**EMCS2000: Introduction to Computer Security**

Assignment: Security Description for Purchasing  
Brian Russel Davis, [brian\_davis@brown.edu](mailto:brian_davis@brown.edu)

## TL;DL Version

### How Can Users Be Sure They Are On The Real ATT Site?

AllTheThings.com ( ATT ) uses a site certificate, issued by a Certificate Authority, namely Let’s Encrypt. In order receive this certificate ATT proved ownership of the ATT domain by performing a series of challenges. Then by using the ACME client on the ATT web server, ATT created a key pair and exchanged these keys to create an encrypted channel for communication. This channel was used to issue the site certificate that users will download once they visit the ATT site. ATT also uses HTTP Public Key Pinning to ensure that the right certificate is used on subsequent visits to the ATT site. The ATT certificate is re-issued every 90 days per the Let’s Encrypt policy. Furthermore the ACME automation policy triple checks, and communicates the valid certificate with 3 other CAs.

### How Can Users Be Sure Their Credit Card Information is Safe?

To ensure privacy and security when executing transactions that include personal financial information, ATT uses Transport Layer Security to establish a connection then share a symmetrical encryption key. To protect against TLS related attacks, mainly those that happen during the Diffie–Hellman key exchange, ATT enforces TLS 1.3 for all connections. Because privacy is very important to us here at ATT we are working on a more advanced key exchange, one that uses Elliptical Curve Cryptography and Supersingular isogeny Diffie-Hellman key exchange. This will protect the site against future quantum computers and should improve the overall speed of the website.

### What Else Are We Doing To Mitigate Against Known Threat MGHC?

1. **We limit the IP addresses ATT resolves to.** If we don’t sell outside the US we can block all non-US IP’s. We can also block the IP addresses of known threat actors through automation.
2. **We are proactive in buying up the misspellings and common variations of our domain.** Low hanging fruit indeed, but think of how easy it would be to fool people who are not paying attention to their Browsers address bar, especially for an attacker who has SEO’d the misspellings or is using social media ads or phishing to lure people into a fake site. If the attackers can get a valid CA cert for a misspelled domain, game over.
3. **We ensure that we also register domain names for foreign extensions and alternate spellings with Cyrillic characters.** One of lesser known attacks is creating a site with a domain that switches some of the vowels with Cyrillic characters, that are indistinguishable from Latin characters by most browsers. One researcher did this with apple.com, replacing the “a” with a Cyrillic а, and getting a valid CA cert for apple.com.
4. **We educate our users on what to look for.** A periodical banner at the top of the browser pointing out the green lock and what it means might be ignored by 80% of the traffic, but it is just another attempted to teach the user about the security features in their browser and how we are keeping them safe. Educated consumers are worth more than their weight in gold.
5. **We use cache-busting techniques** to makes sure that the cached version of the home page and other important pages are being retrieved directly from the server to prevent cache poisoning attacks. Writing Cache Busting code is fairly simple, but the benefits are enormous.
6. **We monitor the DNS Server(s) for DoS, DDoS, DNS amplification and Fast flux attacks**. Another popular way of diverting traffic to a fake site is by taking down the DNS Server tampering with the routing. If the DNS server is down, then the main site should shut down and users should be alerted. Fast Flux is a more sophisticated DNS attack, that is harder to detect or stop. The most effective countermeasure against a Fast Flux attack is to shut down the domain ( which may not be possible if it has been severely compromised ). Also, *“A network admin can force endpoints within their network to only be able to use local DNS servers by blocking all egress DNS traffic, and then blackhole requests for malicious domains at the DNS level.”* [[1]](#footnote-0) DoS and DNS attacks are hard to defend against but if we take item 1 seriously we could reduce the threat by at least 50%.

**EMCS2000: Introduction to Computer Security**

Assignment: Security Description for Purchasing  
Brian Russel Davis, [brian\_davis@brown.edu](mailto:brian_davis@brown.edu)

## NASA Comprehensive Version

### Overview

**Security doesn’t begin by throwing tools at problems**, it actually begins with common sense mitigation that is based on a comprehensive threat assessment. We don’t have time in this assignment for a comprehensive threat assessment. We will however discuss mitigation. Cryptography Tools are most effective when we employ mitigation as a *first step* because mitigation allows us to reduce the scope of the problem; I like to call this the *“close the windows, then lock the front approach.”* Furthermore, before we can mitigate we need to know our adversary, MGHC.

### Who Are We Dealing With?

**We need to understand MGHC’s tactics and attack strategies to mitigate against them.** Putting 100% effort into securing everything, all the time, is not realistic. If we know MGHC’s “M.O.” we can secure the most probable target(s) first. For example, are they creating spoof websites that prey on common misspellings of the domain? ( AllTheThing.com, AllTehThings.com, AllTheThings.co, AlTheThings.com ) Are they scanning for open ports in the ATT ( All The Things ) ecosystem, trying to find access to the databases or holes in a web server? Are they phishing users with emails trying to get access to the system, with the plan of using SQL injection to elevate the rights for that compromised user to root? If they are doing all this can we detect the IP addresses they are using to do this? And if so can we block traffic from these IPs? Are they calling into the help desk pretending to be an aggrieved customer, then seducing the customer service agents to reset a password over the phone? Yes, it’s great to use strong cryptography and valid certificates but if we lock the front door of the house with 10 deadbolts but leave all the windows open, what is the point? Knowing their tactics and blocking them sends a message: “We know you are there, and we are ready to fight you.”

### Mitigation Begins with Planning

Now that we have an imaginary threat assessment in place ( we’re fast right?) we can do a few things right off the bat to make sure we reduced the scope of the problem by limiting the attack vectors and neutralizing popular kill chain events.

1. **We can limit the IP addresses ATT resolves to.** If we don’t sell outside the US we can block all non-US IP’s. We can also block the IP addresses of known threat actors through automation. This doesn’t stop MGHC, it just makes the task of hiding themselves a little more annoying. The harder we make them work, the more likely they are to attack someone else. Usually, this work is done by implementing a strong Firewall, but sometimes specialized hardware is required to protect against DDoS attacks as we will discuss in item 8.
2. **We can require certain versions of Browsers ( for each flavor of Browser ) along with a very strict rule on the version of TLS we support.** While this might cause us to lose certain traffic from people with old Browsers it is a business/security trade-off. An additional strategy is to enable [HTTP Public Key Pinning (HPKP)](https://en.wikipedia.org/wiki/HTTP_Public_Key_Pinning). This ensures that we select and pin the version of the CA cert used by the Browser.
3. **We can be proactive by buying up the misspellings and common variations of our domain.** Low hanging fruit indeed, but think of how easy it would be to fool people who are not paying attention to their Browsers address bar, especially for an attacker who has SEO’d the misspellings or is using social media ads or phishing to lure people into a fake site. If the attackers can get a valid CA cert for a misspelled domain, game over.
4. **We can ensure that we also register domain names for foreign extensions and alternate spellings with Cyrillic characters.** One of lesser known attacks is creating a site with a domain that switches some of the vowels with Cyrillic characters, that are indistinguishable from Latin characters by most browsers. One researcher did this with apple.com, replacing the “a” with a Cyrillic а, and getting a valid CA cert for apple.com.
5. **We can educate our users on what to look for.** A periodical banner at the top of the browser pointing out the green lock and what it means might be ignored by 80% of the traffic, but it is just another attempted to teach the user about the security features in their browser and how we are keeping them safe. Educated consumers are worth more than their weight in gold.
6. **We can use cache-busting techniques** to makes sure that the cached version of the home page and other important pages are being retrieved directly from the server to prevent cache poisoning attacks. Writing Cache Busting code is fairly simple, but the benefits are enormous.
7. **We can monitor the DNS Server(s) for DoS, DDoS, DNS amplification and Fast flux attacks**. Another popular way of diverting traffic to a fake site is by taking down the DNS Server tampering with the routing. If the DNS server is down, then the main site should shut down and users should be alerted. Fast Flux is a more sophisticated DNS attack, that is harder to detect or stop. The most effective countermeasure against a Fast Flux attack is to shut down the domain ( which may not be possible if it has been severely compromised ). Also, *“A network admin can force endpoints within their network to only be able to use local DNS servers by blocking all egress DNS traffic, and then blackhole requests for malicious domains at the DNS level.”* [[2]](#footnote-1) DoS and DNS attacks are hard to defend against but if we take item 1 seriously we could reduce the threat by at least 50%.

Depending on industry-specific threats we may be able to do a lot more ( Training for Customer Service, Machine Learning Algorithms scanning the network, Threat Hunting base on APT reports ), but this is a good first step. **Now that we have “shut all the windows”, we can build a better lock for the “front door,” our CA certificate.**

### Implementing Cryptography Tools

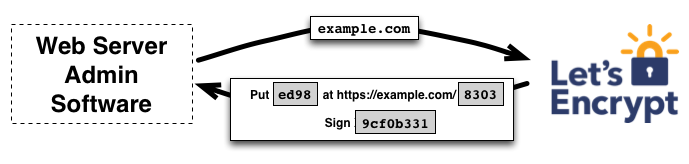
Now that we have limited the scope of the problem, **let’s talk about CA certs and encryption**. The first step is to choose a Certificate Authority. After doing some research I like “[Let’s Encrypt](https://en.wikipedia.org/wiki/Let%27s_Encrypt)”. They are “new-ish” on the scene and have some innovative ways of handling the process of issuing and managing certificates that seem promising. Here’s how they describe themselves:

*“Let's Encrypt is a certificate authority that provides X.509 certificates for Transport Layer Security (TLS) encryption at no charge. The certificate is valid for 90 days, during which renewal can take place at any time. The offer is accompanied by an automated process designed to overcome manual creation, validation, signing, installation, and renewal of certificates for secure websites.” [[3]](#footnote-2)*

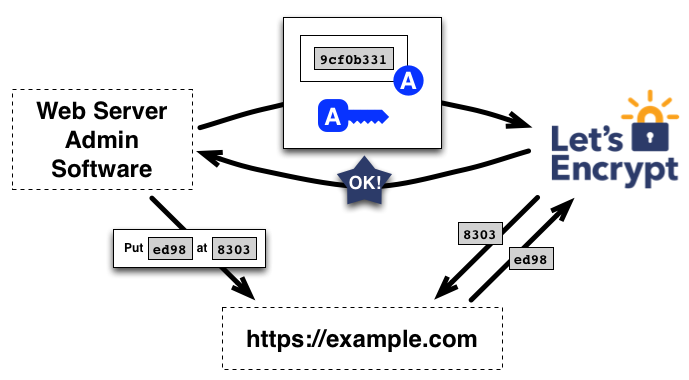
I like this strategy to certificate validation. It forces the admins to renew in a timely fashion and the automation reduces the chance that human error will create a vulnerability. When they issued their first certificate :

*“The root certificate was used to sign two intermediate certificates, which are also cross-signed by the certificate authority IdenTrust. One of the intermediate certificates is used to sign issued certificates, while the other is kept offline as a backup in case of problems with the first intermediate certificate. Because the IdenTrust certificate is preinstalled in major web browsers, Let's Encrypt certificates can normally be validated and are accepted upon installation even before browser vendors include the ISRG root certificate as a trust anchor.” [[4]](#footnote-3)*

To use Let’s Encrypt we basically set up a server to run the ACME client as a bot ( like [CertBot](https://certbot.eff.org/) ) that handles the challenge response needed to validate site ownership and the creation of the sites keys.

*4*

**Pictured above is the first challenge to verify ownership**

4

**This diagram illustrates the key pair creation and exchange**

*“Once the agent has an authorized key pair, requesting, renewing, and revoking certificates is simple—just send certificate management messages and sign them with the authorized key pair.” [[5]](#footnote-4)*

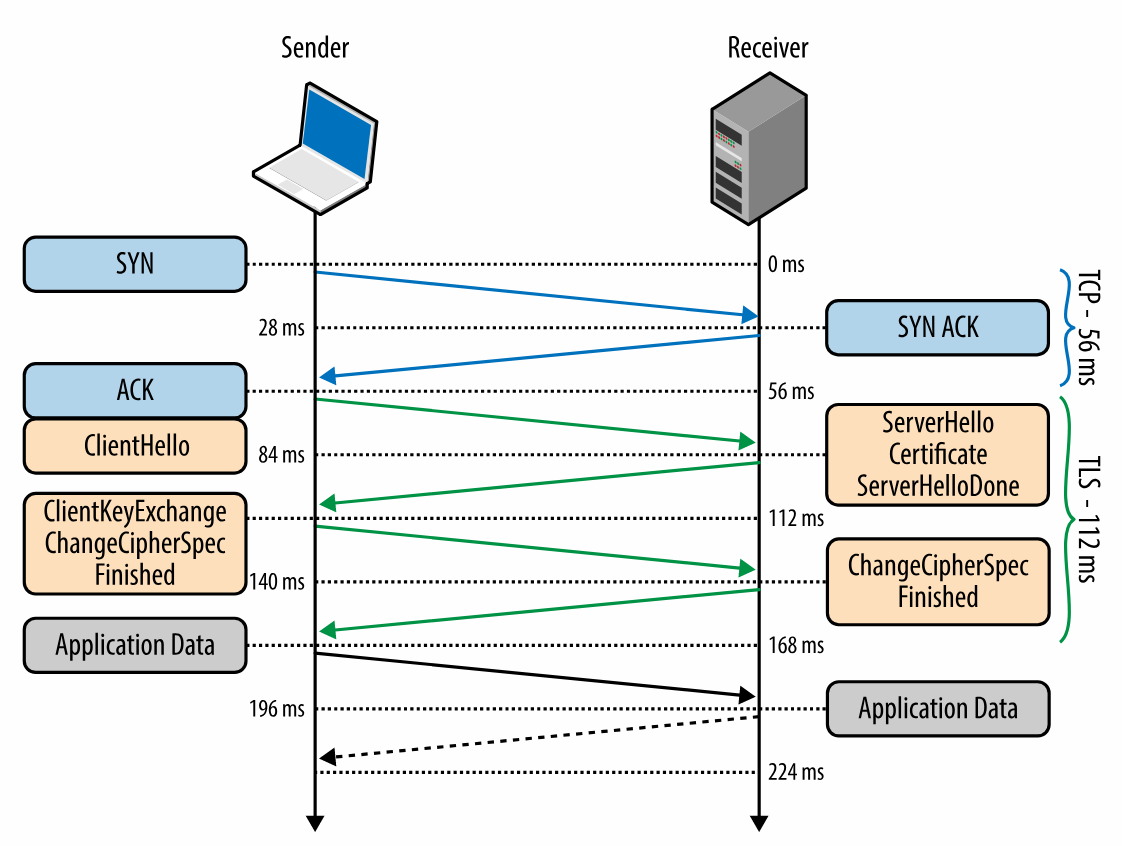
One added feature in Let’s Encrypt framework that I am very excited about is their promise to implement [Elliptic-curve cryptography](https://en.wikipedia.org/wiki/Elliptic-curve_cryptography) by using an [Elliptic Curve Digital Signature Algorithm](https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm). The core benefits being that ECDSA is just as hard to break ( if not harder ) than RSA but requires much smaller keys. The net benefit is faster websites since the cryptologic communications take less time.

To recap, once ownership is verified with multiple challenges, the CertBot communicates with the Let’s Encrypt Server to create RSA key pairs. These set of key pairs are used to communicate in an encrypted fashion onward. The CA certificate is sent to the ATT server through this channel. Installation is automated by the CertBot running of the ATT server.

### Protecting Credit Card Information With

**Protecting credit cards is very important**, and needs to be secured from the prying eyes of MGHC**.** The easiest way to do this is to ensure that the communication between the server and the user is private and encrypted. We can achieve this by creating encrypted sessions using TLS. Previously known as SSL ( Secure Socket Layer ), TLS ( Transport Layer Security ) 1.3 is the latest and greatest.  *“The TLS protocol comprises two layers: the TLS record and the TLS handshake protocols.”* MGHC might be watching the traffic on the network, but we can make everything they see gibberish if we execute a proper TLS handshake, exchange symmetrical keys and encrypt the communication between the server and the user with these keys.

*“Before the client and the server can begin exchanging application data over TLS, the encrypted tunnel must be negotiated: the client and the server must agree on the version of the TLS protocol, choose the ciphersuite, and verify certificates if necessary.” 5*

[[6]](#footnote-5)

**Understanding how TLS works is great, but understanding how attackers bypass it is better.** There are too many TLS attacks to discuss here but they include: Renegotiation attack, Downgrade attacks: FREAK attack and Logjam attack, Cross-protocol attacks: DROWN, BEAST attack, CRIME and BREACH attacks, Timing attacks on padding, POODLE attack, RC4 attacks, Truncation attack, Unholy PAC attack, Sweet32 attack and Implementation errors including the Heartbleed bug, BERserk attack, and Cloudflare bug. Many of these issues have been fixed or mitigated against with the exception of the Protocol Downgrade. 20% of all sites are still vulnerable to this attack. Much of the risk in using TLS is executing the initial handshake.

While most sites can be protected by simply enforcing the highest version of TLS that they can afford. There is exciting and interesting research going on that will make TLS handshakes secure even against quantum computing attacks, namely Supersingular isogeny key exchange. As the research continues a library published by Microsoft is available now: <https://github.com/Microsoft/PQCrypto-SIDH>; along with a Java implementation by a independent engineer: <https://github.com/Art3misOne/sidh>

***However, I see an opportunity here.*** SIDH may work better if implemented as a State Machine. I have started a project that involves converting one of the above libraries to Python and implementing the key exchange using [AWS Step Functions](https://aws.amazon.com/step-functions/) ( a Serverless State Machine ) that calls out to a mesh of [Lambda Functions](https://en.wikipedia.org/wiki/Anonymous_function) ( Serverless Stateless Anonymous Functions ) all inside of an encrypted firewalled virtual private cloud ( VPC ). This would make SIDH usable and highly interoperable, requiring no installation, just the permission to access the Service Mesh via some sort of SAML authentication. <https://github.com/nurdymuny/SIDH>

### More Common Sense Mitigation to Protect Credit Cards

Once TLS is implemented and fortified there are a few more common sense steps we can take to protect the user’s credit card:

1. **Obfuscate the credit card number** on the page with \*\*\*\* after it’s entered.
2. **Make sure the credit card number doesn’t get saved in the cache** in case the user navigates away from the payment page.
3. **Create a timer on the payment page** so attacks that require more time will fail.
4. **Require login and two factor authentication before a user can make a payment**, and lock the account of a user with too many failed password attempts.
5. **Disable copy and paste in sensitive fields** to prevent malicious CSS from stealing card numbers.
6. **Obfuscate the id of the fields in CSS and HTML tags.** This destroys the ability for a user to use autofill but it also reduces the chances of a malicious plug-in seeing these fields and scraping them.
7. **Require the user to save payment details as opposed to entering them over and over again for consecutive purchases.** The less we force users to expose the credit card the better.
8. **Using a payment processor like PayPal, GooglePay, AmazonPay, VisaPay, MastercardPay or Bolt.** These third-party tools cost money but they reduce risk by not requiring the user to enter their credit card number and shift the liability for fraud to the payment processor.

1. Danford; Salusky (2007). "The Honeynet Project: How Fast-Flux Service Networks Work". Retrieved 2010-08-23 [↑](#footnote-ref-0)
2. Danford; Salusky (2007). "The Honeynet Project: How Fast-Flux Service Networks Work". Retrieved 2010-08-23 [↑](#footnote-ref-1)
3. Eckersley, Peter (November 18, 2014). "Launching in 2015: A Certificate Authority to Encrypt the Entire Web". Electronic Frontier Foundation. Retrieved February 27, 2015. [↑](#footnote-ref-2)
4. Aas, Josh (June 4, 2015). "Let's Encrypt Root and Intermediate Certificates". Let's Encrypt. [↑](#footnote-ref-3)
5. “How It Works.” Let's Encrypt, Let's Encrypt, 14 Dec. 2015, letsencrypt.org/how-it-works/. [↑](#footnote-ref-4)
6. Grigorik, Ilya. “Networking 101: Transport Layer Security (TLS) - High Performance Browser Networking (O'Reilly).” High Performance Browser Networking, O'Reilly, 10 Dec. 2017, hpbn.co/transport-layer-security-tls/. [↑](#footnote-ref-5)