPv3 messages, because the encryption can be broken. Thankfully, there is a proposed standard which

uses AES-128 [8], and which has been included in some SNMP implementations such as Net-SNMP.

## 2.6 Implementation Issues

There have been a number of implementation specific vulnerabilities for SNMP over the years [15]. For instance, the Oulu University Secure Programming Group tested multiple SNMP implementations for errors when processing requests [38]. Most of these vulnerabilities have been for SNMPv1 and SNMPv2c, but there are likely others which exist for SNMPv3. Our intent in this paper, however, is not to look at weaknesses in specific implementations, but rather in the protocol itself.

## 2.7 Summary

So far little research has been focused on the overall security of the protocol. Research has been done on improving performance, overcoming obstacles in the network, coming up with alternate ways of securing the protocol, proving (or disproving) the strength of the underlying algorithms, and discovering implementation specific vulnerabilities. What has not been researched is how the algorithms are used together, and the assumptions that the protocol operates under. In many cases, violating such assumptions can invalidate other-wise secure schemes.

# CHAPTER III

# ARCHITECTURE

We wanted to demonstrate our attacks under a realistic scenario. To do this, we made decisions about what is and is not realistic. We also had to choose which factors to simulate, and which not to simulate. In this section, we present the decisions we made for our simulation, and our justifications for those decisions.

## *3.1 Devices and the Network*

The devices that are monitored in our setup are all virtual machines (VMs) configured on a virtual network. These virtual machines are running Fedora 16, a popular Linux distribution.

The differences between VMs running an SNMP agent on Linux and real devices running an SNMP agent will likely be minor. Nothing we are testing is specific to Linux, the SNMP agent we use, or VMs. While real devices will likely use a different agent and a different implementation, they should not behave noticeably differently. The other major difference between the two is their computational power. While our virtual machines could have at least an order of magnitude more computational power than some embedded devices, none of our tests should notice this difference as they should not generate significant load for any manager or managed agent.

Our VMs are configured with a virtual network. For most applications, virtual networks should not perform significantly differently than physical networks. In this case, our virtual network is composed of four hosts on the same broadcast domain with two managed agents configured using DHCP.

The reason we chose to put all four hosts on the same broadcast domain, was to more easily simulate a MITM attack. The virtual network we constructed was

composed of two separate virtual networks bridged by our MITM. This allowed the MITM to reliably see and modify all traffic traveling between the monitored agents and the network monitor.

In most physical networks today, it might be unrealistic to expect to be on the same broadcast domain due to the prevalence of switched networks. However, it is not unreasonable to think that an adversary could become a MITM. On physical networks, it is possible to have a traditional MITM. What is perhaps more likely though, is that an attacker might use ARP poisoning on a switched network to route traffic through their host. This is also a MITM, and easier to do in many cases. What we demonstrate on a single broadcast domain is essentially the more traditional MITM where we are physically located between the communicating hosts, but everything we do should be possible using ARP poisoning.

## *3.2 DHCP*

In the second proof of concept, at least one of the hosts used in the setup must be configured to use the Dynamic Host Configuration Protocol, DHCP. This is done because it is necessary for one of our attacks to be performed. It is hard to determine generally how likely it is for an SNMP agent to use DHCP as opposed to a static IP address. We contend that this is a realistic condition for several reasons. First, SNMP agents are run on a diverse group of devices, and whether or not to use DHCP is likely dependent on the role of the device. Second, our attacks only require that a *single* agent be using DHCP to be performed. Third, even if using DHCP on such devices were abnormal, such a decision *should* never negatively impact the security of this protocol.

## *3.3 The Network Monitor*

The manager, which we refer to as the network monitor, is a Linux virtual machine running the Nagios network monitoring application. Nagios is a popular open source

application used to monitor hosts on a network. For the most part, it is protocol agnostic as checks are written in the form of plugins. These plugins are run by Nagios and their output determines the check status. The specific plugin we use is the "check_snmp" plugin. It is included in many Nagios installations as a common plugin for checking SNMP hosts.

Nagios with "check_snmp" was chosen because both Nagios and the plugin are popular and open source. It is possible that our choice of network monitoring software could affect the difficulty of the attack. Both attacks are based on sending and receiving discovery messages, and our setup may send more discovery messages than others. The "check_snmp" plugin and the way in which it is used require discovery to occur before every check. It is possible other software may cache the results of the discovery process, so as not to have to perform the process every time. Even in such a case, our attacks are still valid because if at any point discovery occurs, we can subvert the protocol.

## *3.4 Net-SNMP*

Both the agents running on the hosts and the manager running on the network monitor use the Net-SNMP implementation of SNMP. Net-SNMP is a popular open source implementation of the SNMP protocol. It supports all versions of SNMP, and even includes AES-128 support for encryption. Since our attacks are not dependent on the implementation, our attacks should not be affected by choosing a different implementation. As mentioned previously it is possible that other implementations may cache the results of the discovery process, but in such a case the attack should still be possible although somewhat more difficult.

## *3.5 SNMP Agent Configuration*

Our agent configuration was very simple. We created a single SNMPv3 username with two different passwords, one for authentication and one for encryption. The

username was given access to a single OID on each agent with a security level of `authPriv`. This OID should be a value that differs between hosts, such as the OID for hostname. We ensured that each agent had a different `snmpEngineId`, and therefore different localized keys by looking in the Net-SNMP "snmpd.conf" file where the agent stores its information.

The reason we used a single username for both devices is because that is the configuration we claim is weak. It was the intention of the protocol, that a single username could be used on multiple devices. This is also the reason for requiring localized keys. We must consider the possibility that some networks may use a unique userName for each device, and in such a case our attacks would be ineffective. This is unlikely to be the case in most networks.

The reason we required SNMPv3 with a security level of authPriv, is because our intention is to show that SNMPv3 can be subverted even using the strongest security level. We understand that in some cases network operators may be less concerned with the security of their SNMP traffic, but there are many cases in which the security of the traffic is vital. By showing that SNMPv3 with authentication and encryption is vulnerable to these attacks, we show that even if SNMPv3 is used, all of the security features are required, and good passwords are chosen, the protocol may still not be secure.

# CHAPTER IV

# 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.*

## *4.1 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[21]. Even if a single key is compromised, the key does not provide an easy way of inferring other keys or the passphrase 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.

## *4.2 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).

# CHAPTER V

# EXPLOITING THE VULNERABILITIES

## *5.1 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.

## *5.2 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.

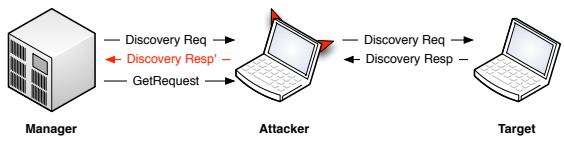


**Figure 1:** A `GetRequest` is forced to use a compromised key. This allows an attacker to successfully read the request

## *5.3 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

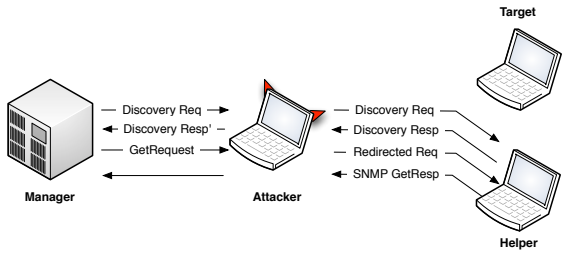


**Figure 2:** A `GetRequest` intended for the target is redirected to the helper.

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.

## *5.4 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 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

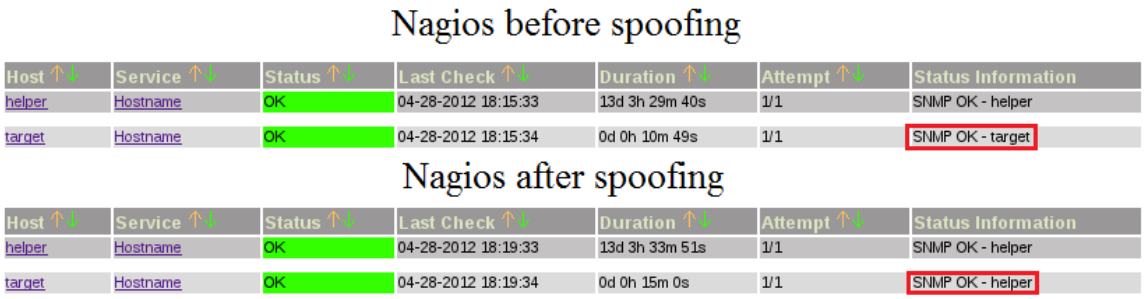


**Figure 3:** Initially, Nagios correctly reports the hostname for both hosts. We then use the helper to spoof SNMPv3 responses causing the target's hostname to change from "target" to "helper" (See "Status Information" column).

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.

From the Nagios interface, 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.

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.

# CHAPTER VI

# DISCUSSION

## *6.1 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 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]. Alternatively, the adversary could potentially modify settings on security appliances [14] 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"[22], it may be possible to force other security models to use USM.

## *6.2 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, temperatures 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.

## *6.3 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.

Because using IPsec is not always feasible, the best solution is to fix the protocol itself. These issues will persist as long as the discovery process is relied upon in its current state. We would recommend removing the discovery mechanism altogether, but we recognize that it may still be useful for helping to synchronize time values. Instead, the manager should be required to keep a list containing each `snmpEngineId` and its associated network address and to solely use discovery for synchronizing clocks. This would remove any ambiguity about which host is being communicated with, and prevent an attacker from being able to determine 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 such modifications would be harder to deploy. In addition to preventing all of the attacks mentioned, this approach would require no modification to the agents, and minimal changes to the protocol.

## *6.4 Similarities to SCADA Systems*

Supervisory Control and Data Acquisition (SCADA) systems are often used to monitor and manage industrial processes and infrastructure. SCADA systems use a variety of network protocols to communicate, some of which share similarities with SNMP. These protocols are often vulnerable due to poor (or nonexistent) authentication and a lack of encryption[24]. An example of this is the Distributed Network Protocol (DNP3). Even though DNP3 is the most widely used protocol for SCADA systems in the energy sector, it has traditionally been vulnerable to a variety of attacks.[19].

DNP3 is similar to SNMP for a variety of reasons. One is that, like SNMP, both protocols are relatively old and were designed with reliability as their first priority[2]. Like SNMP, early versions of DNP3 were vulnerable to a wide variety of attacks that would enable the interception, modification, and falsification of information[19]. Both protocols neglected to provide strong authentication due to its complexity. Like SNMP, later versions of the protocol attempted to retrofit security with minimal changes to the protocol. For instance, secure authentication was added to the standard in 2010 by using an HMAC[2]. The authentication in DNP3 differs from that in SNMPv3 in that it uses a challenge/response mechanism where as SNMPv3 only uses a single message to authenticate. It also attempts to maintain perfect forward secrecy by using a different authentication key for each session. It provides a key change mechanism with the intention that hosts' keys will be changed frequently. Unlike SNMPv3 however, the DNP3 standard does not include encryption.

# CHAPTER VII

# 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.

# 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] "Ieee standard for electric power systems communications – distributed network protocol (dnp3)," *IEEE Std 1815-2010*, pp. 1 –775, 1 2010.

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

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

[5] "Netfilter: Firewalling, nat, and packet mangling for linux." `www.netfilter.org`, May 2012.

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

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

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

[9] Blumenthal, U. and Wijnen, B., "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] C.-W., R. and Phan, "Cryptanalysis of the application secure alternative to snmp (apssnmp)," *Computer Standards & Interfaces*, vol. 31, no. 1, pp. 63 – 65, 2009.

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

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

[13] Case, J., Harrington, D., Presuhn, R., and Wijnen, B., "Message processing and dispatching for the simple network management protocol (snmp)," *Internet standard RFC*, vol. 3412, 2002.

[14] Chandika, N., "Cisco port security mib." `ftp://ftp.cisco.com/pub/mibs/v2/CISCO-PORT-SECURITY-MIB.my`, May 2002.

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

[16] Corrente, A. and Tura, L., "Security performance analysis of snmpv3 with respect to snmpv2c," in *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, vol. 1, pp. 729 –742 Vol.1, april 2004.

[17] Droms, R., "Dynamic host configuration protocol," *Internet standard RFC*, vol. 2131, 1997.

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

[19] East, S., Butts, J., Papa, M., and Shenoi, S., "A taxonomy of attacks on the dnp3 protocol," *Critical Infrastructure Protection III*, pp. 67–81, 2009.

[20] Gilmore, J., "Cracking des: Secrets of encryption research, wiretap politics & chip design," 1998.

[21] Guneysu, T., Kasper, T., Novotny, M., Paar, C., and Rupp, A., "Cryptanalysis with copacobana," *Computers, IEEE Transactions on*, vol. 57, pp. 1498 –1513, nov. 2008.

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

[23] Hia, H. and Midkiff, S., "Securing snmp across backbone networks," in *Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on*, pp. 190 –196, 2001.

[24] Johnson, R., "Survey of scada security challenges and potential attack vectors," in *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*, IEEE, 2010.

[25] Kelsey, J. and Kohno, T., "Herding hash functions and the nostradamus attack," *Advances in Cryptology-EUROCRYPT 2006*, pp. 183–200, 2006.

[26] Manuel, S., "Classification and generation of disturbance vectors for collision attacks against sha-1," *Designs, Codes and Cryptography*, pp. 1–17, 2011.

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

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

[29] Park, C., Jeong, K., Kim, S., and Lee, Y., "Nat issues in the remote management of home network devices," *Network, IEEE*, vol. 22, pp. 48 –55, september-october 2008.

[30] Plummer, D. C., "An ethernet address resolution protocol," *Internet standard RFC*, vol. 826, 1982.

[31] Presuhn, R., McCloghrie, K., and Waldbusser, S., "Transport mappings for the simple network management protocol (snmp)," *Internet standard RFC*, vol. 3415, 2002.

[32] Rosa, M., Nascimento, A., Sytnifc, L., and Duarte, E., "Jxta peer snmp: An snmp peer for inter-domain management," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pp. 645 –659, april 2008.

[33] Schonwalder, J. and Marinov, V., "On the impact of security protocols on the performance of snmp," *Network and Service Management, IEEE Transactions on*, vol. 8, pp. 52 –64, march 2011.

[34] Schonwalder, J. and Marinov, V., "On the impact of security protocols on the performance of snmp," *Network and Service Management, IEEE Transactions on*, vol. 8, pp. 52 –64, march 2011.

[35] Schonwalder, J., Pras, A., Harvan, M., Schippers, J., and van de Meent, R., "Snmp traffic analysis: Approaches, tools, and first results," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, pp. 323 –332, 21 2007-yearly 25 2007.

[36] Stallings, W., "Snmpv3: A security enhancement for snmp," *Communications Surveys Tutorials, IEEE*, vol. 1, pp. 2 –17, quarter 1998.

[37] Stevens, M., de Weger, B., and Schmitz, G., "On collisions for md5," *Master's thesis, Eindhoven University of Technology, Department of Mathematics and Computing Science (June 2007).*

[38] University of Oulu Secure Programming Group, "Protos test-suite: c06-snmpv1." `https://www.ee.oulu.fi/research/ouspg/PROTOS_Test-Suite_c06-snmpv1`.

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

[40] Wang, X., Feng, D., Lai, X., and Yu, H., "Collisions for hash functions md4, md5, haval-128 and ripemd," tech. rep., Cryptology ePrint Archive, Report 2004/199, 2004.

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

[42] Wijnen, B., Presuhn, R., and McCloghrie, K., "View-based access control model (vacm) for the simple network management protocol (snmp)," *Internet standard RFC*, vol. 3415, 2002.

# VITA

Nigel Lawrence is currently a graduate student in the School of Computer science and a GRA for GTRI. He obtained his Bachelor's degree at the Georgia Institute of Technology in 2011. While an undergraduate, he spent much of his time learning and using monitoring systems like Nagios while working as a Co-op student at GTRI.

Vulnerabilities in SNMPv3

Nigel R. Lawrence

39 Pages

Directed by Professor Patrick Traynor

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.