Network Infrastructure Forensics

Felix Lindner

Recurity Labs GmbH
Wrangelstr. 4
10997 Berlin, Germany
fx@recurity-labs.com

Abstract: Incident identification, response and forensic analysis depend on the ability to extract meaningful evidence from the suspected system. Such tools do not exist for network infrastructure equipment. The significantly increased attack resilience of common general purpose operating systems poses a surprising new challenge to forensics, as attackers will likely shift their attention back towards network infrastructure control. The paper discusses the importance of network equipment forensics, the anatomy of devices and the attack types encountered. Finally a method for performing forensics on a widely used type of network equipment is presented.

1 Introduction

According to the definition, the goal of computer forensics is to explain the current state of a digital artifact [Wi08]. Computer forensics is normally performed on common desktop and server computer systems, digital media and data items. In the past, very little attention was paid to computer forensics on network infrastructure equipment.

This paper first discusses the need for forensic analysis methods and supporting technologies for network equipment and will show why the previously neglected capabilities need to get build up quickly. Following, the paper classifies the types of network equipment commonly encountered and the types of attacks that need to be detectable. The paper concludes with a comparison of the currently used evidence gathering mechanisms to a method developed by the author.

2 Network Equipment Forensics

Forensic investigators should be able to extract evidence from any sufficiently complex electronic device in a regular computer network. As new device classes become common on a regular basis, forensic investigators have to identify the means of performing evidence gathering, examination and analysis on the new device classes. Recent challenges for investigators include various mobile phone platforms, MP3 players and portable storage devices. New capabilities have to be developed constantly as users store relevant data in more diverse places, distributing information important for investigations.

Digital forensic procedures have so far largely neglected the device class of network equipment. For the purpose of this paper, we define network equipment as:

> A physically enclosed component of a communications network that is built for the purpose of performing programmed transport actions on data, based on information within the transported data.

The definition encompasses all so-called “active network gear” with the exception of simple repeater stations or otherwise “non intelligent” equipment, but excludes multi-purpose operating systems that could perform the same task.

We will now review the requirement to have analysis methods and supporting technologies for network equipment.

2.1 Attacker Focus Shift

The need to perform digital forensic investigations on a device class is usually created by users storing personal information on devices of this class. Within this category also falls the case of the device class becoming a target of malicious activity by itself, as attackers must inject data into the device to perform directed attacks. Alternatively, the need for digital forensics can also arise when the device class exhibits detectable traces of user data or communication. The later argument holds true for network equipment by its very nature, the former requires inspecting network equipment as a target more closely.

Directed attacks on network equipment were the norm in the early days of communication networks, when sender or sink devices on the network were either dumb or closely monitored equipment in the hands of experts. Directed attacks on network equipment were declining in the more recent past, as the connected computer systems with common operating systems became increasingly powerful, interconnected and commonplace. The effort required to successfully attack a network device was now comparatively large in contrast to the development of an attack against a common operating system platform or widely used software. Additionally, successful intrusions into network equipment are not as easily detectible as intrusions on multi-purpose computer systems are. Accordingly, digital forensics has had little demand to perform investigations on network equipment and the numbers on un-detected attacks are probably much higher.

In the recent years, fueled by public demand to stop large scale intrusions and worm outbreaks, the vendors and publishers of common operating systems have significantly improved the protections shipped with their products. Security features, mitigation technologies and better control over privileges have become an important aspect of any operating system deployment decision made, from professional applications down to home users. As security becomes a market factor, competition drives significant improvements on all major operating system platforms, compilers and libraries. Additionally, a plethora of third party security products is available and marketed to close the remaining gaps left open by default installations and hard to secure application software.

The success of this market-wide shift towards more secure software and installations can be easily verified by the drastically lower numbers of newly detected vulnerabilities in operating systems. Another indicator is the apparent end of the era of worms that rely on software vulnerability exploitation for propagation. Not a single large outbreak has been observed in the last years on platforms that used to be notoriously infected just a few years ago.

It can, however, not be shown or even expected that the total number of attackers or their skill level decreased over the same period of time. To the contrary, cyberspace warfare has become an integral part of many first world nation states' military operations. Offensive cyber-warfare groups have now to prove their usefulness. Privately operated groups, mainly illegal, have also banked on business models involving intrusions into computer systems.

This produces a gap between the increase in efforts required for a successful intrusion and the goals of such groups, may they be government sponsored or privately run operations. Since network equipment provides significantly less resilience against attacks than modern operating systems on desktops and servers, it is the opinion of the author that the focus of professional attack groups will shift towards network equipment again.

2.2 Network Equipment Attacks in the Wild

Very little is publicly known about successful attacks on network equipment. While a number of articles have been published in so-called underground magazines [Ph00] on different procedures to perform attacks and establish control of network equipment, intrusions remain mostly undetected and hence undocumented.

Reports about the abuse of illegal access to network equipment are only recently disclosed publicly. [Di08] discusses a case of blackmail in which the attacker(s) obtained control over the router network and blackmailed the legitimate owner of the network for restoration of their access without network downtime. The alternative for the legitimate owners would have been an entire network shutdown and incremental restoration and restart procedure, potentially requiring several days of downtime.

When inspecting publicly accessible information on attacks in general, it can be observed that the real number of attacks on network equipment is not to be neglected. In [BC04], an open HTTP proxy server is inspected for malicious activity performed through its service. 0.94% of all activity targets Cisco routers. When searching for information on a single Cisco IOS vulnerability [CVE01], sources such as [Te06] show that it is actively scanned for and exploited in the wild five years after the vulnerability has been reported and fixes are available. Since attackers rarely scan networks for vulnerabilities that they consider extinct, reports like [Te06] indicate clearly that the attackers consider it likely to still find vulnerable equipment.

2.3 Underlying Threats

To understand the underlying threat of compromised network equipment, it is necessary to review the basic functionality of today's communication networks. Almost any network today uses the TCP/IP protocol suite [Po81], which by design relies on intelligent network nodes taking decisions on the transport of user data from one point in the network to another.

The communication endpoints in IP have very limited abilities to influence the path a segment of the data (packet) takes from the sender to the sink. With the rapid expansion of the Internet, all remaining functionality of sender-decided paths was blocked in the network to prevent endpoints from influencing the availability of specific network paths.

The intelligent network nodes required by IP are network equipment. They rely on so-called routing protocols to communicate network paths to each other. Without this communication, the nodes cannot fulfill heir role. All routing protocols support their implementation without authentication between the peers taking place, only some of the more widely used protocols even support authentication. The inherent problem, even with authentication deployed, is the full trust relation of nodes within the communication network. Compromising a single node can in some cases already yield full control over the communication path decisions in the entire network.

In reaction to the inherent insecurity of today's computer networks, a variety of security protocols on top of the existing infrastructure has been proposed and deployed (e.g. [DR06], [DD03]). Common between all the security protocols is the protection of the communication contents (integrity) and un-tempered sender identification. Additionally, most of them offer confidentiality through encryption.

The security protocols protect their payload, but cannot influence the communication path taken by the individual packets, as the communication infrastructure is transparent to the security protocol. Accordingly, they can only offer to discard the communication data items (packets) when they were modified by an unauthorized party during transport. None of the communication protocol suites in wide use today offers the ability to choose a different communication path or channel when the current channel is misbehaving maliciously, and none of them could, as it would open the possibility to attackers for direct misuse of this very functionality.

2.4 The Goal of Network Forensics

Based on the observations above, the goal of network equipment forensics must be the ability to extract meaningful and complete evidence from network equipment. Analysis should enable the legitimate operator of the network equipment to identify a successful intrusion and show beyond reasonable doubt that an intrusion occurred. Additionally, it should be possible to identify the sender of data that can be shown to have a relation to the intrusion.

3 Network Equipment Anatomy

Network equipment, as any other computer system, is built by the basic blocks of at least one processor (CPU), several types of volatile random access memory (RAM), permanent (non-volatile) storage and interfaces to other systems. In contrast to multi-purpose computer systems, network equipment makes use of the permanent storage almost exclusively at boot time, as the storage provides the system software and initial configuration. Once started completely, all information and state is kept in RAM, with the only exception being configuration changes that are written back into permanent storage and minor data items in small non-volatile memories. Additionally, network equipment often has separate memory banks for special purposes, such as local module memory or network interface memory, all of which is completely volatile.

As outlined, almost all evidence is volatile data, which is a significant difference to general-purpose computer systems, where volatile data is only a small part of the evidence collected. The forensic investigator needs a way to preserve a snapshot of the volatile data from the device, as it contains almost all the evidence there is. The potential to collect the evidence depends on the device class of the network equipment examined. Two general device classes can be identified:

3.1 The Monolithic Device Class

Older network equipment is generally built as a combination of custom hardware designs and software specifically developed for the hardware platform. Since the network equipment market has significantly longer product cycles as other market segments in information technology, the majority of deployed network equipment still follows the monolithic design approach. Prominent examples of monolithic system designs are routers and switches from vendors like Cisco Systems (IOS based) and HP.

The majority of monolithic devices run software that is termed “firmware”. The term is used to indicate that the software is not a modular operating system, but rather a single binary program compiled into self-contained executable code, with the goal of minimizing the required amount of resources. Along with this design decision, vendors usually abandon concepts such as kernel and process separation, virtual memory protections and concurrent scheduling.

It should be noted that monolithic devices have no ability to recover from memory corruption, the by far most common critical software fault class in such environments. Due to the missing memory boundaries and process separation, the system cannot terminate and restart selected parts. Therefore, the only solution is to restart the entire device, which appears to be the generally accepted way of handling memory corruption. Forensics must deal with this behavior; especially since attacks on vulnerable code (e.g. buffer overflow exploitation attempts) cause memory corruptions.

Monolithic device firmware must support the collection of evidence through an explicitly implemented functionality, as no standard interfaces exist to introduce the feature. The functionality available is often designed for the vendor's developers and not for a forensic investigator. At the time of this writing, the author is not aware of a single monolithic network device that explicitly offers evidence collection functionality for the purpose of forensic analysis. However, developer functionality, although largely undocumented and unsupported, can be used to achieve the same goals of full volatile date collection.

3.2 The Embedded OS Device Class

The second large device class uses existing operating systems designed for embedded platforms and implements the desired functionality as part of the system. Those systems generally do have a separate kernel and user land processes, concurrent schedulers and virtual memory with inter-process boundaries. They also provide more or less open interfaces, as the embedded operating system is normally not implemented by the same vendor and is therefore designed for flexibility rather than a unique use case. Typical examples of this class are Juniper routers running a FreeBSD based system and the large group of home network equipment (e.g. wireless access points and DSL routers) running Linux.

Devices utilizing an embedded operating system as core provide better stability, as processes are usually separated from each other and can crash individually. All embedded operating systems support post mortem analysis of individual processes, which can be used by the forensic investigator to analyze and identify the cause of the crash. Unfortunately, access to post mortem analysis information is not exposed to the user or administrator by most vendors. Therefore, the forensic investigator must find other ways to access the information.

While vendors of this device class commonly do not support the addition of runtime code to their devices, it is considerably easier to do than with monolithic devices. This fact can be exploited by attackers and forensic investigators alike. While the attacker will strive to embed binary code into the operating system kernel or a critical process, the forensic investigator can use the build-in crash dump and logging mechanisms to obtain evidence from the device.

3.3 Access to Evidence

Wile the device classes are extremely different by software design and architecture, the forensic investigator will in both cases need to identify post mortem analysis features usually left for the developers of the device. Depending on the device, the feature will likely provide any of the following:

- External debugging access to the device for the purpose of obtaining a partial or complete memory dump
- A preconfigured location for full memory dumps
- A fault analysis summary in text form
- An on-system kernel debugger
- A network accessible kernel debugger

It is common to all methods mentioned, that a configuration and preparation of evidence collection must be performed before the to-be-observed event takes place. Currently, the only option left when the necessary preparations have not been completed before the event, is the immediate mirroring of the device's memory contents using methods similar to [Ap08], which is impractical for even the advanced forensic investigator.

It should be noted that none of the methods discussed in this section is currently in wide use or part of any best practice recommendation.

4 Types of Attacks

Network equipment is generally targeted by the same attack types as common operating system platforms are. However, in contrast to the later, network equipment is most commonly attacked using modified or specially crafted network communication messages. This includes modification of address caches of the ARP protocol, Domain Name Resolution or neighbor information about other network devices. Within the same class of attacks are injections of invalid information on network communication paths, commonly referred to as routing protocol attacks, where the attacker emulates network equipment himself and becomes part of the communication infrastructure. This type of attack is easily recognizable with existing tools (see 5).

The same type of attack can also be applied to influence the network equipment directly with the goal of achieving administrative access to it and being able to load modified binary code or configuration settings. For the attack to succeed, the network equipment must at some point in time request either parts of its binary code or its configuration over the network. Commonly, such requests happen at boot time and involve entirely insecure protocols, such as TFTP, in the process. If the attacker is able to redirect the communication path for the device to a system he controls, the attacker can provide arbitrary code or a configuration that will be loaded onto the device. Changes to the configuration are in almost all cases recognizable by existing tools and procedures, while changes to the runtime code require evidence collection methods as described in 3.3 and further analysis.

The second important attack type against network equipment is the exploitation of vulnerabilities in services provided by the equipment itself. The network equipment offers network accessible services, such as remote management access, protocol translation, protocol de-capsulation or informational services such as network time. Since the exposed services have to handle requests from the outside, they can be attacked by a malicious party. The types of vulnerabilities discovered in services on network equipment naturally mirror those found in other software, namely buffer overflows, integer overflows and format string vulnerabilities. All of them allow the attacker to corrupt the memory of the process exhibiting the vulnerability, hereby diverting the execution of binary code in a way favorable to the attacker.

While on multi purpose operating systems, the attacker usually injects binary code that, upon execution, will open a command line interface towards the network (so-called "shell code"), the approach taken is different with network equipment, as the attacker will want to change the behavior of the device in a non-obvious but permanent way. Therefore, the attacker modifies the binary code of the network equipment to divert execution into a small chunk of code provided by the attacker, hereby introducing additional behavior options. [Mu08] provides a discussion of such modification for Cisco IOS devices. It is not possible to discover modifications of the binary code with the regularly exposed user interface, even for the experienced network equipment operator.

5 Current Evidence Gathering

The gathering of evidence from network equipment is today limited to different types of remote monitoring and management.

5.1 Simple Network Management Protocol

The globally accepted standard for device management is the Simple Network Management Protocol (SNMP), as it allows querying exposed information from the device as well as changing the configuration settings at runtime remotely. While the protocol supports the generation of messages to a centrally located message sink in the network upon specific events, most information is obtained via query requests to the device. This limits the amount if information the operator is able to collect, since the network bandwidth is degraded by every query and therefore not available for customer traffic.

Another limiting factor, from a forensics point of view, is the type of information exposed via the SNMP Management Information Base (MIB). Vendors and network operators alike focus naturally on the network specific information and only include general system state information in the MIB. The internal state of the operating system software is not regarded as important or useful to the network operator. Therefore, SNMP rarely exposes data of that type.

5.2 Syslog

Another widely deployed information gathering approach is syslog. Syslog is a simple and purely event driven protocol that transports logging information to a remotely located host, where they can be stored in text files. Configured correctly, syslog allows for slightly better monitoring of the network device's software states than SNMP, as events can be generated by any part of the software and are free-form text strings. However, the information provided by most network equipment is still a small subset of the internal state.

5.3 Debug filters

A third method of evidence collection is employing vendor specific debug output filters. Usually available on the local or remote console of the device, vendor specific debug output can be enabled for specific subsystems of the software. This often provides more detailed information than SNMP and Syslog, at the expense of the device's performance. Enabling all available debug information could provide enough evidence to investigate an intrusion, but is rarely practical, since it degrades the device's performance to a fraction of its regular throughput.

5.4 Looking Glasses and Monitors

To monitor the data network equipment uses to make communication path decisions, operators use software on external hosts that participates in the routing protocols or uses mechanisms like SNMP to query the respective tables from network equipment. These so-called looking glasses and routing protocol monitors provide the operator with an overview of the network's state and greatly simplify the identification of misbehaving network equipment.

5.5 Traffic Accounting

Network operators must monitor their devices and bandwidth for signs of overuse. They also must be able to correlate network traffic to customers, since most billing models depend on the traffic generated. Therefore, almost all larger networks make use of accounting information generated by the network equipment and sent off to centrally located accounting systems. An example of such functionality is Cisco System's NetFlow.

Accounting records may include data on traffic type, source network addresses and physical ports used. In correlation with other data available, accounting records can be a valuable source of evidence, since they are collected for billing and therefore the collection mechanism has to fulfill requirements that make billing legally enforceable. Accordingly, these mechanisms are implemented so that tempering with the records is detectable and access to them is closely monitored, giving the forensic investigator a rare case of solid evidence.

5.6 Summary of Current Methods

The currently widely deployed methods allow the detection of changes in the communication paths of the network. The evidence collected by the methods described can be sufficient to show the source of the communication path changes. However, it is often hard to establish if the sender intentionally or inadvertently caused that change. This is a design limitation of currently deployed communication networks. However, it can be stated that evidence collection and therefore the respective analysis can be conducted in reasonable depth, if the attack only influenced the communication path logic of the network equipment.

Whichever of the currently deployed methods is considered, none of them provides detailed enough evidence to perform an analysis regarding attacks that affect the network equipment's software. It is therefore hard to identify malicious network equipment software changes or single out software failures caused by intentional attacks against those caused by functionality issues. In the rare case that logging information provides enough indicators to conclude an attack could have taken place, the forensic investigator is currently at a loss for means of collecting the full evidence for further analysis.

6 Forensics for Monolithic OS Designs

To augment the methods described in 5, the network equipment family with the largest market share, Cisco System's IOS routers and switches, was analyzed for possibilities of significantly improved evidence collection mechanisms and procedures. The focus is on a separation of the evidence collection and analysis steps.

Evidence collections should be simple and easy to understand, preferably using a mechanism comparable to accepted methods such as hard drive imaging. The result should be unaltered and complete. The analysis tools, working on copies of the evidence, shall allow the dissemination of the evidence to a point where intrusions can be clearly detected and shown, including any modifications of binary code or critical data structures.

6.1 Evidence Collection through Memory Dumps

As already mentioned in 3.3, access to evidence in closed network equipment is normally achieved by means originally developed for the vendor's developers. In the case of Cisco IOS devices, the functionality used is the creation of full system memory dumps. The devices can be configured to write an unaltered memory dump of all mapped system memory onto a flash memory card or a remote host.

The advantage of the method is the ability to create memory dumps at any point in time during runtime of the device as well as upon critical software failures that cause the system to restart. In the event of a system restart, the memory dump is the only way to preserve any evidence from volatile memory.

The disadvantage from a forensics point of view is the reliance on functionality of the potentially compromised system to obtain the evidence. Technically, the modified device software can write arbitrary data instead of the memory dump, if the right code region has been modified by the attacker. Additionally, carefully crafted device crashes can leave the device in a state that doesn't allow it to write memory dumps anymore, as all information about memory organization and structure is already lost.

6.2 Evidence Collection through GDB

Cisco System's IOS devices offer another undocumented developer access to their system: the GNU Debugger (GDB) serial debug protocol, a widely used text based protocol for driving embedded kernel debuggers. The protocol relies on a GDB debugger stub in the target device; a very small code element that handles basic commands received from the serial console port.

Using the GDB debugger stub, the forensic investigator can obtain partial or full memory dumps from an IOS device by connecting a respective dump tool and halting the device's execution. Only the very small GDB stub will be executed when the device is in kernel debugging mode.

This presents another method of obtaining on-device evidence. The method's clear disadvantage is that it cannot be executed automatically upon critical system failures and requires a specialized software tool to be connected through the serial console port. Additionally, the method is extremely slow and therefore time consuming. Its advantage lies in the possibility to query arbitrary memory addresses and as an alternative for the full memory dump process, due to the GDB stub's functionality of halting execution and reporting CPU exceptions directly to the connected inspection tool, presenting a way to capture otherwise unhandled cases.

6.3 Analysis

Evidence colleted using the methods presented in 6.1 and 6.2 consists of a raw copy of the system memory. To analyze the obtained data, an analysis tool must be able to determine the former memory structure and gradually recreate abstraction from pure data. Based on the recreated data structures and their relation to each other, the analysis tool must be able to distinguish patterns of abnormal behavior, memory corruptions, modified binary code and additional functionality running on the inspected device at the point of the evidence collection. Such analysis tool has been developed and made available for Cisco IOS devices [Li08].

6.4 Abstraction Recovery

For the initial recreation of the memory layout, an analysis tool can utilize information from the device's software. The device software is in almost all cases available as separate package, allowing users and operators to upgrade their devices. The software distribution contains all information about the internal memory layout of the device. Using reverse engineering techniques, an analysis tool can identify the intended memory layout of the running system and compare it to the obtained evidence as well as to the limits the CPU platform of the device imposes on memory layouts.

Given a successfully reconstructed memory map, the analysis tool can search for patterns of well known data structures. The information base must be created manually from reverse engineering results, as the device vendor usually does not publish much detail about internal data structures. When data structures are found, their information can be added to a dynamic knowledge base, which in turn can provide more abstract information to higher level analysis code.

6.5 Backdoor Detection

Once sufficient abstraction and detail is built up, the detection of attack footprints can be performed. The tests performed are, by nature, very specific to the device type and device software employed. A generic test applicable to all device classes is the comparison of the binary code segments identified in the evidence to the same segments in the device's original firmware. If a difference is detected, the runtime code was modified. Since self-modifying code is rarely used on embedded devices in general, the differences highlight areas for future reverse engineering to the forensic investigator [Ci08]. Other detections include the inspection of process stacks for signs of code redirection into non-code areas of the memory, heap integrity checks and extraction of background scripts.

6.6 Traffic Extraction

Given a full memory dump of the device in question, a successfully implemented advanced forensics method is the extraction of data packets residing in the device's memory at the time of the evidence collection. In case the evidence was produced due to a crash of the device, this greatly increases the chances to identify both the offending data packet and the source of the offense.

In a first step, responsible data structures are inspected for information on the memory location of network protocol packets. Identified data packets can easily be extracted into a common file format for network traffic analysis, such as the widely used PCAP format, and analyzed by the forensic investigator further in publicly available tools (e.g. Wireshark).

Once this list is obtained and verified, the process stacks and other user memory areas can be inspected for data that directly references the responsible network data structures. If the device's firmware tracks ownership and references of memory blocks, these can also be used to reference code functionality to data handled. If relations between binary code of the device's software and a data packet can be shown, the forensic investigator can inspect the disassembled binary instructions for signs of a vulnerability exploited by the data packet in question. This allows to uniquely identifying vulnerabilities exploited, even if they are not previously known to the vendor or the general public.

7 Conclusion

The Internet Protocol based infrastructure used worldwide today is unable to cope with single compromised network nodes by itself, not even with manual intervention. Security protocols allow the detection of mischief in such networks, but don't present a way to resolve the issue once detected. Given the tremendous importance of network communication and availability in today's world, it can be argued that users will be forced to accept to work across compromised infrastructure if the alternative is to be offline.

Forensic investigators must be urgently put into the position to perform on-demand in-depth analysis of potentially attacked and compromised network equipment, as it is likely that attacker focus will shift in that direction. The lack of methodologies and tools for performing forensics on network equipment must be overcome to be able to detect, analyze and trace infrastructure attacks as well as attackers.

The methods presented in this paper have been shown to work well for the specific subset of Cisco IOS network devices, but may be applicable to a large range of monolithic devices, given comparable memory access methods. The incremental analysis approach allows producing compact and targeted information for the forensic investigator and preserves the evidence in its unmodified form. While the developments in this area are fairly recent, the author believes that network infrastructure forensics abilities will become crucial in the future.

8 Bibliography

[Wi08] Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Computer_forensics

[Di08] Jerry Dixon, National Member Alliance's Vice President for Government Relations, Infragard: Keynote - Quest for the Holy Grail, BlackHat Briefings Washington DC, 2008

[Ph00] Anonymous: Phrack Magazine, Volume 0xa Issue 0x38, 0x10[0x10], 05.01.2000

[Fx02] FX: Burning the bridge: Cisco IOS exploits, Phrack Magazine, Volume 0x0b, Issue 0x3c, Phile #0x07, 28/12/2002

[BC04] Brown, D.; Clore, E.: Honeynet Scan of the Month 31 submission, http://honeynet.opensourcecommunity.ph/scans/scan31/sub/doug_eric/, 2004

[Te06] Anonymous: Forum entry "GET /level/16/exec/-///pwd HTTP/1.0 is this a hack attempt?", http://www.techienuggets.com/Comments?tx=420, 2006

[CVE01] Common Vulnerabilities and Exposures: CVE-2001-0537 "IOS HTTP Authorization Vulnerability", http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0537

[Po81] Postel, J (Ed): Internet Protocol, DARPA Internet Program, Protocol Specification (STD0005), September 1981

[DR06] Dierks, T.; Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1, RFC4346, April 2006

[DD03] Doraswamy, N., Harkins, D.: IPSec – The New Security Standard for the Internet, Intranets, and Virtual Private Networks. Second Edition Auflage. Prentice Hall, 2003

[Ap08] Applebaum, J. et. al.: Lest We Remember: Cold Boot Attacks on Encryption Keys, Princeton University, Electronic Frontier Foundation, Wind River Systems, 2008, http://citp.princeton.edu/memory/

[Mu08] Muñiz, S.: Killing the myth of Cisco IOS rootkits: DIK (Da Ios rootKit), Core Technologies, March 2008

[Li08] Lindner, F.: Developments in Cisco IOS Forensics, January 2008, http://www.recurity-labs.com/content/pub/Recurity_Labs_Whitepaper_Cisco_Forensics.pdf

[Ci08] Detected Cisco IOS rootkit using text segment comparison: http://cir.recurity.com/cir/case.ashx/120EF269A5BC2320730E60289A4B84D9047CECEE/report-detailed.html