Good morning. My name is Brad Whitehead. I'm the Chief Scientist at Formularity. At Formularity, we are concerned with the collection and protection of sensitive personal information. Using our forms, our clients collect sensitive financial, health, and legal information from their citizens, customers, and employees. It's our job to design our forms and collection systems to make sure that information is only seen and processed by the people for which it is intended.

Which brings me to the subject of today's talk. Every day we hear about large collections of information being hacked and stolen. We hear about zero-day vulnerabilities, and spearfishing, and trojans and worms. We seem to think that it takes a subversion of the normal information protection privacy processes to steal this information. What we ignore it that the normal, standard, accepted processes for safeguarding information are already broken and totally inadequate. It's not that we don't know the process it broken – everybody in this room is already aware of the flaws. We just find it easier to "do it like it's always been done" and ignore the consequences. After all, we just did what the law required of us and nothing more. Harsh words, but let's examine the realities.

<Slide Change to HIPAA>

In the United States, we have two “gold standards” of how sensitive information should be handled. Both our Health Insurance Portability and Accountability Act (HIPAA) and our Payment Card Industry (PCI) defined best practices as the encryption of data in motion and the encryption of data at rest.

<Slide Change to Safe Harbor>

These best practices, especially HIPAA, provide the IT industry with a “Safe Harbor”. If you follow HIPAA and PCI guidelines, you have a reasonable defense in court.

<Slide Change to Inquiring Minds>

Now, inquiring minds want to know about the edge case when data transitions from motion to rest! OK, I started life as a coder and it has always continued to color my perspective on the world. I learned early on that most programming problems arise from the boundary condition or the edge case. It's easy to know what happens when a variable is positive or when it's negative. But what happens when it's zero? Or when the computer representation of the variable overflows? So why doesn't HIPAA or PCI tell us how to protect information when it transitions from motion to rest? We'll get back to that little mystery later. For now, it's just go with the two conditions of data – Motion and Rest

<Slide Change to Simple Transaction>

Let's walk through a simple transaction. Our user is at work and has been asked to fill out a form on the website of the company's health insurance provider. It's company business, so the user is permitted to use company IT assets and to do the work on company time. Our user is intelligent and informed. He or she knows that the browser should have a lock indicator on the address bar before any personal information is sent over the web. The user also knows to confirm that the correct address for the insurance company is visible in the address bar. He or she has carefully typed that information in themselves. The user is smart enough to know to never click on a link in an email! So, the user has a lock and feels that they are safe in sending their personal information. Is this the case? Actually, this connection has a number of leaks. How many? Let's count the ways!

<Slide Change to Leak #1>

OK, before the information even leaves the user's computer, the secure connection to the insurance company website is broken and the user's sensitive health insurance is available as plain text. Seems that the user has a normal anti-virus suite installed on their computer. As part of the installation of this AV suite, a trusted signing certificate has been added to the browser's certificate store. Using this bogus signing certificate, the AV suite can pretend to be the insurance company. It becomes a "man in the middle" to inspect the information coming from the insurance website to be sure it isn't malicious. It breaks the secure connection with the best intent, but it still weakens the privacy of the information. What if the AV suite is a malicious decoy? Or, since the AV software has access to all the web traffic, a malware program need only target the AV program to gain access to the same information. Let the AV software do the heavy lifting and the malware can skim off the cream.

<Slide Change to AV Cert>

Here's an actual example of a bogus certificate being used by an AV product to intercept web connections. I'm quite sure that Accenture doesn't use Avast Software for their SSL certificates! But if you didn't take the extra steps to view the certificate, you'd only see the normal green lock and never know that your SSL connection was compromised.
Now, I'm not singling out Avast. They make a good product and virtually all the popular anti-virus suites use this bogus signing certificate process to provide the user protection from malicious web content.

<Slide Change to Leak #2>

The user's company is worried about proprietary information being accidentally sent out through email or web connections. So they have installed a Data Loss Prevention (a DLP) SSL Proxy. All of the company's web traffic is send through this proxy. The SSL proxy uses the same bogus signing certificate process as the anti-virus suite did. There are a large number of vendors that sell this type of DLP software, including Bluecoat and Intel. This practice is so prevalent that there is an IEEE Special Advisory Group dedicated to trying to standardize the technology behind DLP SSL Proxies. Formularity is a member of this IEEE Encrypted Traffic Inspection working group. Our forms are immune to DLP proxies, so we are working within the group to be sure that provisions are made for white lists and that Formularity Safe Forms information isn't blocked just because it can't be inspected.

Now, the corporate IT department already has enough to do. They couldn't care less about health insurance information. Unless they bored at lunch, and the health insurance involves a highly placed executive and his or her substance abuse issues….

<Slide Change to Leak #3>

Now this company uses an ISP that uses bogus signing certificates to reduce bandwidth, through data compression. They also intercept the secure web connection in order to add “super cookies” to the web session, and to add advertising banners and footers. Yes, Verizon, I’m talking about you! So, Leak #3 isn’t just a possibility. At least two major carriers have been caught engaged in modifying secure SSL connections!

<Slide Change to Leak #4>

This poor user! The cards are definitely stacked against him or her. The health insurance company wants fast response times and to localize their web site content based on the user's location. So they use a Content Distribution Network (CDN).

<Slide Change to CDN>

Now, this CDN provider offers three different types of connections between their edge servers and the health insurance company's website. The cheapest connection type does not require an SSL certificate on the health insurance company's web servers. As you would expect, our health insurance company has decide to be economically responsible and to reduce health care costs where they can. They have chosen the "No SSL certs on the servers" option. Their IT department told them that this wasn't prudent, but the CFO doesn't have a clue about SSL certs. He or she however certainly understand reducing operating costs! So, all our user's sensitive health information is now traveling across the country unencrypted. And the user still gets their nice green lock icon!

Again, I'm not singling out a particular provider. At Formularity we are strong supporters of CloudFlare and the work that they are doing to make the internet more secure. But their Flexible SSL solution can be abused and we've seen actual cases of this abuse.

<Slide Change to Leak #5>

Ah, now we are up to the 5th leak in our supposedly secure SSL connection! The health insurance company's hosting provider uses SSL accelerator load balancers where the network connections come into their data center.

<Slide Change to SSL Accelerator>

These high speed boxes terminate all SSL connections, offloading this computationally intensive task from the individual customer servers. Decrypting connections as they enter the data center also helps in load balancing and traffic switching. So this “leak” is intentional, but it definitely violates HIPAA best practices. Our data is still very much in motion but it’s no longer encrypted. It’s going to go through a data center with thousands of other customer’s data flows. Which brings us to the 6th leak in our pipeline…

<Slide Change to Leak #6>

Our poor user's embarrassing health information is being routed through a number of physical and software switches within the data center. All of these switches have spanning ports for diagnostic and troubleshooting purposes.

<Slide Change to Spanning Port>

Easy for anybody with access to the data center to add sniffers to the switches. Because it's all plain text, there's no SSL protection.

<Slide Change to Leak #7>

With modern hosting providers, it’s very likely that our health insurance company doesn’t have dedicated physical servers, and are on virtual machine servers. Their web site may be running on the same physical server as scores of other customers.

<Slide Change to VMs>

Now virtual servers have always been associated with good security. After all, the processors use hardware mechanism to isolate the virtual machines from the hardware and from each other. But, there are two things we know: 1) modern computer hardware is complex, and 2) hackers are clever!

<Slide Change to RowHammer>

Using techniques like Rowhammer and Flip Feung Shi, researchers have been able to discover encryption keys on neighboring virtual machines! Hardware vendors will correct the defects in their designs and operating system vendors will use techniques like advanced address space randomization to defeat these known attack vectors, but the security myth of virtual machines has been broken.

<Slide Change to Hostile Environment>

Let’s face it, the modern hosting provider data center is a hostile environment. Your applications are surrounded by lots of potentially malicious actors!

<Slide Change to Spy>

All these leaks are predicated on known exploits of SSL connections. We aren't even considering the unknown exploits from domestic and foreign espionage!

Here's the greatest threat to your client's private information:

<Slide Change to System Admin>

OK, a large number of us here at LISA2016 are or have been system admins. So, I believe that 99.9% of all system admins are honest, trustworthy, dedicated professionals. I also know that there a few rotten apples out there. Money talks and there are always a few out there that will listen. Insider threat is real.

<Slide Change to Insider Threat>

I get a laugh out of the title of this report. It talks about insider threat in the private sector. After all, we know how well the US Government manages the insider threat problem

<Slide Change to NSA>

Here's a rouge's gallery of recent compromises by US government employees and contractors that have gone through routine polygraph tests and intense background investigation! So, it doesn't matter what measures you take, you are still vulnerable to insider threat.

<Slide Change to Shadow Broker>

Oh, and you would think after Snowden and Manning that the NSA would have cracked down in insider threat. However, it's now been determined that the Shadow Broker leak of NSA infiltration techniques and software was committed by an inside contractor!

<Slide Change to Inquiry Revisited>

Speaking of US Government data leaks, I one of those people that won't have to worry about credit monitoring for the next year, thanks to the Office of Personnel Management! OPM lost the sensitive personal details of millions of government employees and contractors. Interesting Catch-22 type situation. You need a security clearance to work in a number of agencies and activities of the US Government. You have to provide detailed personal information to be granted a security clearance. They then investigate you, digging up more personal information, to be sure you are trustworthy to protect the secrets of the US Government. And then the Government loses all your information. Now who's trustworthy?

But I digress. Remember that question I had earlier about what happens when data transitions from moving to storage? Well, that was actually central to the Office of Personnel Management hack. Sensitive personal information was in plain text as it was being received and put into encrypted databases.

<Slide Change to Column Encryption>

Let's look at encrypted databases for a moment. There are basically two types of encrypted databases. First are those where selected columns of information are encrypted. Like the credit card field or the social security number field. You can query the database on the remaining unencrypted columns. This encryption approach, most often used today still has a number of drawbacks:

First, how is the encryption key generated, stored, and applied to the database? The best approach is a hardware security module (an HSM). The key is generated on site and is stored in a protected hardware device. It can be remotely commanded to provide the keys to the database during startup, without exposing the key over the internet. However, many hosting providers do not have provisions for the attachment of dedicated hardware to their networks.

Use SSL to provide the key to the database server? Really? Weren't you paying attention when I showed you how broken SSL is?

Besides, how the key is transmitted and stored is moot. The key is always available within the running database process. It can be seen by Rowhammer-type virtual machine attacks or by insiders.

<Slide Change to Tablespace Encryption

The other type of encrypted database encrypts the entire disk or tablespace. The database operated on top of the encrypted layer. This makes encryption transparent to the database and does not require any special programming on the part of the developer. Any field can be queried. However this approach has all the same vulnerabilities of the column encryption database. Like SSL, we are encrypting the information, but we really aren't protecting it.

OK, so now that we know the problem, how do we solve it? Well, two changes in our standard protection processes increase our privacy and protection of collected information tremendously. One of these is well-known but isn't normally employed. The other change in our processes seems like a magic trick, but really it's not.

OK, the biggest thing we can do to protect our clients and customer's information is to:

<Slide Change to Encrypted Payload>

Encrypt the information independent of SSL and before we transmit it from the browser. In light of what we've seen regarding the inability of SSL alone to protect information, this seems like an obvious solution. But how many of your company's applications and solutions take advantage of end-to-end encryption outside of SSL?

This "obvious" solution solves all the problems we have seen with SSL and the protection of data in motion

<Slide Change to Solved Problems>

Up until recently, we didn’t have the tools at the browser or client end to do end-to-end encryption well. But now we do:

<Slide Change to Tools>

With AJAX, we moved from our browsers just displaying HTML pages and we've started putting logic at the browser end. AJAX has now been refined into Single Page Applications. We write programs for the client end that are now every bit as sophisticated and capable as the microservices we run on the backend servers. These Single Page Applications can handle strong encryption. [As an aside, it's good to be old – we have now moved back into the days of client-server computing. Been there, done that, even have the t-shirt – although it may have some moth holes in it ;-)]

All the modern browsers have built in strong encryption capabilities through the new Crypto object. If you decide not to use the encryption algorithms in the Crypto object, please at least use the Random Number functions!

Speaking of random numbers, Elliptical Curve Cryptography has greatly simplified the application of public key cryptography. It has also opened up a few new capabilities in cryptography and information protection that are not possible with older "product of primes" public key cryptography. We'll discuss these capabilities in a moment.

Finally, doing crypto is easy. Doing crypto right is hard! We no longer have to make the implementation decisions that decide whether our information protection is strong or laughable. We have libraries like the Stanford Javascript Cryptography Library and

Libsodium. These libraries have been designed to keep us mere mortals save in the world of cryptography. Please, please, please use them!

<Slide Change to Authentication>

Now, at this point, somebody always says to me: “Are you saying that we should stop using SSL and start doing our own encryption?!?! That’s crazy”. And I would agree. It would be foolish to see this as an either-or situation. I’m advocating for you to use BOTH SSL and payload encryption. First, two completely independent protection layers are stronger than either alone. Belt and suspenders! More importantly, we need a trust anchor from which we can build upon. SSL authentication helps us establish this trust. Archimedes is incorrectly quoted as saying “give me a lever long enough, and I can move the whole world”. His writings are more accurately translated as “Give me A PLACE TO STAND and with a lever I will move the whole world.” We need that trusted place to stand. Combining SSL and end-to-end encryption helps give us that trusted standing point.

Now, I promised you a magic trick, a silver bullet:

<Slide Change to Homomorphic Encryption>

Homomorphic encryption! Who here is familiar with homomorphic encryption? Great! For those of you that are not, it's says that we can perform mathematical operations on two pieces of encrypted information without having to decrypt them first. In practical terms, what this means is that we can put already encrypted information into a database and then do selects and queries without ever having to decrypt the information. Think about it – with a homomorphic database at the hosting provider's data center, nobody can steal the encryption keys because they are never in the data center!

<Slide Change to Formal Definition>

We've known about special cases of homomorphic encryption since Ron Rivest co-developed the RSA public key encryption scheme. These are called, logically enough, Semi-Homomorphic Encryption, or SHE. In fact, unpadded RSA encryption is homomorphic relative to multiplication. If you multiply one RSA encrypted number by another RSA encrypted number, the product, when decrypted is, in fact the product of these two numbers. So, we just multiplied two numbers without ever having to know what the numbers were originally! RSA actually had to introduce padding and nonces to prevent this from happening.

Craig Gentry at IBM has now proven that fully homomorphic encryption is possible – just very slow and difficult. As with everything in information technology, fully homomorphic encryption will get faster, smaller, and easier to accomplish. But it turns out we don't have to wait. Somewhat Homomorphic Encryption is sufficient to build robust, production quality homomorphic databases.

<Slide Change to CryptDB>

MIT has developed an experimental proxy that can turn an unmodified MySQL database into a homomorphic database. They have tested it with thousands of WordPress blogs in daily use at MIT. It can handle over 97% of all the WordPress queries presented to the database, and it only adds about 20% overhead. If you're interested in playing with MIT's proxy, called CryptDB, you can download several virtual machines from the Formularity website.

<Slide Change to Always Encrypted>

And homomorphic encryption isn't just an MIT science experiment. Google, Microsoft, and SAP have either already released, or are in the process of releasing a homomorphic database solution. So, encourage your developers to use these products in the future. My privacy and security may depend on it!

<Slide Change to Re-Encryption Proxy>

There's another new magic trick on the horizon that has got our clients particularly excited. It's called a Re-Encrption Proxy. Basically, you can stored homomorphically encrypted information in the cloud. Then, when you want to transfer information to a third-party, you can calculate a special key, based on your third party's public key, which allows only the third party to decrypt the information. We are used to using public keys to encrypt unencrypted information to send it to somebody else securely. A Re-Encryption Proxy lets you do this with already encrypted information. For our clients, this means that you can collect and protect health information using homomorphic encryption, and then you can send a subset of those records to a health insurance provider, all without ever having to decrypt the information in the hosting data center or expose any private or public keys. This is based on the work of Mathew Green at Johns Hopkins University.

<Slide Change to Closing>

So, now you know how current data protection is broken and you also know how to fix it. I hope I've been successful in encouraging you to review your existing data collection and protection policies and consider applying techniques like end-to-end payload encryption and homomorphic cloud data storage.

Thank you for taking the time to attend this talk, and I'd like to thank the Board and organizers of LISA and USENIX for giving me the time to speak to you.

These slides and the general text of the presentation will be available from Formularity's website later today.

Questions?