J Comput Virol (2007) 3:51–60
DOI 10.1007/s11416-007-0037-1

EXTENDED VERSION OF WTCV'06

Malware: a future framework for device, network and service management

Radu State · Olivier Festor

Received: 26 September 2006 / Accepted: 9 February 2007 / Published online: 8 March 2007

Abstract While worms and their propagation have been a major security threat over the past years, causing major financial losses and down times for many enterprises connected to the Internet, we will argue in this paper that valuable lessons can be learned from them and that network management, which is the activity supposed to prevent them, can actually benefit from their use. We focus on five lessons learned from current malware that can benefit to the network management community. For each topic, we analyse how it is been addressed in standard management frameworks, we identify their limits and describe how current malware already provides efficient solutions to these limits. We illustrate our claim through a case study on a realistic application of worm based network management, which is currently developed in our group.

1 Introduction

Integrated network and service management is the main component in a network allowing to operate a network and provide value added services with respect to contracted service levels. Network management activities are divided in five main functional areas (FCAPS) related to fault management, configuration, accounting tasks, performance and security monitoring. Management operations are performed within an administrative domain. From an historical perspective, an administrative domain is well delimited both by contractual and operational measures. In the new and rapidly evolving Internet, these limits become blurred. End users purchase devices and services requiring at least partial management from a third party. Multi provider service gateways become a reality and new management paradigms must emerge in order to cope with these challenges. There are multiple challenges that must be addressed. Managing a large scale infrastructure is probably among the first and major issues. Evolving towards a management plane capable to deal with millions of devices, spread over the whole Internet and accessible through firewalls and network address translation devices, is the main driving direction of research in network and service management. We argue that the ingredients for the future management plane are already being developed, although in a community that has not always the best intentions. The creators of malware and worms had been confronted from the early days with large scale infrastructures operated in hostile environments. We will highlight in the following the main features in current malware that we consider essential building blocks for managing the future Internet.

Our paper is structured as follows. The five main challenges that network management must face are presented in individual sections. Each section concludes with conceptual solutions inspired from current existing malware. Section 2 addresses the issue of large scale network management. Next, we consider the cases of multi-vendor and heterogeneous equipment which is the subject of Sect. 3. Three essential building blocks for any network management plane are given by its ability to be flexible, adaptive and to operate reliable and securely. These issues are addressed in the Sects. 4, 5, and 6,

R. State (✉) · O. Festor
Management of Dynamic Networks and Services, INRIA, Lorraine, France
e-mail: state@loria.fr

O. Festor
e-mail: festor@loria.fr

respectively. A case study of a large scale distributed honeypot is described in Sect. 7, where a malware based management is the only viable approach. Finally, we conclude the paper and highlight future works in Sect. 8.

2 Large scale device management

2.1 Scale processing in the standard management approaches

Scale has always been a concern in the device, network and service management community and various proposals to deal with it have been made over the last 20 years. In the early stages of management standardization, namely late 1980s, management framework designers did address the scale issue by extending the simple albeit efficient manager/agent model to a centralized hierarchical model where managers could be cascaded to deal each with a subset of the managed environment, thus applying a divide and conquer method. For example, some commercial OSI management platforms did provide a smart name-based routing scheme enabling management requests to travel to their destination agent through a tree of managers. While this approach has been proven efficient in some contexts, it did also show its limits; typically in the domain of fault-management and event correlation where the routing of events among a big tree of management nodes caused long delays in their processing.

A major step forward in the evolution of management architectures came through the proposal of management by delegation [1]. This model went beyond a tree of managers by proposing (and enabling) processing to be dynamically assigned to management agents themselves. This model has been instantiated over the years in many ways and efforts were undertaken within IETF to provide a standard support for such a delegation as part of the DISMAN initiative which has led to some RFCs on delegation and remote operations [2].

The pressure of scale in the management plane has again increased in the last 5 years for two reasons. First, new network infrastructures and services push the scale requirements on management to never imagined levels beyond millions of devices. This applies for example to dynamic update (e.g., patch distribution) of many millions of hosts [3], configuration and monitoring of several millions of cable modems and services to end users by a single operator, management of large P2P communities [4]. Second, management cannot live anymore in a world where resources can be considered infinite. While the benchmarking activity has seen a good evolution in the community, really large scale issues have not been addressed in a satisfactory manner. In fact, researchers of the management community did mainly focus their work on management integration enabling new network domains to be remotely managed (e.g., by making home networks managed by UPnP visible over TR-69 protocols in the provider domain through gateways), but these efforts are far from complete. For example, manual configuration is very often required on any single device to make it manageable and interference among services in many networks remains a problem for the management plane. In fact most standard management protocols simply do not work in most networks where firewalls and NATs are in place.

2.2 Malware and large scale management

It is known that botnets (network of compromised machines) under the sole control of one malicious user can range well into hundreds of thousands if not even millions of devices. A well documented case [5] is more that illustrative. One person was charged with having compromised more than 400,000 machines, installing malware and allowing third party spammers to use them as relay servers. Spammers could use this on-demand infrastructure over a delimited time frame and billing was done online. It is more than surprising to discover that (1) managing a large scale of bots is possible, (2) allowing partial third party control to this infrastructure [in network management terms, we would call it customer network management (CNM)]: is possible and that (3) billing and accounting have been also well addressed. It is true that the functionalities of the network were limited to only a limited set of well defined operations, but nevertheless even these primitive management operations are good examples for the viability of such an approach. The idea of using malware type of techniques (see Fig. 1) for doing beneficial activities is not new. In fact, the "ethical worm" [6] did already attack machines that were infected by the Code Red worm and removed the latter. The Code Green worm attacked infected machines using the same exploit code and thus provided for a first real world example of worm based management platform. A similar case is given by the Dabber and Gaobot. AJS which perform direct attacks against the Sasser worm. We will not discuss further the legal and ethical implications of managing equipment belonging to another administrative domain but it is worth to note that this real world experiment showed the viability of such management, although it used only the propagation features of worms. We argue that such approaches extended with administrative domain bounded propagation and additional middleware are the ingredients for highly efficient

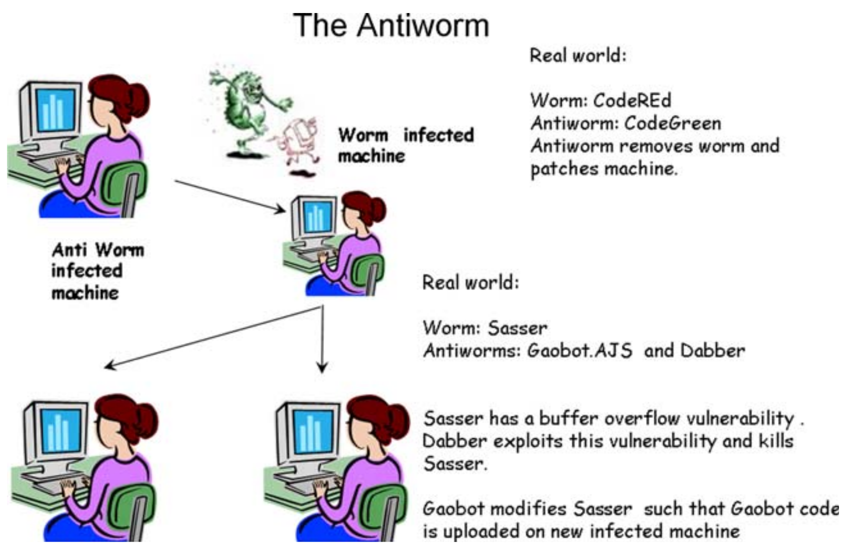
The Antiworm
Real world:
Worm: CodeREd
Antiworm: CodeGreen
Antiworm removes worm and patches machine.
Worm infected machine
Anti Worm infected machine
Real world:
Worm: Sasser
Antiworms: Gaobot.AJS and Dabber
Sasser has a buffer overflow vulnerability .
Dabber exploits this vulnerability and kills Sasser.
Gaobot modifies Sasser such that Gaobot code is uploaded on new infected machine

Fig. 1 The antiworm

network management bricks. A promising early work on beneficial (nematodes) controlled worms is described in [7], while a truly pioneering work in this area is described in [8].

3 Heterogeneous multi-vendor equipment

3.1 Standard network management protocols

The goal to manage in an integrated way, heterogeneous devices and services was (and still is) the "raison d'etre" of standardization in the management plane. Undertaken in the late 1980s, standardization of the management plane was initially driven by ISO and ITU which defined the foundations of Integrated Network and Service Management: the Manager/Agent model, the functional areas, the concept of device independent information model and the standard service and protocol to remotely access management information. This model was very ambitious, but complex, resource consuming and often ambiguous and hardly usable despite the large industrial support to instantiate the standard through industrial experience and tuning (as done in the Network Management Forum for example). While today many parts of the OSI management model remain in most management approaches, its information modeling approach and language together with its service and protocol have vanished. In fact, they still remain operational in many telecommunication networks but are not deployed in any new infrastructure. Following the ISO efforts, other bodies did initiate their own management standardization activities. The first one was of course IETF which came out with the SNMP (initially designed as a transition technology before the full support and deployment of OSI-based management technologiess...). More recently, DMTF, Oasis-Open and many other technology groups (3GPP, ATM Forum, ...) did publish their own management frameworks dedicated to the services they address.

There is no doubt that standardization has been of great value to establish common foundations and models for device, network and service management. History however, tells us that the goal to build a generic widely accepted management model applicable to universal domains has clearly failed. In fact, even in a single standardization body, one finds several management standards, although designed for specific functions, often featuring services that compete with others. A typical example can be found in IETF with Netconf, the various COPS and SNMP efforts. A second element that confirms the failure of standardization to handle heterogeneity is the success of proprietary non-standardized solutions provided by vendors (e.g., CLI, ad-hoc protocols & frameworks like Webmin) or even private branches of standardized approaches (most MIB information available today are proprietary data not standardized in an information model). The inability of management standards to cope with the reality of the networks that evolve very fast is increasing every day. A simple, albeit strong, reason is the increasing delays that management evolutions face in standardization bodies. Even if not encouraged by the various standardization bodies, today standards emerge long after the protocol/service they are supposed to manage and evolutions of existing management standards take a very long time (if not eternity).

The more we progress, the less management standards are usable to solve the heterogeneity problem of managed devices. Standard management is not existent anymore and if available, not operational in many networks that need management on a large scale. The time has come to find alternatives.

3.2 Malware and cross-system integration

In the world of malware, dealing with multiple types of systems and access mechanisms did not really seem to be a major problem. Worms [9] and [10] are capable of spreading using at least two very different operating systems. These type of malware are exploiting known vulnerabilities on these systems and during their propagation, an accurate remote fingerprinting capability allows to attack a target with the matching exploit code. An exploit code will typically inject a shellcode in the process space of a vulnerable application. The shellcode represents in most cases machine code that can be directly executed on the targeted machine. Although this machine code is highly specific to a target machine architecture, recent advances in shellcode generation moved towards multisystem shellcode [11] capable to be executed on different platforms.

The lesson is that a promising approach for managing multivendor and heterogenous systems, might be to shift from addressing the integration at the agent side, towards flexible management proxies able to interact with various agent endpoints. If we consider the relative long timeframe required to standardize a network management protocol, such an approach might prove efficient on both the short- and long-term.

4 Dynamic management plane

4.1 Dynamic delegation

The need to offer the ability to dynamically change the behavior of a management entity (either the manager or the agent) was recognized very early in the management community and has led to a full range of investigations and innovations over the last decade. Dynamics was first introduced in the concept of delegation in the mid 1990s [1]. This model did extend existing scripted agent approaches with elastic servers for management enabling any type of delegation code to be shipped to and run by remote management agents.

Active networks [12] represent the ultimate approach for dynamic network management. The basic approach is similar to the delegation approach of Yemini but the active network model assumes a strong linkage between the network activity and the associated management code. The model supports various dynamicity levels for delegation of management code (from an external pre-provisioning of management code on the devices to a per packet level management code deployment) and has led to interesting proposals in the area of fully distributed cooperative management where several management packets with different capabilities are cruising in the network calling in specific support (i.e., other packets transporting specific management code) on the discovery of specific events (e.g., a long-lasting congestion, an identified DDOS, . . .) [13–15].

While active networking for management was a promising approach, it failed to gain acceptance because of many reasons. First, its focus on the network plane did limit its potential deployment in the dedicated network devices which are often closed devices with no capability for hosting external code for local processing. The second reason for the low impact of active networks so far is the lack of security of the infrastructure and the difficulty for administrators to maintain the management plane under control.

4.2 Artificial life and worms

While active networking technology promised seamless deployment of new applications and functional extension of already existing software frameworks, its real world deployment and effective usage remained more than modest. Network operators did not endorsed this concept and the lack of a standardized software interface and application programming interface did limit the usefulness of these approaches. One of the major problems that was not solved by the work done in this area was the identification of a real “killer application” that would otherwise not be possible.

In the field of Artificial Intelligence, a major wave of research activities were driven in the late 1980s by the notion of genetic programming [16] and artificial life [17]. The major idea behind these concepts was to build self replicating programs capable to drive the evolution of an initial population towards better and improved future generations. Simulating a Darwinian type of evolution, a computer program was considered to be an individual and genetic operations between two individuals allowed to generate new offsprings bearing the most important and relevant functionalities. Most of the approaches in these works considered that a computer program is represented under a tree type of structure and that such tree structures can be combined and evolved towards better and more suitable code structures. Unfortunately, in real world scenarios, combining

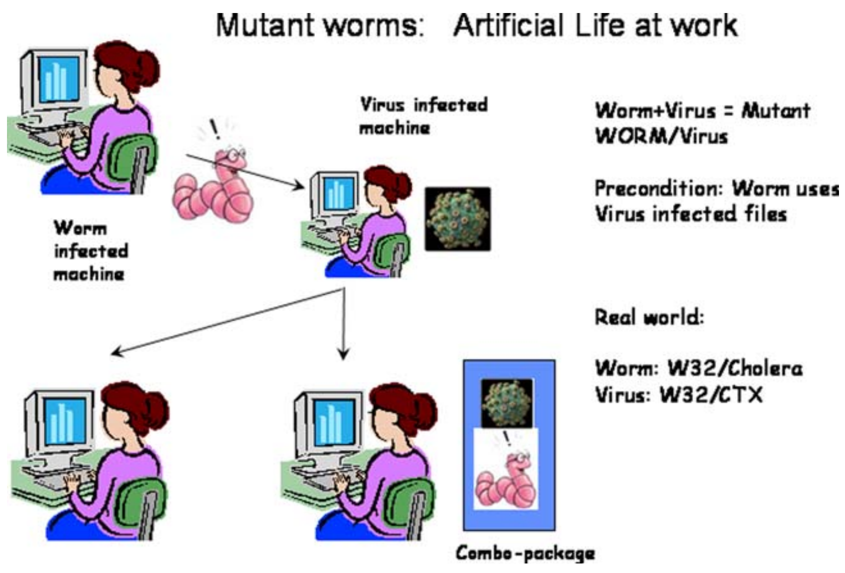
Mutant worms: Artificial Life at work
Virus infected machine
Worm+Virus = Mutant WORM/Virus
Precondition: Worm uses Virus infected files
Worm infected machine
Real world:
Worm: W32/Cholera
Virus: W32/CTX
Combo-package

Fig. 2 Artificial life in malware

code structures without addressing the semantics and without using domain specific knowledge does seldom lead to something meaningful. Most of the evolved code structures are not fit and capable to provide reliable and functional programs. On the other hand, Internet worms show that active technology, or at least some of its components existed before and survived the works done by the active networking community. The major illustrative example is the online combination of a worm with a virus (see Fig. 2) as described in [10]. The Cholera worm and the CTX virus use the same files for hiding and propagating. When the CTX virus infected machine is next infected with the Cholera worm, a new creature is born combining functionalities from both malwares and having more propagation capabilities.

What should we learn from this case study? The lesson learned with the application of active network technologies in the management plane is that autonomy for management functions is interesting and in some cases useful but that at any point in time, centralized control must be offered. Also, deployment models that do not rely on the standard fixed, pre-configured manager/agent paradigm are promising but need to be refined and evaluated in several large scale management scenarios. Techniques coming from the artificial life community could be very interesting for having evolving and adaptive management functionalities. The important issue that we must deal with, is to define the right code granularity. Unlike to the traditional artificial life approaches, where a very fine granularity (at instruction level) is used to represent each evolving creature, we should use a higher level granularity (at file level or sub-parts of a file) and allow combinations and the generation of new offsprings from these artificial creatures.

5 Reliable middleware

5.1 The dedicated management network

Standard management makes a couple of assumptions among which the fact that the management channel is fixed and operational. In several telecommunication networks, the management plane is supported by a dedicated physical network on which both managers and agents interact over a reliable infrastructure and middleware. The most used middleware for supporting management applications, i.e., the middleware used to bind all the applications of a management platform has without doubt been Corba combined with messaging systems. Today this middleware is slowly being replaced by web services.

In most emerging networks the provisioning of a dedicated network solely for management purpose is not feasible anymore. A good example is the wireless world were management data is shipped to and from the devices over the same channels as application payload (e.g., shipping management commands to a cell phone is often done over SMS). Moreover, the advent of new large scale distributed services, like P2P systems, offer an alternative to dedicated large and complex management centers, enabling the management activity to be spread among a community providing sufficient assurance on the availability and the quality for the offered service.

5.2 Worms IRC and P2P

There are two notable examples of worms using state of the art supporting middleware. The first one is the case of the Slapper worm [18]. The second notable illustration

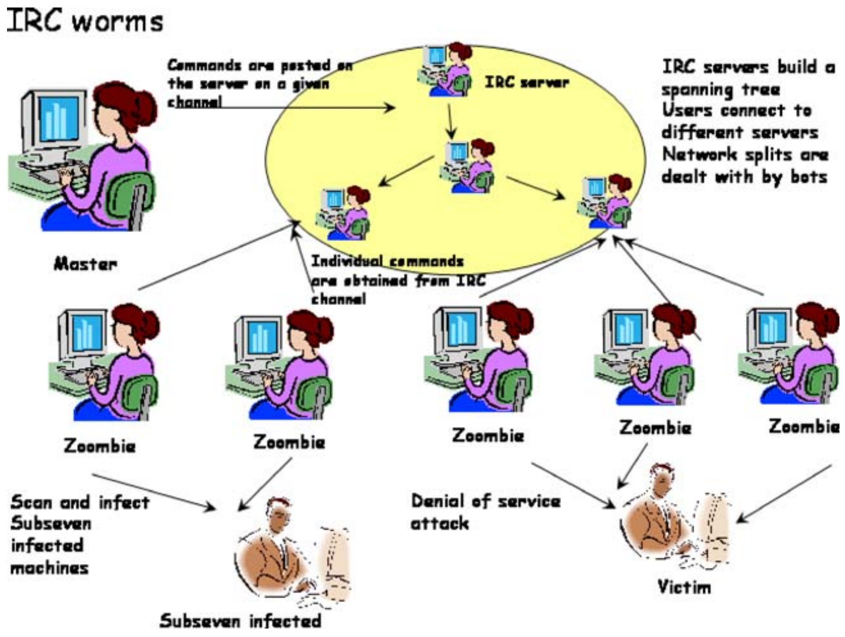
IRC worms
Commands are posted on the server on a given channel
IRC server
IRC servers build a spanning tree
Users connect to different servers
Network splits are dealt with by bots
Master
Individual commands are obtained from IRC channel
Zoombie
Zoombie
Zoombie
Zoombie
Zoombie
Scan and infect Subseven infected machines
Denial of service attack
Subseven infected
Victim

Fig. 3 IRC based worms

is given by a large category of worms using the IRC protocol (see Fig. 3 for an example of the Tendoolf worm) as the supporting service. The key components of an IRC network are clients, servers and channels. A client will join a channel on a particular server. A channel is supported by multiple servers and these servers are responsible to relay messages among the multiple clients. These IRC servers build an overlay network which behaves like an application level spanning tree.

A client connecting to a server will directly communicate with this one, but all messages sent to a channel will be relayed among all the servers. A channel can be conceptually seen as the equivalent of a multicast group. A server will receive messages on a given channel if and only if it supports clients on that channel. It is considered in [19] that this approach does not scale well for two main reasons:

(1) Network congestion might cause a link connecting two servers to fail,
(2) Additional state information about the channels and clients is requSired to be supported by the servers.

These arguments are sound, but as of today, the IRC provided a sufficient scalable solutions for large scale communication among infected zombies and their master. The key architectural design behind the worms using IRC is the use of a publish/subscribe communication system. IRC Servers keep channels allowing only to publishers to talk. These publishers can be both infected machines or the zombie master. Infected machines will typically post multiple types of information: the IP address, details about the configuration and the operating system as well as replies to requests coming from the zombie master. The zombie master posts requests (these can be commands to continue the infection and propagation), perform denial of service attacks, or to stop the propagation and perform a collective suicide [6]. In the same time, clients will listen on a channel, wait for events and act if required to. An even more advanced middleware is embedded in the Slapper worm (see Fig. 4). The Slapper worm [18] builds an overlay network capable to provide advanced features among which, the most notable are:

- Reliable end to end message delivery using the overlay network. Each node in the P2P network implements a simple and efficient message delivery process allowing retransmissions and probabilistic routings. A node that has to forward a message to another node, will randomly select a neighbor to which the message will be forwarded. A timeout mechanism will assure the retransmission and an additional message tracking will assure that no routing loops are generated.
- Coping with network partitions and reshaping. Firewalls and system security management might lead to network partitions or machines that are taken out from the bot network. The P2P middleware of the Slapper worm is capable to adapt to such events, respawn a new network (if required) and maintain the logical topology of the network.
- Anonymous message delivery. One of the key features that Slapper implements is that application level

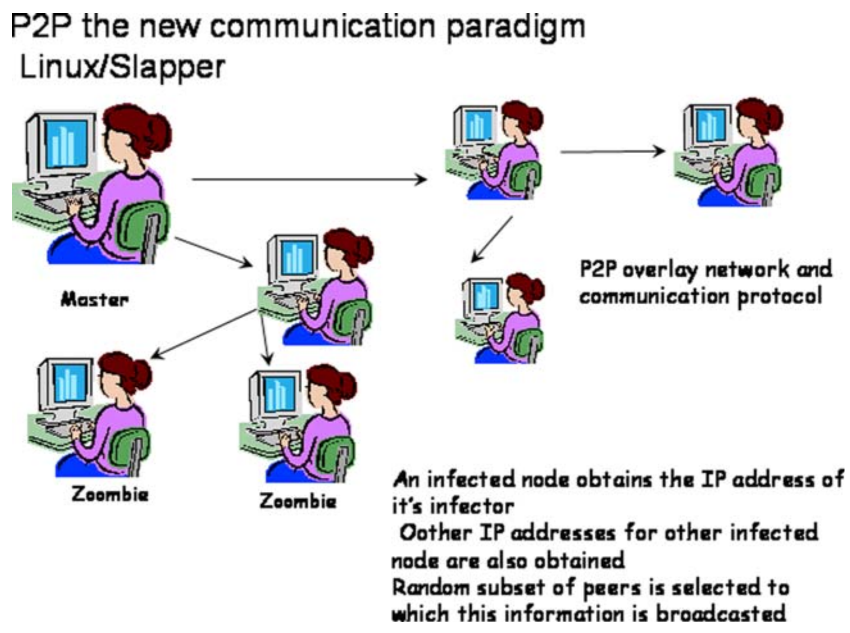
P2P the new communication paradigm
Linux/Slapper
Master
Zoombie
Zoombie
P2P overlay network and communication protocol
An infected node obtains the IP address of it's infector
Oother IP addresses for other infected node are also obtained
Random subset of peers is selected to which this information is broadcasted

Fig. 4 P2P based worms

routing allows to one node to communicate with another node such that their identity (IP addresses) remain anonymous. This is implemented by the probabilistic routing mechanism. This hinders any effort to identify the endpoints of a communication. From a conceptual point of view, this scheme is equivalent to a simplified onion routing [20] scheme.

The lesson that network management should learn from these architectures is that the classical client server interaction between a manager and an agent has to evolve towards either a novel peer to peer based management framework or a native publish subscribe one. Such a scheme is better suited for intermittent network connectivity and allows a high degree of privacy for network management.

6 Security and configuration management

6.1 Security in the management plane

At times where a dedicated network was in place which carried all the management traffic and was inaccessible to any intruders, security was physically enforced. When management traffic started to be mixed in application traffic, security suddenly became important to network administrators (who discovered that the community string sent in clear text over the wire was a real threat towards the management plane). The advent of secure protocols like SNMPv3 in the management plane unfortunately did not solve the security problem in the management plane as it should have, mainly for two reasons. First, very few administrators did move from an unsafe version of SNMP (v1) to SNMPv3 on their networks. Second, the multiplication of management interfaces on a single device has opened many vulnerable channels. As an illustration, while the CLI interface of a large campus switch was secured, it took us 3 min to get full control of the device simply by changing the accounts on the device over SNMP which was left open. This is often the case with other devices today like printers and their web-based management interfaces.

Recent work in our group has led to the design of generic management plane level security and we have designed several mapping approaches that enable to maintain a consistent security policy on various management channels (Netconf, CLI and SNMPv3) over many devices [21]. While this work is promising, it does not yet include all possible management interfaces (e.g., Web-based interfaces, Web-services specific interfaces, ...) and would need standardization support for the definition and maintenance of policies and technology interfaces. This is unlikely to happen very soon leaving the management plane unsecured still for a long time.

6.2 Secure worm based configuration

Although from the early years of Internet and its worms, authors of malware considered that a secure configuration management lifecycle has to be an essential and integrated building block. Among the different existing approaches, the most illustrative case is the Hybris virus [10,22]. As it is illustrated in Fig. 5, an advanced

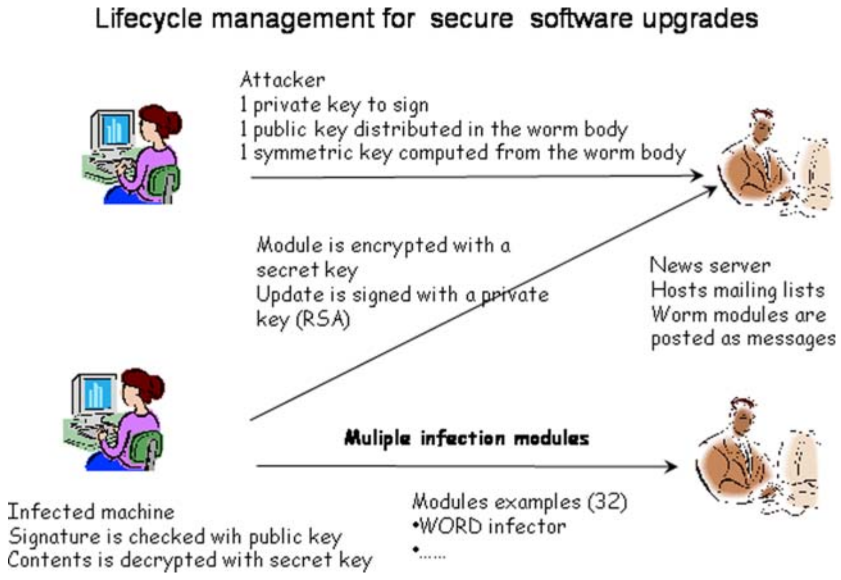
Lifecycle management for secure software upgrades
Attacker
1 private key to sign
1 public key distributed in the worm body
1 symmetric key computed from the worm body
Module is encrypted with a secret key
Update is signed with a private key (RSA)
News server
Hosts mailing lists
Worm modules are posted as messages
Muliple infection modules
Infected machine
Signature is checked wih public key
Contents is decrypted with secret key
Modules examples (32)
•WORD infector
•......

Fig. 5 Secure configuration life cycle

cryptographic authentication mechanism is used to identify the manager and assure the integrity and confidentiality of the software updates. The authors of this virus wanted to guarantee that only legitimate updates can be deployed via a public medium (news server) and the contents of these updates remains accessible only to the infected and controlled agents.

The underlying idea is very simple and elegant. The updates can be checked for integrity using a public key based cryptography. A shared symmetric encryption key is computed from the code of the virus and used to decrypt the contents of these updates. The large number of different updates that emerged in the following years shows that both from a conceptual and operational aspect this design was sound.

Even though a secure life cycle management represents only a subset of service management it is worth to notice that using well established cryptographic protocols and a simple key distribution mechanism are the necessary ingredients for a well secured service management plane. A more extended approach should address a broader scope of management functionalities including service monitoring and configuration as well as more network level related operations.

For a comprehensive overview of armored malware and a detailed case study, the interested reader is encouraged to check the references [23] and [24], respectively.

7 Towards a worm based management plane: a case study

Several target deployment environments could benefit from a worm based management architecture. The common features of such environments are: multiple devices operated in hostile networks (a hostile network is a network operated by a third party or end user), where strong security and efficiency are required. Typical examples are: set-op devices managed by IPTV providers, Internet enabled home appliances, service gateways, VoIP terminals and large scale honeynets. The latter is also the case that will be sketched out in this section.

The Internet as of today is becoming the virtual playground of a more and more younger user category, but is unfortunately also the working environment of criminal predators, among which a major category is associated to crimes and abuses against children. A safer Internet can be made by keeping the predators away from a vulnerable user population. Technically this can be done by a sort of black list, where IP addresses from this list are not allowed to connect to virtual chat rooms, direct communication or blogs used by young Internet users. The key information required to configure such a blacklist are the IP addresses of potential predators. These predators are involved in trading/exchanging illegal files (in most cases multimedia files which are illegal and violate any moral and legal law) using multiple types of overlay distribution networks. Such users can use a large variety of client software, ranging from the well known P2P clients (emule, edonkey, dc++) and up to the more efficient bittorrent or IRC networks. One solution towards the identification of such a user category is a large scale distributed overlay honeypot. The major objective of such a honeypot would be to advertise the availability of such illegal content and identify the clients trying to download it. Although, some false positives might be possible, it is reasonable to assume that one IP address that is frequently trying to download such content can be safely assumed to harbor a predator. From an operational perspective, obviously no real content will be distributed over the network. Rogue data will be advertised and clients trying to download the data will be logged and put forever on a download waiting list. Our main working assumptions are the following:

1. A large population of users will deploy the client side of the honeypot. Many well intentioned users will want to help and participate in this collective hunt of predators.
2. Malicious users will install the honeypot and tamper with it in order to identify and attack the management plane.
3. The operations of the honeypot should be highly secure in a potential hostile environment, while still being transparent to well intentioned users.

Why do we require a worm based management framework for such a purpose? If we assume that a very large category of users participate in this hunting effort, then we will have to address the management of a set of devices and services having the major features:

- Large scale distributed management. For a large population of users taking part in such a honeypot, we will have to manage the individual honeypots. This management must assure at least the configuration (list of file names to be advertised) as well as the monitoring (requesting the logs of IP addresses having tried to download a particular file). Since downloads are operated over TCP and the requests to download a file require at least the establishment of the TCP three way handshake, a malicious user is not capable to use a spoofed IP address.
- Secure update and configuration. The management must be done in full security. The minimal requirements are to authenticate the manager and securely retrieve the information about the clients having tried to upload a given file.
- Provide to the client deploying the honeypot the information about the advertised file names (if requested by the client) while still assuring that no tampering from the latter is possible. This can be done using cryptographic mechanisms that are similar to the ones encountered in the case of the Hybris virus.
- Preserve the anonymous identity of the management plane. If one honeypot is to be deployed by a malicious user, he should not be able to identify where the management operations are initiated from. For this case, publish/subscribe communication paradigms and onion routing mechanisms similar to the ones used by the Slapper worm or the IRC based worms are the only viable solution.

The described architecture (see Fig. 6) is under current development in our group and an open source implementation will be released in the very short future. The implementation is based on a custom extension of an emule client with an additional support for monitoring and management extension as described in this paper. The implementation runs on Windows based systems and is developed in MS Visual C++. The current implementation targets the emule/edonkey P2P networks. The main idea of our honeypot can be

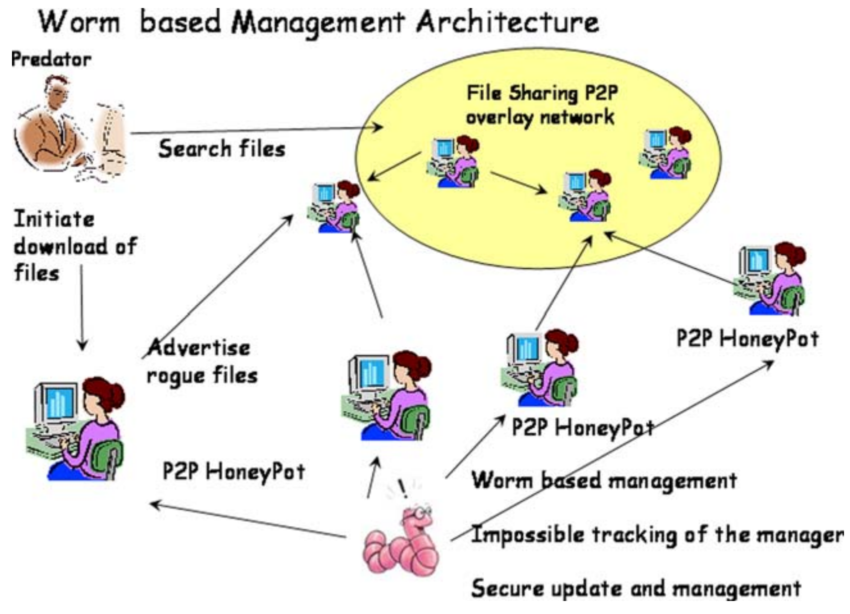
Worm based Management Architecture
Predator
File Sharing P2P overlay network
Search files
Initiate download of files
Advertise rogue files
P2P HoneyPot
P2P HoneyPot
P2P HoneyPot
Worm based management
Impossible tracking of the manager
Secure update and management

Fig. 6 Worm based management: a case study

applied also to other filesharing networks, but for each of these networks, we need to implement a dedicated advertisement mechanism. As of today, we have not yet performed realistic tests with our prototype.

8 Conclusions and future work

Valuable lessons can be learned from worms and current malware. Even though the authors of malware are driven by less honorable intentions, from a technical point of view they successfully managed to overcome some major technical challenges. We argue in this paper that similar technical solutions must become essential components for current network and service management frameworks. Addressing in a cost efficient way the management of large scale infrastructures with the current traditional approaches is no more viable. On the other hand, Internet worms are solid proofs that large scale device management is possible with relative low costs in highly hostile environments. Firewalls, network intrusion detectors as well as system and network managers are the hostile environments, in which worms and malware are capable to propagate and exist. The rigid and well established manager agent interaction based on standardized information models and management protocols requires a major shift towards a lightly coupled, epidemiologic management architecture. In order to meet these goals, two paths are possible. The first path is highly radical and assumes to start with a complete clean slate for network management grounded on worm based behavioral features. The second path is more smooth and transitional, where existing management frameworks will gradually include such features. Only time will show which is the right path to go, but in order to succeed major future research and technological challenges must be faced by the network management community.

Acknowledgment This paper was supported in part by the EC IST-EMANICS Network of Excellence (#26854).

References

1. Goldszmidt, G., Yemini, Y.: Distributed management by delegation. In: 15th International conference on distributed computing systems. IEEE Computer Society (1995)
2. Schoenwaelder, J., Quittek, J.: Secure management by delegation within the internet management framework. In: 6th IFIP/IEEE international symposium on integrated network management,Boston. IEEE Computer Society (1999)
3. Gkantsidis, C., Karagiannis, T., Rodriguez, P., Vojnovic, M.: Planet scale software updates. In: Proceedings of ACM SIGCOMM‘2006 (2006)
4. Doyen, G., Nataf, E., Festor, O.: A hierarchical architecture for a distributed management of p2p networks and services. In: Schönwälder J., Serrat J., (eds). DSOM, ser. Lecture Notes in Computer Science, vol. 3775. Springer, pp 257–268 (2005)
5. Lemos, R.: Major prison time for bot master, http://www.securityfocus.com/brief/205 (2006)
6. Nazario, J.: Defense and Detection Strategies against Internet Worms. Artech House Publishers (2003)
7. Aitel, D.: Nematodes, http://www.immunitysec.com/resources-papers.shtml (2006)
8. Schoch, J., Hupp, J.: The “worm” programs— early experience with a distributed computation. Commun. ACM (1982)
9. Cohen, F.: Computer Viruses, Wiley, London (1985)
10. Szor, P.: The Art of Computer Virus Research and Defense. Addison-Wesley Professional, Reading (2005)
11. Koziol, J., Litchfield, D., Aitel, D., Anley, C., Eren, S., Mehta, N.: The Shellcoder’s Handbook: Discovering and Exploiting Security Holes. Wiley, London (2004)
12. Tennenhouse, D., Wetherall, D.: Towards an Active Network Architecture. Comput. Commun. Rev. **26**(2), (1996)
13. Sugauchi, K., Miyazaki, S., Yoshida, K., Nakane, S., Covaci, K., Zhang, T.: Flexible Network Management Using Active Network Framework. In: Covaci S. (ed.) Active networks: Proceedings of 1st international working Conference, IWAN‘99, Berlin. LNCS 1653, pp. 241–248. Springer, Heidelberg (1999)
14. Brunner, P., Stadler, R.: The impact of active networking technology on service Management in a telecom environment. In: Proceedings of 6th IFIP/IEEE international symposium on integrated network management (IM’99) Boston
15. Schwartz, B., Jackson, A., Strayer, W., Zhou, W., Rockwell, D., Partridge, C.: Smart packets: applying active networks to network management. ACM Trans. Comput. Syst. **18**(1), 67–88 (2000)
16. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
17. Adamatzky, A., Maciej Komosinski, M.: Artificial Life Models in Software. Springer, Heidelberg (2005)
18. Arce, I., Levy, E.: An analysis of the slapper worm. IEEE Security and Privacy (2003)
19. IETF.: Network working group c. kalt request for comments:2810 (2000)
20. Goldschlag, D., Reed, M., Syverson, P.: Hiding routing information. LNCS **1174**, 137–150 (1996)
21. Cridlig, V., Festor, O., State, R.: Role-based access control for xml enabled management gateways. In: Sahai A., Wu F. (eds.) DSOM, ser. Lecture Notes in Computer Science, vol. 3278, pp. 183–195. Springer, Heidelberg (2004)
22. Filiol, E.: Computer Viruses: From Theory To Applications.Springer,Heidelberg (2005)
23. Young, A., Yung, M., Malicious Cryptography: Exposing Cryptovirology, Wiley, London (2004)
24. Filiol, E.: Strong cryptography armoured computer viruses forbidding code analysis: the bradley virus. In: EICAR2005 annual conference 14, StJuliens/Valletta-Malta (2005)