The User-Space VPN and OpenVPN

Understanding the User-Space VPN – History, Conceptual Foundations, and Practical Usage – By James Yonan

What is a VPN and how is it different from other security software?

- Fundamentally, a VPN is a set of tools which allow networks at different locations to be securely connected, using a public network as the transport layer.
- VPNs use cryptography to provide protections against eavesdropping and active attacks.
- VPNs are most commonly used today for telecommuting and linking branch offices via secure WANs.

The Wide area network before VPNs

- Firms would spend thousands of dollars per month for private, dedicated circuits to link branch offices.
- The rise of the internet created cheap but insecure bandwidth.
- The VPN concept was to produce the virtual “dedicated circuit”, pump it over the internet, and use cryptography to make it secure.

A brief history of VPNs

- IPSec was the first major effort to develop a standard for secure networking.
- First version in 1995.
- IPSec, like other early crypto developments, were hamstrung by export controls and insufficient processor power in the routers where they were to be implemented.
- Some components of IPSec, e.g. IKE are still in development today. Long Development time!

IPSec problems

- Slow progress resulted in a splintering of efforts during the mid-90s
- SSL was one such offshoot, developed to provide application-level security rather than network level security.
- Traditional IPSec implementations required a great deal of kernel code, complicating cross-platform porting efforts.
- IPSec is a complex production with a relatively steep learning curve for new users.

The rise of SSL and user-space VPNs.

- IPSec's slow progress and complexity caused many to turn to other solutions.
- By contrast, SSL matured quickly, due to heavy usage on the web.
- SSL runs in user space, simplifying implementation and administration.
- The so-called SSL VPN is really just a web application that tries to give users the services they need without a full VPN implementation.

Linux and virtual network interfaces

- The maturing of the Linux OS by the late 90s provided an excellent test bed for experimental networking concepts.
- One such innovation is the “tun” or “tap” interface.
- The first tun driver for linux was written by Maxim Krasnyansky.

What is a tun interface?

- A tun interface is a virtual network adapter that looks like point-to-point network hardware to the OS, such as a T1 line.
- But instead of pushing bits out a wire, the tun driver pushes them to user space.
- A user space program can open the tun device just like a file and read and write IP packets from and to it.
- A “tap” interface is a similar production, only it emulates ethernet rather than point-to-point.

How is a tun interface used to build a VPN?

- Suppose I have a tun interface on machine A, and another on machine B.
- I write a simple network application with two threads.
- Copy bits from tun device -> network socket.
- Copy bits from network socket -> tun device.
- If I run this app on machine A and B I will have constructed a very simple VPN minus the security component.

How is a tun interface used to build a VPN (continued)?

- From A I can ping the tun device on B, and from B I can ping the tun device on A.
- That ping will actually travel over the socket connection, i.e. the ping packet will be encapsulated within a UDP or TCP packet and sent between A and B.
- The problem with this very simple VPN is it's missing the security – it is what is known as a **cleartext** tunnel.

Adding security to the VPN

- The simple VPN we have constructed, tunnels a virtual network interface over a TCP or UDP connection.
- By forwarding such a TCP connection over a secure port forwarding tool such as SSH, we can build a real VPN.

Problems with using SSH to build a VPN

- The previous example has a problem, however.
- IP is what is known as an “unreliable” protocol.
- This is not a value judgment.
- Rather, it means that IP assumes that packets sent over a physical or virtual network might be lost or corrupted.
- Protocols in the IP family such as TCP try very hard to work under this assumption.

Reliable and Unreliable protocols

- TCP is a reliable application protocol that utilizes an unreliable transport layer.
- This means that your web browser (HTTP is a TCP protocol) expects TCP to handle the glitches in the connection between your client and a possibly distant web server.
- TCP does this by retransmitting packets which are lost due to network congestion.
- TCP is a reliability bridge between the application and physical network layers.

Encapsulating Protocols

- One of the cool things about networking is that you can take one protocol and encapsulate it into another.
- Getting back to our simple VPN example, we are encapsulating IP into a TCP port, then using SSH to secure that TCP connection with another remote host.
- As far as encapsulation is concerned, we are encapsulating IP (which includes TCP and UDP protocols) into TCP.

Encapsulating TCP in TCP – the problem

- There is a fundamental problem, however, in this encapsulation graph.
- TCP is designed to flow over unreliable networks. Pushing TCP into TCP means that we are nesting one reliability layer into another, essentially producing a whole level of redundancy.
- This redundancy translates into less efficiency and less robustness during congested network conditions.

Fixing the problem

- A better solutions is to encapsulate TCP in UDP.
- UDP is the “unreliable cousin” of TCP. It strips out the whole reliability layer of TCP, giving the application the responsibility to sort out problems of dropped packets, or packets arriving in a different order from how they were sent.

Why is UDP better for encapsulating IP?

- The fundamental reason is that IP was designed to flow over wires, fiber, or wireless links which are all unreliable physical media that can suffer from glitches or congestion.
- Because UDP is itself an unreliable protocol, it gives IP a transmission medium which is as close as possible to its native environment.
- Encapsulating IP in UDP is the ideal choice.

VPNs and UDP

- The modern, portable, easy-to-configure, user-space VPN has several basic properties.
- IP packets from tun or tap virtual network adapters are encrypted and **encapsulated,** onto a UDP connection, and sent to a remote host over the internet.
- The remote host decrypts, authenticates, and de-encapsulates the IP packets, pumping them into a tun or tap virtual adapter at the other end.

The VPN is invisible to applications tunneling over it.

- This user-space VPN model essentially links a local tun virtual adapter with a remote tun virtual adapter.
- One can apply routes or firewall rules to tun or tap interfaces in the same way that you can apply them to ethernet interfaces.
- Applications using a VPN would find them indistinguishable from a wide area network implemented with dedicated circuits.

Enter OpenVPN

- There are several Open Source VPNs today that follow the user-space tun/tap model.
- OpenVPN, VTun, Tinc, Cipe, and many more are being actively developed today.
- They stand in contrast to IPSec solutions such as FreeSwan which attack the problem in a very different way.

User-space Tun/Tap vs. IPSec

- There is some controversy about which approach is better.
- User space is more portable and easier to configure.
- IPSec is more complex, and offers multi-vendor and dedicated router support.
- IPSec's complexity sometimes makes it difficult for vendor A's implementation to talk to vendor B's.

IPSec in a nutshell

- IPSec is a complex modification to the IP stack itself.
- IPSec examines packets coming out of an IP interface, determines if a security association exists with the destination, and then tries to automatically encrypt packets at one end and decrypt them at the other.
- The dream of IPSec is that it just works and you never need to know it's there (this concept is often referred to as **opportunistic encryption**).

IPSec limitations

- As IPSec evolved, the internet evolved along with it.
- The IPv4 address shortage created a profusion of private networks that use NAT to access the internet through a single IP address.
- The IP address shortage also caused an increase in the use of dynamic IP addresses.
- IPSec proved somewhat inflexible to these new developments.

IPSec limitations (continued)

- Because IPSec considered the source and destination addresses to be part of the secured payload, it broke interoperability with NAT.
- Since then, the IPSec standard has tried to evolve around these limitations.
- IPSec has also been both lauded and criticized for its security.
- Sometimes such praise/blame emanates from the same individuals! (see next slide)

The “Two Minds” of IPSec -- N. Ferguson and B. Schneier

- We are of two minds about IPsec. On the one hand, IPsec is far better than any IP security protocol that has come before: Microsoft PPTP, L2TP, etc. On the other hand, we do not believe that it will ever result in a secure operational system. It is far too complex, and the complexity has lead to a large number of ambiguities, contradictions, inefficiencies, and weaknesses. [...] We strongly discourage the use of IPsec in its current form for protection of any kind of valuable information, and hope that future iterations of the design will be improved. However, we even more strongly discourage any current alternatives, and recommend IPsec when the alternative is an insecure network. Such are the realities of the world.

How does a VPN achieve security?

- A VPN must protect against **passive** and **active** attacks.
- A passive attacker is an eavesdropper who has no ability to interrupt or modify the data channel between two parties.
- Encryption is effective at defeating passive attacks.

Active Attacks

- An active attacker has the ability to insert himself into the communication channel and add, modify, or delete data packets between both parties to the channel.
- For this reason, such attacks are commonly referred to as **Man-in-the-middle** attacks.

Active attacks are thwarted through the use of **authentication**

- While many believe that VPN security is all about encryption, the larger and more difficult problem to solve is the problem of authentication.
- Authentication in the VPN context involves “signing” every packet with a secure hash, so that the recipient can prove that it originated from a legitimate source.
- Both OpenVPN and IPSec use the **HMAC construction** to authenticate packets.

HMAC isn’t a 100% solution against active attacks.

- Even after applying HMAC, we are still vulnerable to two types of active attacks:
- Replay attacks.
- Known plaintext attacks.

Replay Attacks

- Suppose an attacker was able to tap into his bank's T1 line at 3am when traffic is low.
- While observing the encrypted bits flowing across the line with a tool such as "snort", he logs onto his bank's web site and does a number of small wire transfers, observing the encrypted packets flowing over the bank's T1 line.
- He is able, by timing analysis, to gain access to a sample of encrypted packets that represent his money transfers.

Replay attacks, continued

- What if he then spams the T1 with a large number of those sampled packets.
- He doesn’t need to know the encryption algorithm, he only needs to reproduce the packets.
- If the bank is only using encryption without replay protection, they may find an unexplained deluge of questionable transfers the following morning.

Replay attacks, continued.

- The solution to the problem is to embed a unique ID or timestamp in every packet before it is signed.
- The receiver needs to keep track of this timestamp, and make sure that it never accepts a packet with the same timestamp twice.
- Both OpenVPN and IPSec implement replay protection using the **Sliding Window Algorithm.**

Known plaintext attacks.

- Getting back to our bank cracker, suppose that he makes 5 transfers of differing amounts of money.
- By analyzing the ciphertext stream over the T1 as his transfers are taking place, he is able to discern the byte offsets in the packets that represent the dollar amount of the transfer, even though the amounts themselves are encrypted gibberish.

Known plaintext attacks (continued).

- Suppose the $ amount is a 32 bit integer.
- He inserts some bogus packets onto the link with the dollar amount altered.
- He doesn't know what the final dollar amount will be when it is "decrypted" – but he knows if he tries enough values, some of them will turn out to be large and disruptive.

This would be impossible (I hope) in 2003.

- This scenario could not, of course, happen today.
- The importance of this kind of thought experiment is to show that encryption, even if it is unbreakable, is not enough to secure against an active attacker.
- Encryption must be combined with authentication (HMAC), randomized IVs, and replay protection, to protect against the previously discussed attacks.

OpenVPN and Cryptography

- Cryptography is an advanced and specialized field.
- OpenVPN takes a modular approach to cryptography.
- Most crypto functions are offloaded to the OpenSSL library.
- OpenVPN has protection against both passive attacks and known types of active attacks.

OpenVPN and keying

- OpenVPN tries to supply the best of both worlds when it comes to keying.
- Static, pre-shared keys are provided for ease of configuration.
- Full RSA PKI, through the OpenSSL library, is provided for full certificate and private key operation.
- SSL/TLS can be used for initial authentication and symmetric key exchange.

Authentication only leads into a bigger problem – key management.

- The HMAC construction is a strong and elegant contribution from the cryptography community – but it still needs a shared secret key to exist at both ends of the secure connection.
- How do two parties “bootstrap” their key exchange process in a way that protects against the exchange being hijacked by an attacker?

Enter public key cryptography.

- In the September, 1977 issue of The Scientific American, Ronald L. Rivest, Adi Shamir and Leonard M. Adleman introduced to the world their RSA cipher, applicable to public key cryptography and digital signatures. The authors offered to send their full report to anyone who sent them self-addressed stamped envelopes, and the ensuing international response was so overwhelming the NSA balked at the idea of such widespread distribution of cryptography source code. When no response was made by the NSA as to the “legal basis of their request”, distribution recommenced, and the algorithm was published in The Communications of the ACM the following year.

Public Key cryptography is really about the problem of authentication

- Since long before the age of computers, cryptography was practiced between individuals who possessed a shared key.
- The innovation of Public Key cryptography was to show how individuals could communicate securely without needing a pre-existing secure medium over which to share their keys.

Public Key technology solves the key sharing problem.

- Public key cryptography solves the problem of providing the secure medium over which the initial shared secret key can be exchanged.
- The real encryption still occurs with a shared, symmetrical key. The public key process only gives us a means of sharing this key electronically over an insecure medium.

Public key cryptography.

- Public key cryptography allows you to generate a public and private key pair.
- The private key never leaves your hard drive.
- The public key is published far and wide.
- To communicate with someone, you only need their public key.
- But once content has been encrypted with a public key, only the private key can decrypt it.

Public key cryptography and authentication.

- Public key cryptography as described thus far still has a missing link.
- How do you know that the person on the other end of the communication channel is who they say they are?
- They can present their public key, but that proves nothing about their identity.

Enter the Certificate.

- Public key cryptography and RSA pioneered the concept of secure signatures.
- I can “sign” a file with my private key.
- I can publish my public key.
- Anyone who receives the file can verify that it was signed by my public key.
- The mathematics of the algorithm behind digital signatures ensures that it would be infeasible to forge a signature without having the correct private key.

The Certificate Authority.

- The **certificate authority** (CA) is the final result in a long linkage of developments in applied cryptography that attempt to solve the problem of authentication.
- The CA has a super-secret key that they keep under armed guard.
- They have a team of investigators who verify the identity of clients.
- They then sign the keys of clients with their super secret key.

CAs Continued…

- The CA's public key becomes a public commodity, embedded in applications and operating systems.
- The CA's "root certificate" forms a the root of a chain of public keys which can be used to verify the indentity of any of the CA's clients.
- The CA solves the problem of authentication by trusted referral.
- CAs are the basis of authentication on the secure web.

Cryptography conclusion

- While OpenVPN draws heavily on the cryptography-related developments of IPSec, there are details about any encrypted communication session which cannot be hidden.
- **Traffic Analysis** is one type of attack that no internet-based, modern cryptosystem can protect against.
- But when considering the needs of most VPN users, the modern crypto technology proves more than sufficient.

OpenVPN Features

- OpenVPN tries to take advantage of all the capabilities which are possible to a user space VPN.
- Portability.
- Familiar daemon-style usage.
- No kernel modifications required.
- State-of-the-art cryptography layer provided by the OpenSSL library.

OpenVPN Features, continued.

- Very comfortable with dynamic addresses or NAT.
- Supports most operating systems in the known computing universe, including Linux, Windows, Mac OS X, the three BSDs, and Solaris.

OpenVPN's 3 tier security model

- One of the maxims of computer security is that **complexity is the enemy of security**
- One way of reducing the impact of software complexity on overall software security is to force incoming network traffic to pass through a kind of security gateway that is a much simpler piece of code than the applications behind it
- A prime example of this is the firewall.

OpenVPN's 3 tier security model (continued)

- The key is to reduce the number of lines of code which can be touched by unauthenticated packets. These fewer lines of code can then be more rigorously scrutinized for vulnerabilities.
- OpenVPN expands on the concept of a firewall, using the **–tls-auth** option to subject incoming packets to a preliminary digital signature test before they are passed on to the actual SSL/TLS code.

OpenVPN's 3 tier security model (continued)

- Tier 1 – Use HMAC-based –tls-auth option to prevent an attacker from injecting packets into the SSL/TLS subsystem.
- Tier 2 – Use SSL/TLS for bidirectional client/server authentication.
- Tier 3 – Downgrade OpenVPN daemon's privilege level using --user/--group to help contain a successful code injection exploit.

VPNs and Networking

- As much (or more) can be written about the topic of VPNs and networking as can be written about VPNs and cryptography.
- 95% of the tech support problems that people have with VPNs are with the networking and firewall layers, not the cryptography layer.
- The two major techniques for VPN networking are **routing** and **bridging**.

Bridging vs. Routing in the VPN context

- Bridging is a technique for creating a virtual, wide-area ethernet LAN, running on a single subnet.
- Routing solves the problem of a wide area VPN by using separate subnets and setting up routes between them.

Bridging Advantages

- Broadcasts traverse the VPN -- this allows software that depends on LAN broadcasts such as Windows NetBIOS file sharing and network neighborhood browsing to work.
- No route statements to configure.
- Works with any protocol that can function over ethernet, including IPv4, IPv6, Netware IPX, AppleTalk, etc.
- Relatively easy-to-configure solution for road warriors.

Bridging Disadvantages

- Less efficient than routing, and does not scale well.

Routing Advantages

- Efficiency and scalability.
- Allows better tuning of MTU for efficiency.

Routing Disadvantages

- On Windows, clients must use a WINS server (such as samba) to allow cross-VPN network browsing to work.
- Routes must be set up linking each subnet.
- Software that depends on broadcasts will not "see" machines on the other side of the VPN.
- Works only with IPv4 in general, and IPv6 in some special cases.

The nuts and bolt of bridging (1)

- Suppose you want to create a secure ethernet bridge that serves multiple mobile clients, using Linux as the server.
- First generate a bunch of persistent tap virtual ethernet interfaces on your server, using “openvpn –mktun”.
- Then use the brctl tool to bridge them together with your real ethernet adapter.

The nuts and bolt of bridging (2)

- When clients connect to the server, the tap virtual ethernet interface at their end can be assigned an IP address from the actual subnet of the physical ethernet LAN connected to the server.
- So I could have a subnet 10.4.7.0 netmask 255.255.255.0 which is a bridged ethernet.
- 10.4.7.5 could be a machine in Moscow, Idaho. 10.4.7.6 could be a machine in Moscow, Russia.

VPNs and firewalling

- The modern user-space VPN presents virtual tun and tap interfaces as VPN endpoints.
- Suppose you have a vpn network device called “tun0”
- You can apply the same kinds of firewall rules to tun0 as you could to eth0 or any other networking device.

VPNs and firewalling (continued).

- One of the more troublesome security issues of VPNs is the way that they create trusted relationships between different networks.
- This can be bad, as in the case where a worm or virus infects someone's home machine, then jumps across the VPN to corporate headquarters.
- Firewall rules applied to the VPN itself can create a trust relationship between two networks that is more than untrusted but less than fully trusted.

Future directions -- OpenVPN 2.0

- In OpenVPN 1.x, a single openvpn daemon can support a single tunnel over a single tun/tap interface, using a single UDP or TCP port for daemon-to-daemon communication.
- This model offers maximum flexibility, as the configuration for each tunnel can be customized.
- The weakness in this model is that it is hard to set up an OpenVPN configuration that handles connections from a large number of dynamic clients.

Future directions -- OpenVPN 2.0 (continued)

- OpenVPN 2.0 (currently in beta) solves this problem by allowing an arbitrarily large number of UDP clients to connect to a single openvpn daemon, which itself uses one tun/tap interface and one UDP port number.

Conclusion

- VPNs tie together concepts from cryptography, networking, and firewalls.
- VPNs can be used as building blocks to construct anything from a small secure telecommuting solution, to a large-scale secure WAN.
- The user-space VPN is an elegant solution to the VPN problem in a modular package.
- VPNs still have a long way to evolve before they are as easy-to-configure as other networking subsystems, such as IP.