Recurity Labs GmbH

http://www.recurity-labs.com

July 26, 2009

Cisco IOS Router Exploitation

A map of the problem space

Abstract

This paper describes the challenges with the exploitation of memory corruption software vulnerabilities in Cisco IOS. The goal is to map out the problem space in order to allow for the anticipation of developments in the future, as current research suggests that exploitation of such vulnerabilities in the wild is not currently the case. By understanding the challenges that an attacker faces, defensive strategies can be better planned, a required evolution with the current state of Cisco IOS router networks.

Author

Felix 'FX' Lindner

Head of Recurity Labs

1 Introduction

Successful exploitation of software vulnerabilities in Cisco IOS has been shown independently by different researchers and groups in the past, employing different techniques and basing of different assumptions. Notable incidents using targeted exploits against Cisco IOS vulnerabilities, known or unknown, have however not been registered by the security community at large.

With the development of the Cisco Incident Response tool and free on-line service[1], Recurity Labs aimed at the identification of successful compromises of Cisco IOS routers. Since the initial offering of the service, it became apparent that attackers targeting network infrastructure equipment still rely largely on mis-configurations and functional vulnerabilities, such as CVE-2008-0960. This observation indicates a fundamental difference between infrastructure attacks and attacks against network leaf nodes, such as servers and clients of any kind.

This paper will highlight reasons for the lack of binary exploits and which developments will herald the dawn of reliable remote exploitation of Cisco IOS based network infrastructure equipment. The author strongly believes that eventually, attacks on network infrastructure will use binary exploitation methods to massively gain unauthorized access. Therefore, research from the offensive point of view must be conducted and published, in order to allow the defenses to be chosen in anticipation of such future developments.

1 Recurity Labs CIR, http://cir.recurity-labs.com

2 Available Vulnerabilities

The first observation about Cisco IOS vulnerabilities is, that there is only a small number of them published. Cisco Systems' Product Security Advisory listing[2] mentions 14 vulnerabilities for Cisco IOS in 2008. Almost all of the advisories suggest that exploitation of the described issues will at maximum impact cause a denial of service condition. At closer inspection, it appears reasonable to accept that most of the published vulnerabilities are not a form of memory corruption vulnerabilities but rather malfunctions caused by insufficient handling of exceptional states in processing of certain types of network traffic.

2.1 Service Vulnerabilities

In the realm of network leaf nodes, vulnerabilities in network exposed services are the most powerful points of entry for the attacker. A network exposed service suffering from a memory corruption vulnerability, preferably before performing any authentication, is the primary target for any exploit developer. Since the purpose of any server on the network is to expose services, attackers have historically focused their efforts onto finding vulnerabilities in them.

With the widespread adoption of firewalls, for both enterprise networks and personal computers, the exposure of potentially vulnerable services has massively decreased. Attacker focus has shifted onto the client-side, where untrusted data is constantly handled by a human user, may it be through the delivery of email attachments or through visiting a web site. Attackers can execute even more

2 http://www.cisco.com/en/US/products/products_security_advisories_listing.html

control over a human controlled web browser than they can over an autonomously running service.

Cisco IOS can operate as a network server and network client respectively. IOS network services include a HTTP server for configuration and monitoring, a HTTPS server for the same purpose, Telnet and SSH remote access, a FTP and a TFTP server for remote flash file system access. Memory corruption vulnerabilities in the HTTP, FTP and TFTP services have been identified in the past and proof-of-concept exploits have been developed and published.

For attackers seeking to gain control of important network infrastructure, such services are not of interest, as well-managed networks will not make use of such services on their core routing infrastructure.

Routers also need to expose services specific to their purpose. This includes services for routing protocol communication (EIGRP, OSPF, ISIS, BGP) as well as network support services, such as DHCP relaying and IPv6 router discovery. In contrast to the aforementioned HTTP and FTP servers, these services are required in most network designs and will be available on a large portion of the networking equipment. However, as most routing protocol services are vulnerable to spoofed routing protocol announcements (unless configured to use MD5 authentication), they are often guarded and rarely exposed to remote networks, e.g. the Internet.

The Cisco IOS implementation of the BGP service is a good example, in which the service will not be visible as such to any remote network node. BGP requires a TCP session between two configured peers. If such TCP session is requested from a system not configured as a peer on Cisco IOS, the router will respond with a TCP RST packet, exactly as if the service is not available or configured on the router at all. This simple design reduces the attack surface of the BGP service on Cisco IOS to attacks from systems that were configured as peers by the networking engineer.

Other routing specific services, such as OSPF and EIGRP, require the network traffic to be received on an IPv4 multicast address, effectively making sure that the sender is within the same multicast domain as the receiving router. For an attacker on the Internet, such services are of little use as targets, since they are effectively not reachable from the attackers position.

A notable exception from this list is the Cisco IOS IP options vulnerability[3], where the handling of several IPv4 protocols was implemented incorrectly. Here, the protocols affected were commonly handled when addressed to an IOS router (e.g. ICMP echo requests) and the code generating the response was suffering from a memory corruption vulnerability in the form of a stack based buffer overflow. It is those rare vulnerabilities in services that every network uses and that are reachable all the way across the Internet, that pose a significant threat to Cisco IOS.

In the recent past, Cisco has started to add enterprise and carrier services to IOS that will be implemented more widely once the IOS versions incorporating them are considered stable enough by networking engineers. Those new services include[4] a rapidly growing set of VoIP services, Lawfull Interception, SSL VPN termination, Web Service Management Agent (allowing configuration of Cisco IOS through a SOAP Web Service), XML-PI and H.323. The more these services are adapted in

3 cisco-sa-20070124-crafted-ip-option

4 http://www.cisco.com/en/US/docs/ios/12_4t/release/notes/124TNEWF.html

enterprise and carrier networks, the more attack surface the individual routers expose.

Once these new services are deployed in a wider scale, the playing field will significantly change with regard to attacks using binary exploitation. Therefore, it should be thoroughly considered by network security engineers if they want additional services on the Cisco IOS routers in their domain of influence. If such new services are unavoidable for any reason, monitoring and post mortem analysis must match the new exposure level of the network infrastructure.

2.2 Client Side Vulnerabilities

Cisco IOS suffers from client side vulnerabilities as much as any other network client software, probably even more so. However, the vulnerabilities identified in the past have rarely been even reported to Cisco PSIRT for fixing. The reason is probably that client side vulnerabilities are only useful to attackers if the client is actually used. And since it's likely that Cisco wouldn't care about client vulnerabilities, the incentive to report them is low.

Network engineers and support personal doesn't usually use Cisco IOS routers to access other services on the network. Accordingly, attackers can't exploit the vulnerabilities, even if they are known to them.

This situation might also change with the introduction of new functionality into Cisco IOS. It depends on the level of control an attacker can execute over the functionality on IOS remotely. If, for example, the attacker can cause an IOS router to connect to a third party HTTP server for any purpose (e.g. VoIP services), the whole range of vulnerabilities in HTTP client code becomes available as an attack vector.

But up until now, client side vulnerabilities have not played any role in Cisco IOS attacks.

2.3 Transit Vulnerabilities

From the attack vector point of view, the most powerful vulnerability class in Cisco IOS are vulnerabilities, which can be triggered by traffic passing through the router. For the sake of terminology, we will call them Transit Vulnerabilities.

Transit Vulnerabilities are extremely rare. Any router is built with the goal of forwarding traffic as fast as possible from one interface to another. Consequently, the number of bytes per packet that are inspected before making the forwarding decision is kept to an absolute minimum. In any routing device above the access layer class, routing decisions can often be taken on the interface or line card already. In those cases, only the first packet of a communication is inspected by higher level software and all following packets are processed in hardware, hereby eliminating the need to even inform the main CPU of the machine that a packet passed through the system.

Considering the above, there are situations in which a packet gets “punted”, which is Cisco slang for pushing packets up from fast forwarding mechanisms like CEF to “process switching” or “fast switching”, which use the main CPU for forwarding decisions. Such situations of course include all traffic destined for one of the router's interface addresses, but this wouldn't be transit traffic. More interesting cases are IP fragment reassembly, packets with IP options as well as IPv6 packets that feature hop-by-hop headers, which need to be processed.

So far, no true Transit Vulnerability is known to the author. If one would be discovered and successfully exploited, it's effects would be devastating, especially if the vulnerability is triggered after the forwarding decision was made and the traffic is forwarded to the next hop.

3 Architectural Issues

The lack of reliable binary exploits against Cisco IOS is also caused by the architecture of the target software. IOS is a monolithic binary running directly on the hardware of the respective router platform. While it features the look and feel of a head-less operating system, IOS is better compared to a multi-threaded UNIX program.

IOS uses a shared flat memory architecture, leveraging the CPUs address translation functionality only to the point where a global memory map comparable to any UNIX ELF binary is created. IOS processes are little more than threads. Every process has its own CPU context and a block of memory as stack, but no further separation from other processes is enforced or enforceable. All processes on IOS share the same heap, a large doubly linked list of memory blocks, referenced by a couple of smaller linked lists for free block tracking. Depending on the router platform, there are additional memory regions that are all accessible from every piece of code running on the machine.

IOS uses a run-to-completion scheduling for its processes. All processes that receive execution must return to the scheduler in due time, in order to allow the execution of other processes. In case a process fails to return to the scheduler, a watchdog, supported by CPU hardware, is triggered and causes a so-called Software Forced Crash, which reboots the router.

Cisco IOS routers generally run on PowerPC 32 Bit or MIPS 32 or 64 Bit CPUs. Both CPU families support privilege separation for implementing kernel vs. user land execution. None of these features have been observed to be used in IOS so far. Any execution is performed on the highest privilege level the CPU supports, commonly referred to as supervisor level.

As a consequence of the architecture discussed above, the default behavior in case of a CPU exception or software detected but unrecoverable data structure consistency problem is to reboot the entire device. The architecture of IOS does not allow for any type of partial restart, since a misbehaving process could have already written into any writable memory area without the CPU noticing. Therefore, the only safe action is to reboot the entire system.

This behavior increases the difficulties for reliable exploitation of memory corruption vulnerabilities.

On common operating system platforms, primarily the Windows platform, using CPU exception propagation as a way of gaining code execution is a well established practice. On Cisco IOS however, every CPU exception will cause the machine to reboot. This might appear as acceptable for an attacker, but given that network infrastructure of any measurable importance is monitored for crashes and reboots of its components 24 by 7, it raises the bar for reliable exploitation.

The same problem also concerns any shellcode that would be executed once control over the instruction flow is obtained. Not only should the shellcode not trigger any CPU exception during its execution, it must also clean up and attempt to return execution to the exploited IOS process in order to allow normal processing to continue.

Finally, the allocation of process stacks on the common heap results in another challenge for exploitation. Stack based buffer overflows are the most simple and versatile memory corruption vulnerabilities, and IOS is not any different in that respect. Unfortunately, the stacks allocated by default to an IOS process are relatively small (6000 bytes) and the call graph of functions within the code is relatively short, so that buffers that could overflow are often close to the upper bound of the stack and hence the heap block. Overflowing the buffer with too much data will often cause overwriting of the next heap block's header. Once the heap header is destroyed, any allocation or deallocation of memory by any process on IOS will trigger a partial heap integrity check and cause the router to reboot when the corrupted heap header is spotted.

Additionally, IOS features a process called CheckHeaps, which will periodically (every 30 seconds) traverse the entire heap linked list and verify its integrity, also causing a reboot if any inconsistency is found.

Given that both CPU families in Cisco equipment employ fixed size 32 Bit instructions, a stack based buffer overflow is often hard to exploit within the bounds of available space up to the header of the following heap block.

4 The Return Address

Cisco IOS images are loaded similar to a regular UNIX program in ELF format. When initialized, the memory is separated into read-only sections for program code and read-only data as well as read-write sections for the data region and the common heap. Ignoring other memory areas that are not executable, such as the so-called IO-Memory, an area dedicated to packet handling on the router, the image's internal layout is the only deciding factor on the resulting memory layout on the router.

This poses a tremendous challenge for the exploit developer when control over the instruction pointer is achieved: Where should it point to?

Since the stack of any IOS process is an arbitrarily allocated block of memory on the heap, its location is random enough to make it unpredictable. Techniques like Heap spraying only apply to situations where the attacker executes a large amount of control over the target, which is clearly not the case when attacking networking equipment. This leaves only the class of “code reuse” methods, which use existing code on the target to perform their initial bootstrapping before running attacker provided code.

4.1 Return into Known Code

Using any “code reuse” method requires to know the exact location of the code that should be reused. This holds true for calling known functions with an attacker prepared stack layout as well as for the technique known as Return Oriented Programming[5].

Unfortunately, Cisco IOS images are built individually by Cisco engineers and the image content, and hence internal layout, depends on:

- Target Cisco platform
- Major Version
- Minor Version
- Image Train
- Release Version
- Combination of features

When querying the Cisco Feature Navigator[6] for all known images that support a feature known as “IP

5 https://www.blackhat.com/presentations/bh-usa-08/Shacham/BH_US_08_Shacham_Return_Oriented_Programming.pdf

6 http://www.cisco.com/go/fn

routing" (the most basic functionality on any router), the result shows 272722 different IOS images at the time of this writing. Taking the 7200er platform alone as an example,15878 images are available. This presents a higher uncertainty about the memory layout than any of the address space layout randomization (ASLR) implementations that are in use today on common operating system platforms.

Additionally, and in contrast to ASLR, an attacker wishing to leverage "code reuse" on Cisco IOS images will need to have a copy of the same for analysis purposes. However, IOS images are actually a product of Cisco Systems and therefore not legally available for free. Some special image series are not available to anyone outside special interest groups, such as the military or law enforcement.

4.2 Returning to ROMMON

To overcome the problem of high uncertainty in memory layout, a memory section is required that allows execution of its contents as code and preferably already contains code at a stable location.

Cisco routers use a piece of code called ROMMON as the initially available code to execute after the CPU has been reset. ROMMON is screened into memory at the initial reset vector and serves as bootstrapping code for IOS. The ROMMON also contains functionality for disaster recovery (allowing to load a new image when the available one is broken or corrupted) as well as some basic setup functions.

On the Cisco platforms reviewed by the author, ROMMON is placed the uppermost memory regions after the CPU's virtual addressing and address translation has been initialized to match the IOS image's memory map. Therefore, its location is known and invariant.

The factor decisive for using ROMMON as return point is the relatively small number of versions published for each router platform. Taking the 2600 access router platform as an example, there are 8 different versions of ROMMON known to the author. With a few exceptions due to hardware support added into later ROMMON versions, deployed infrastructure equipment rarely receives ROMMON upgrades. Therefore, the large majority of the routers runs the ROMMON version that was current when the equipment was manufactured. Since such equipment is usually ordered in bulk when new infrastructure is installed, the versions will neither differ nor will later versions be very common, because the initial version will be sold the most.

Applying Return Oriented Programming to the code found in ROMMON, it has been shown[7] that 32 Bit arbitrary memory writes to the memory area that contains the exception handlers can be used on PowerPC and MIPS based Cisco routers to gain reliable code execution with stack based buffer overflows.

The method employs returns into function epilogues that perform a memory write to a register that was controlled by the attacker already, with the contents of another register under the attacker's control. On PowerPC, these are usually registers that, by the PowerPC ABI, should be saved across function boundaries (i.e. R13 to R31).

Beneficial for the attacker is the fact that ROMMON also contains code used to disable the instruction and data cache of the CPU, allowing to write data and directly afterwards execute it as code without cache consistency issues.

7 http://cir.recurity.com/wiki/PPCreliableCodeExec.ashx

4.3 ROMMON Uncertainty

The method of employing ROMMON as the vehicle of choice for more reliable code execution has a couple of drawbacks.

The first is connected to the uncertainty about how many versions of ROMMON are to be found in the wild when dealing with any Cisco router platform. Low end routers usually don't support upgrading ROMMON, so not even the vendor web site will give an indication on which versions are to be expected. Even when updates are available for the platform, it is not known how many other versions were initially shipped.

Second, the exploit developer will need to obtain a copy of every ROMMON he knows of for the platform he is targeting. Since the initial versions (the ones with the widest distribution) are never available for download, this involves obtaining temporary access to routers that run the most common versions. Additionally, it will be generally hard to say which is the most common version.

It should also be noted that an attacker will still need to know the hardware platform of the Cisco router he is attacking, since this will decide the ROMMON memory layout as well as the instruction set for the attacker provided code (i.e. PPC vs. MIPS).

The third issue with the ROMMON based method is the inability to ensure the right addresses are used before the exploit is launched. Applicable vulnerabilities and reliable exploits against Cisco equipment have a high monetary value at the time of this writing. Accordingly, attackers in the possession of such an item would rather like to ensure that they will use the right set of addresses before launching the exploit and risking the target to reboot, giving away their presence as well as the valuable exploit itself.

4.4 Code Similarity Analysis

Another approach actively researched by the author is finding similarities across a set of IOS images. While the images theoretically differ completely from each other, it can be assumed that images of the same version but different feature sets, as well as images with the same feature set and slightly different release version may contain code sections that differ only slightly.

At the time of this writing, outcomes of this research are not available yet.

5 Shellcode

The final area in which exploitation of network infrastructure equipment differs significantly from exploitation of network leaf nodes is the attacker provided code.

It is common practice within exploitation of network leaf nodes to spawn a shell back to the attacker, either by making it available on a TCP port or by connecting back to the attacker's host. Similar shellcode has been shown for specific IOS images.

An alternative method, which proved to be more reliable than a “bind shell”, is to rely on the fact that almost any Cisco IOS router will already have a remote shell service configured, either via Telnet or SSH. By removing the requirement to authenticate on said shell, either through patching the code that performs the validation or by modifying entries in the data structures that hold the authentication configuration for remote terminals, it is easy to use the existing service for obtaining a remote shell.

Once a privileged interactive shell is obtained on a Cisco IOS router, the attacker can use all the functionality provided by IOS to fulfill his goals. Alternatively, the attacker provided code can already implement the desired application of IOS functionality, without requiring the attacker to connect to a shell and manually change the configuration.

However, this brings up the question of what could be of interest to an attacker?

5.1 Arbitrary Services using TCL

An increasing number of deployed IOS images support scripting from the command line by using TCL scripts. This feature is mostly used to automate monitoring of the device or automatically act upon certain log messages encountered.

However, it has been shown[8] that TCL scripts can be used for implementing more complex services, including implementation of Botnet clients or making the router participate in the Twitter service.

As the number of TCL controllable functionality in Cisco IOS increases, attackers may well find everything they need for the purpose of "regular" on-line crime and fraud by using customized TCL scripts for IOS.

5.2 Ultimate Sniffer

It is naïve to assume that a router under an attacker's control can easily be turned into the ultimate password sniffer. Referring back to the packet handling of IOS discussed in 2.3, only a fraction of the traffic is ever visible to the main CPU, which is the context of the executed attacker code.

Additionally, the run-to-completion scheduler will make the implementation of a password sniffer most challenging. Considering that productive IOS routers of larger types are usually running with their regular CPU utilization well within 40%-60%, the additional load would immediately kill the machine. Even if the CPU resources would be sufficient to perform password sniffing, the sudden increase in traffic latency due to all packets getting "punted" will immediately attract the network engineers attention.

This is an area where the introduction and wide scale deployment of lawful interception enabled IOS images within carrier networks may potentially have an impact besides the intended. LI functionality is required to be transparent to the network engineer, i.e. he should not be able to observe an active interception. LI is also designed to most efficiently and selectively copy traffic matching a certain interception rule to a third party as well. When this functionality is available within the image and the attacker is aware of how to control it (e.g. by calling the appropriate functions that would otherwise be triggered through the SNMPv3 CISCO-TAP-MIB and CISCO-TAP2-MIB), he is in the position to selectively monitor traffic that is of interest to him on any remote system that the compromised router can reach.

5.3 MITM Tool

Similar to the sniffer scenario, a compromised router of sufficient importance cannot be easily converted into a MITM tool, as the same limitations apply that prevent if from being the ultimate sniffer.

However, it is possible with some lines of Cisco IOS routers and images to use Access Control Lists (ACL) to match certain traffic and apply special behavior to it. This functionality could be used within

8 http://ph-neutral.darklab.org/talks/christoph.html

shellcode to obtain packets that contain information relevant to the attacker, with the strict limitation that the first packet in the conversation must already contain that information of interest. Since the first packet is very likely to get “punted” anyway, the performance impact should be minimal.

As an example, any protocol that relies on a sequence number, query ID or other value only known to sender and receiver to prevent spoofing (e.g. TCP, DNS) could be matched and the relevant number pushed out to the attacker. In this scenario, the attacker would be able to arbitrarily spoof DNS replies or inject data into TCP sessions, since the secret value is now know to him.

5.4 Selective Redirection

One of the more trivial uses of a compromised router is to selectively redirect clients that attempt to communicate with a specific IP address or range. This is part of the core functionality of a router and therefore does not pose any problems to the attacker, while being relatively hard to identify for the network engineer when not done by configuration changes.

Selective redirection is known[9] to be a simple and effective tool to prevent regular users from using web sites and services with encryption (HTTPS), as most users first hit the unencrypted HTTP port and expect to be redirected but fail to recognize when that's not the case.

5.5 Other Uses

This paper highlights a number of issues with exploitation of Cisco IOS routers. Since stable exploitation is a prerequisite to deploying smart shellcode, it is likely that, once some of the problems discussed have been solved, more practical approaches to the use of compromised routers are developed.

6 Conclusion

As interest in attacking network infrastructure equipment increases, new players in the field will face the issues discussed in this paper, as well as some that are unknown to this day. It is the strong believe of the author that only be realizing the problems of the offensive party, that we can anticipate potential ways the attackers will be taking in order to circumnavigate or solve these problems.

When reliable exploitation and independence or semi-independence from the vast variance of IOS images has been achieved by an attacker, enterprise and carrier networks need to be prepared to change the way and frequency they select and deploy IOS images. This can only be achieved if Cisco changes the way they release images, providing clear and proven update paths that allow a large organization to update to a new IOS version without the issues normally connected with such exercise.

In today's Cisco router networks, updating breaks the network's functionality, preventing networking engineers from maintaining recent versions of IOS on their routers. This fact is leaving the network vulnerable to attacks, because the availability of the network is of significantly higher value than the integrity of its core nodes.

9 http://www.blackhat.com/html/bh-europe-09/bh-eu-09-speakers.html#Marlinspike