NIC

Future Edition

Andy Malone

Available Now

Shadows Rising is the electrifying sequel to Andy Malone's award winning, The Seventh Day. Set in both 1710 Scotland and the modern world, think Highlander meets the Davinci Code.

From historic Scottish villages, to the streets of New York and the ancient catacombs of Vatican City. Shadows Rising is a race against time thriller, packed with rich historical references entwined into a tense story that will leave you breathless and on the edge of your seat …

Some secrets are better left undiscovered …

Book Signing today @ Glasspaper Booth

DARK NEBULA PUBLISHING


THE NIGHT IS DARK & FULL OF HACKERS
SECURITY TIPS & TRICKS FROM BEYOND THE WALL

Andy Malone

(Scotland, UK)

- Microsoft MVP (Enterprise Security 11 Years)
- Microsoft Certified Trainer (21 years)
- Microsoft Internal Staff Instructor
- Founder: Cybercrime Security Forum!
- Microsoft Ignite Speaker
- Author off the Sc-Fi Thrillers
- “The Seventh Day” & “Shadows Rising”
- Book Signing @NIC

Session Objectives

- Latest vulnerabilities
- Venerability Categories
- Exploring the technology
  - What it is?
  - How it works?
  - How it could be exploited
  - Countermeasures
- Review

The World of Cybersecurity in 2018

- 2017 saw some of the worlds biggest hacks
  - The rise of the Shadow Brokers
  - WannaCry Ransomeware
  - Petya/NotPetya/Nyetya/Goldeneye: Attack actually masked a targeted cyberattack against Ukraine
  - Bad Rabbit
  - WikiLeaks CIA Vault 7
  - Cloudbleed
  - 198 million US voters details leaked
  - Marcron Campaign Hack
  - And not forgetting Equifax, Yahoo …

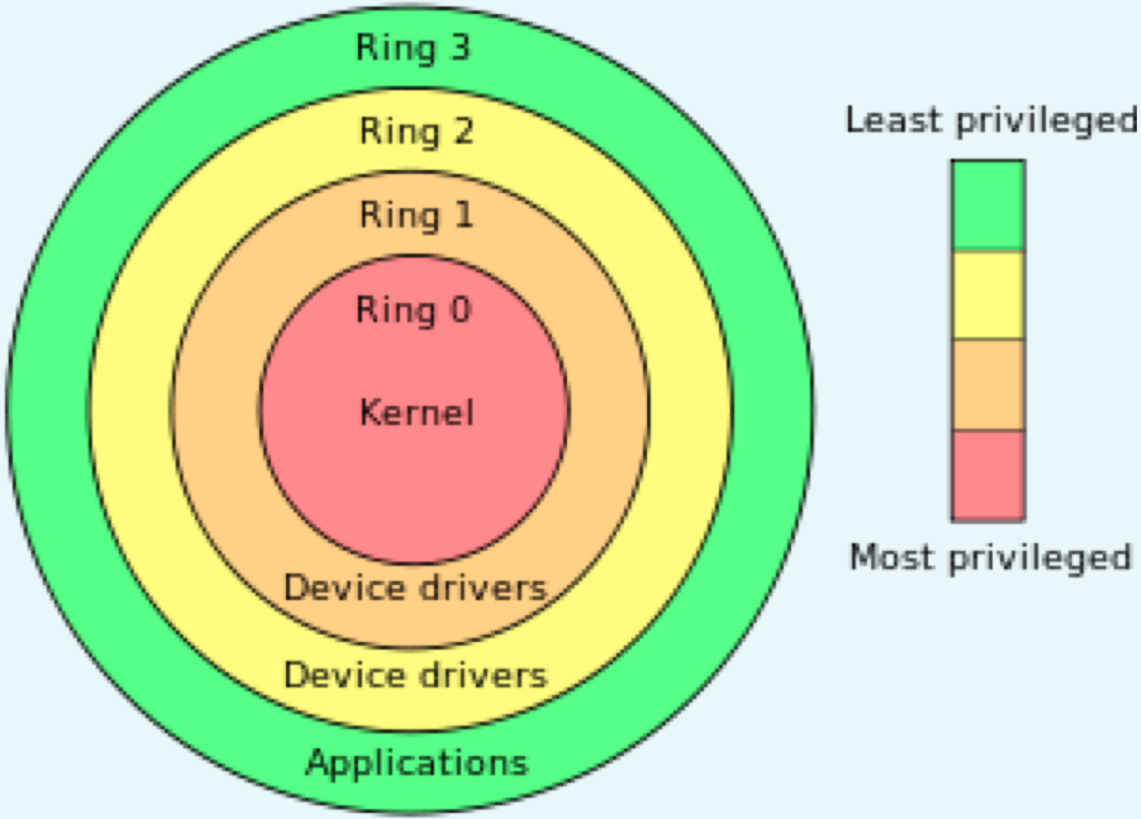
Demo

Norse

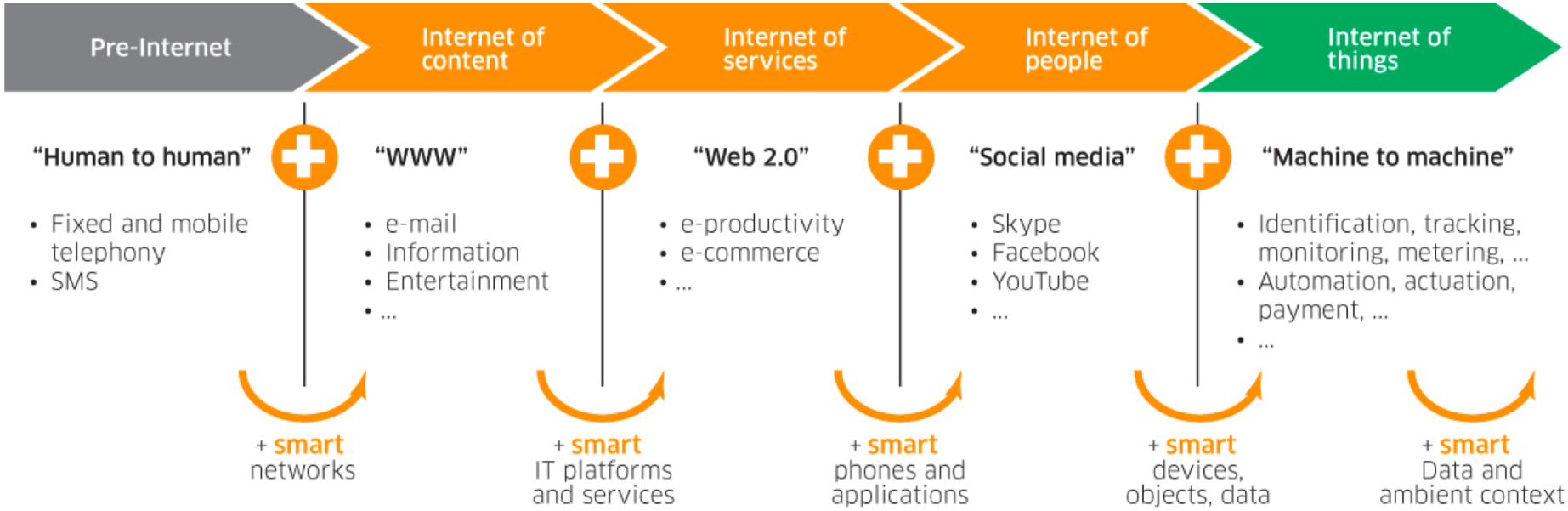
First, let’s start with a Story!

The Great Wall of the Internet

Rings of Protection



The Great Wall that is the Internet



Sounds Great!

So what’s the problem …

Vulnerabilities are everywhere …

- Flows is Operating Systems
- Poorly Configured systems
- Known bugs often ignored.
- Key ESCROW widely practiced
- Mis-configured apps
- Poorly trained staff

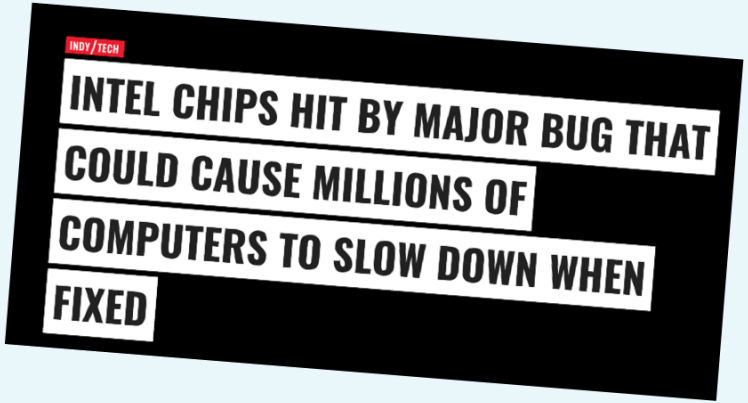
- Hardware issues
- Quickly rushed to market without considering security
- Customers not upgrading
- Flaws bugs often purposefully overlooked due to replacement costs

- Flaws in Communications
- Many companies still using old technology
- Poorly designed
- Lack of security knowledge, poorly configured

- Flaws in Infrastructure
- Often not designed with Security in mind.
- Often still using old technology
- Lack of government funding

What’s the Fuss?

- Infrastructure (Cabling, Communication Providers etc.)
- Communications Infrastructure, Protocols
- Physical Hardware, CPU / Chip / Device flaws
- Operating Systems
- Application Software
- Internet is based solely on Trust!
- Other Factors, people etc. …



Communication Vulnerabilities …

Communication Vulnerabilities …

- Often Built on Old Technology
- Flaws in Communication Protocols, e.g. SSL, TLS
- IP Address Spoofing
- ARP Spoofing
- DNS hijacking
- Certificate Trust Issues
- External Hacks

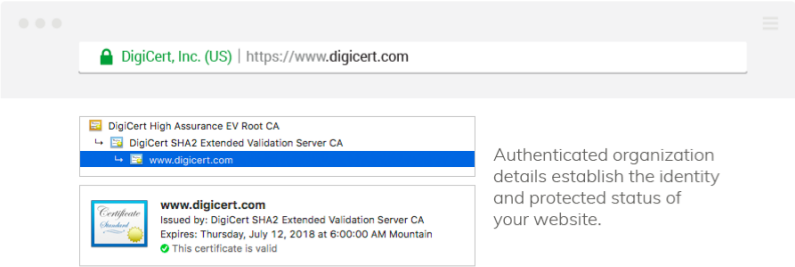
Beneath the Technology
- SSL Secure Sockets Layer -

It’s all in the Protocol: SSL

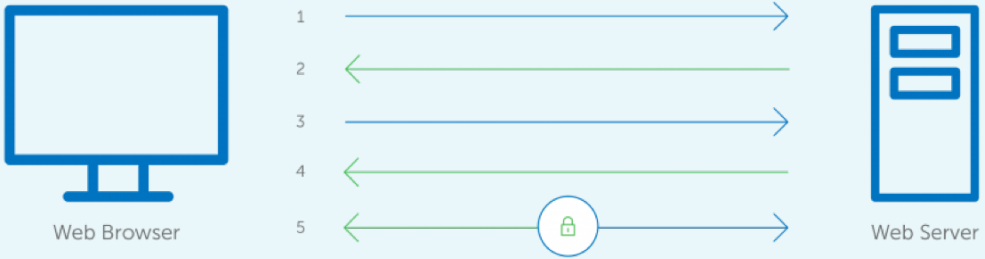
- A standard security technology for establishing an encrypted link between a web server and a browser
- Ensures that all data passed between the web server and browsers remain private and integral
- SSL is an industry standard and is used by millions of websites in the protection of their online transactions with their customers

SSL becomes TLS

- The SSL protocol has traditionally been used to encrypt and secure transmitted data
- Each time a new and more secure version was released, only the version number was altered to reflect the change (e.g., SSLv2.0)
- However, when the time came to update from SSLv3.0, instead of calling the new version SSLv4.0, it was renamed TLSv1.0
- TLSv1.2. is the latest version



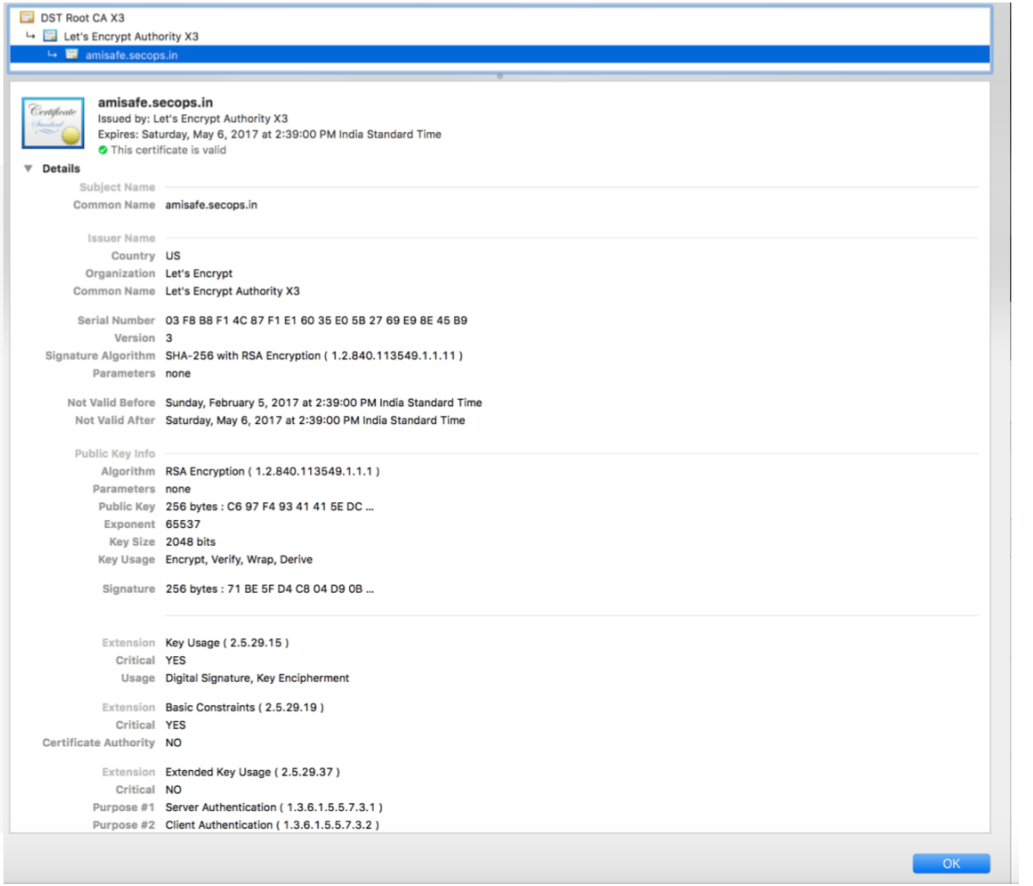
How does SSL Work?



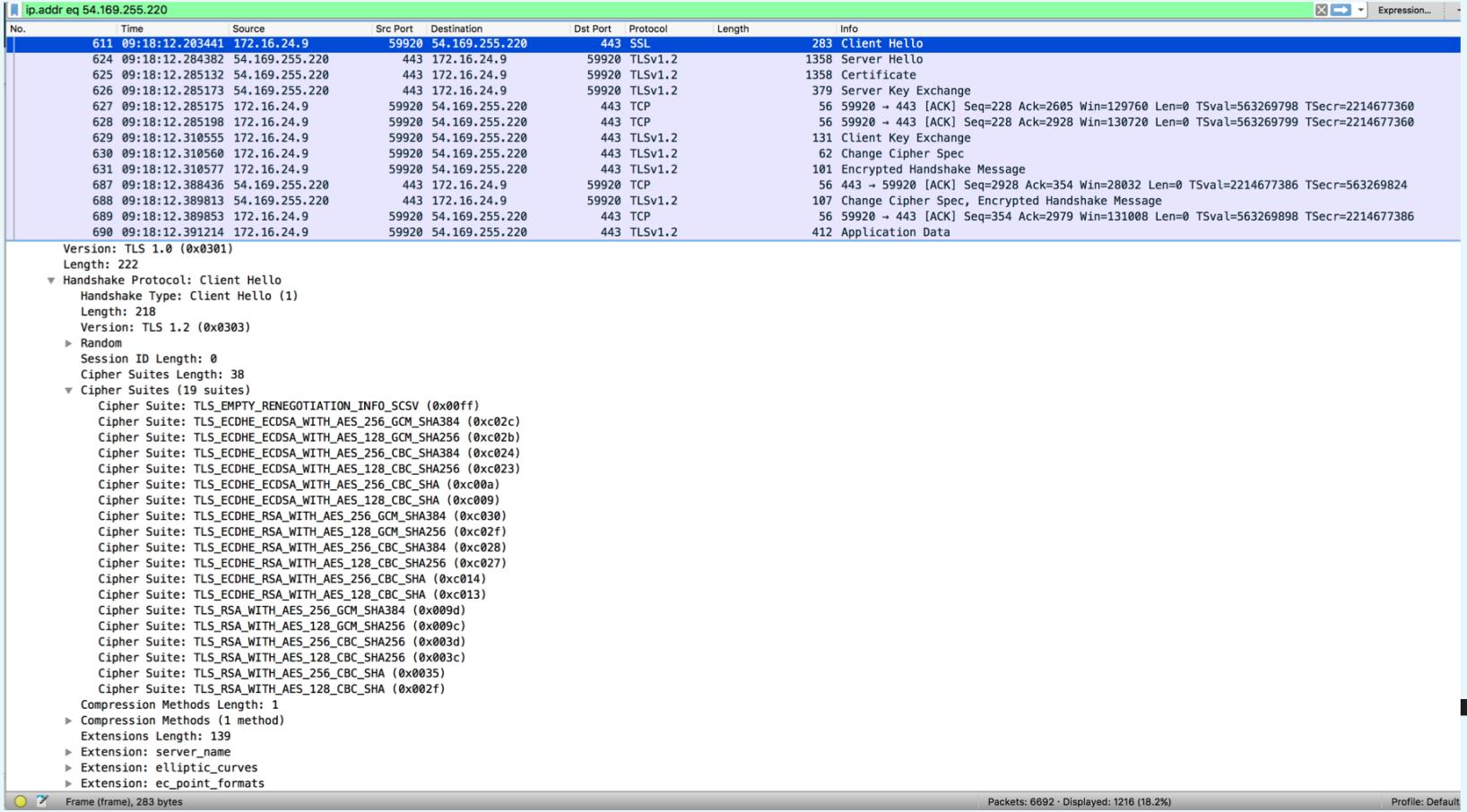
1. **Browser** connects to a web server (website) secured with SSL (https). Browser requests that the server identify itself.
2. **Server** sends a copy of its SSL Certificate, including the server's public key.
3. **Browser** checks the certificate root against a list of trusted CAs and that the certificate is unexpired, unrevoked, and that its common name is valid for the website that it is connecting to. If the browser trusts the certificate, it creates, encrypts, and sends back a symmetric session key using the server's public key.
4. **Server** decrypts the symmetric session key using its private key and sends back an acknowledgement encrypted with the session key to start the encrypted session.
5. **Server** and **Browser** now encrypt all transmitted data with the session key.

Under the Hood:

```
Certificate
  -Version Number
  -Serial Number
  -Signature Algorithm ID
  -Issuer Name
  -Validity period
    --Not Before
    --Not After
  -Subject name
  -Subject Public Key Info
    --Public Key Algorithm
    --Subject Public Key
  -Issuer Unique Identifier (optional)
  -Subject Unique Identifier (optional)
  -Extensions (optional)
Certificate Signature Algorithm
Certificate Signature
```



Under the Hood: #1 Client SSL Hello (RSA Key)

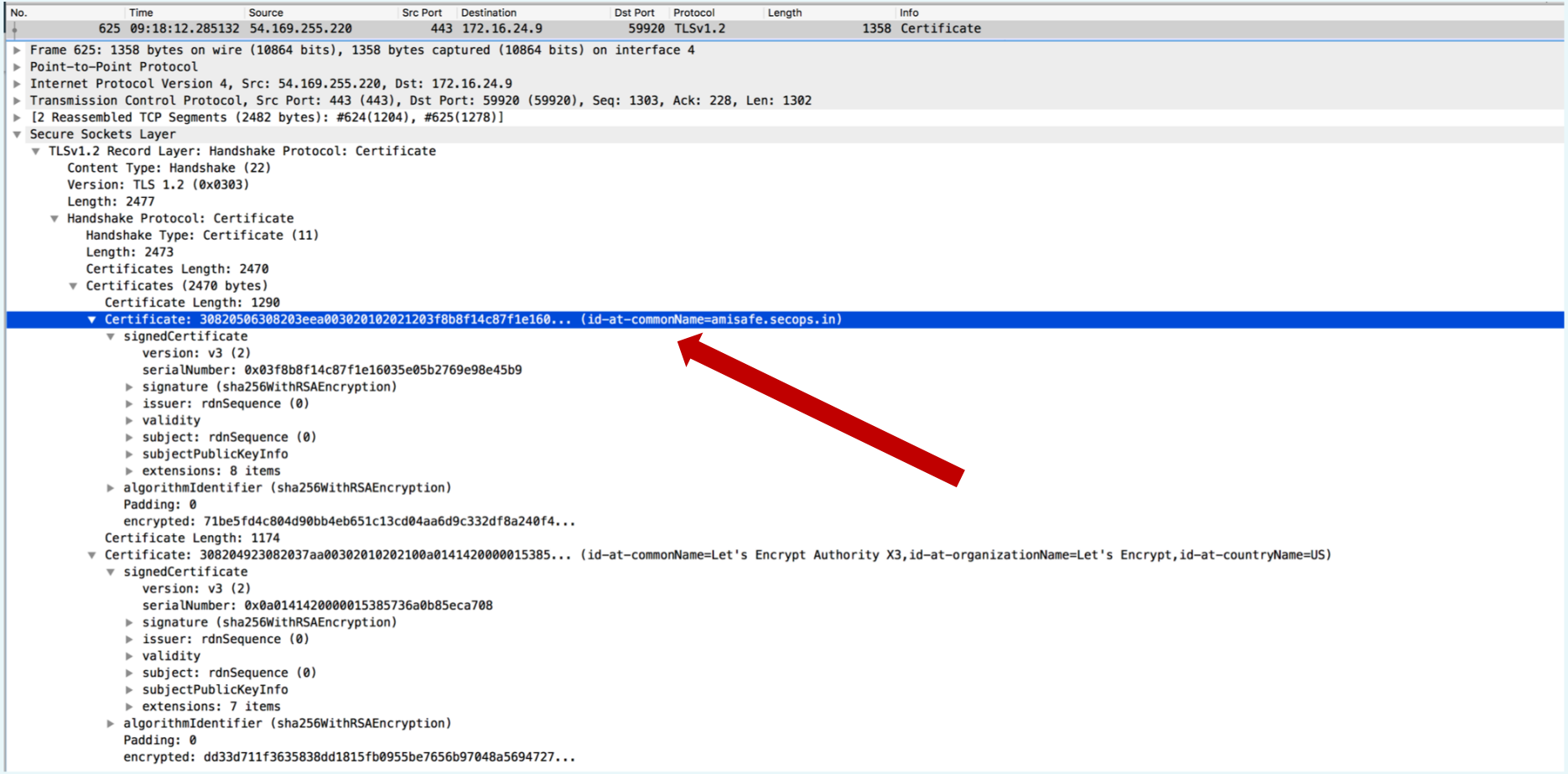


Under the Hood: #2 Server Hello (RSA Key)

No.	Time	Source	Src Port	Destination	Dst Port	Protocol	Length	Info
611	09:18:12.203441	172.16.24.9	59920	54.169.255.220	443	SSL	283	Client Hello
624	09:18:12.284382	54.169.255.220	443	172.16.24.9	59920	TLSv1.2	1358	Server Hello
625	09:18:12.285132	54.169.255.220	443	172.16.24.9	59920	TLSv1.2	1358	Certificate
626	09:18:12.285173	54.169.255.220	443	172.16.24.9	59920	TLSv1.2	379	Server Key Exchange
627	09:18:12.285175	172.16.24.9	59920	54.169.255.220	443	TCP	56	59920 → 443 [ACK] Seq=228 Ack=2605 Win=129760 Len=0 TSval=563269798 TSecr=2214677360
628	09:18:12.285198	172.16.24.9	59920	54.169.255.220	443	TCP	56	59920 → 443 [ACK] Seq=228 Ack=2928 Win=130720 Len=0 TSval=563269799 TSecr=2214677360
629	09:18:12.310555	172.16.24.9	59920	54.169.255.220	443	TLSv1.2	131	Client Key Exchange
630	09:18:12.310560	172.16.24.9	59920	54.169.255.220	443	TLSv1.2	62	Change Cipher Spec
631	09:18:12.310577	172.16.24.9	59920	54.169.255.220	443	TLSv1.2	101	Encrypted Handshake Message
687	09:18:12.388436	54.169.255.220	443	172.16.24.9	59920	TCP	56	443 → 59920 [ACK] Seq=2928 Ack=354 Win=28032 Len=0 TSval=2214677386 TSecr=563269824
688	09:18:12.389813	54.169.255.220	443	172.16.24.9	59920	TLSv1.2	107	Change Cipher Spec, Encrypted Handshake Message
689	09:18:12.389853	172.16.24.9	59920	54.169.255.220	443	TCP	56	59920 → 443 [ACK] Seq=354 Ack=2979 Win=131008 Len=0 TSval=563269898 TSecr=2214677386
690	09:18:12.391214	172.16.24.9	59920	54.169.255.220	443	TLSv1.2	412	Application Data

```
▶ Frame 624: 1358 bytes on wire (10864 bits), 1358 bytes captured (10864 bits) on interface 4
▶ Point-to-Point Protocol
▶ Internet Protocol Version 4, Src: 54.169.255.220, Dst: 172.16.24.9
▶ Transmission Control Protocol, Src Port: 443 (443), Dst Port: 59920 (59920), Seq: 1, Ack: 228, Len: 1302
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 93
    ▼ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 89
        Version: TLS 1.2 (0x0303)
      ▶ Random
        Session ID Length: 32
        Session ID: 973b39d4d5d6d5cf6bdfa7449fc53c17dc760bf6ac9d22ca...
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Compression Method: null (0)
        Extensions Length: 17
      ▶ Extension: server_name
      ▶ Extension: renegotiation_info
      ▶ Extension: ec_point_formats
```

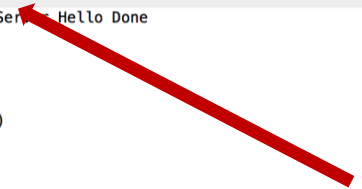
Under the Hood: #3 Auth & Pre Master Secret



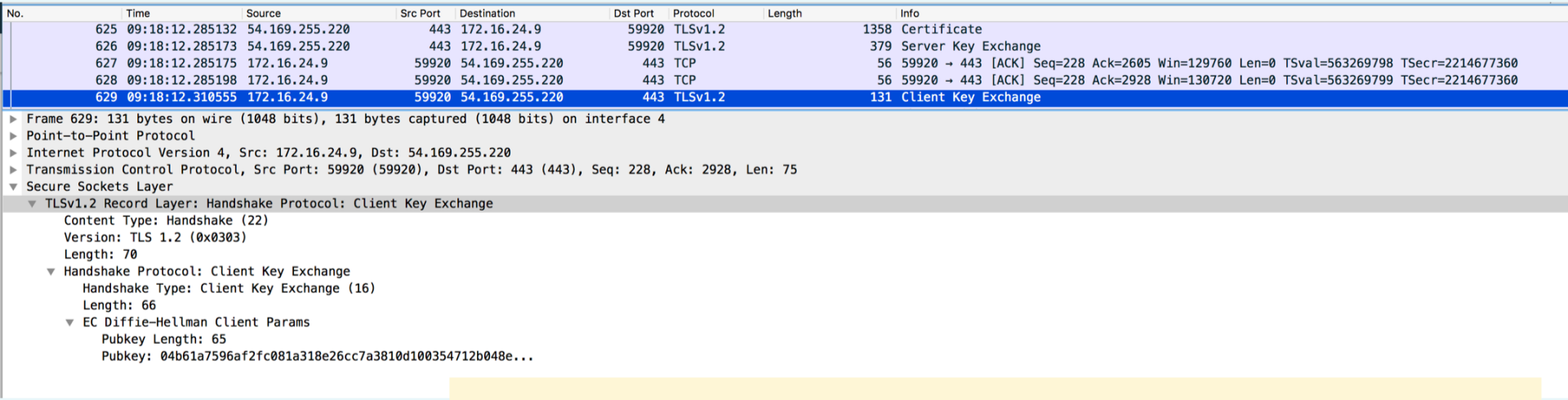
Under the Hood: #4 Encryption with Session Key

No.	Time	Source	Src Port	Destination	Dst Port	Protocol	Length	Info
625	09:18:12.285132	54.169.255.220	443	172.16.24.9	59920	TLSv1.2	1358	Certificate
626	09:18:12.285173	54.169.255.220	443	172.16.24.9	59920	TLSv1.2	379	Server Key Exchange
627	09:18:12.285175	172.16.24.9	59920	54.169.255.220	443	TCP	56	59920 → 443 [ACK] Seq=228 Ack=2605 Win=129760 Len=0 TSval=563269798 TSecr=2214677360
628	09:18:12.285198	172.16.24.9	59920	54.169.255.220	443	TCP	56	59920 → 443 [ACK] Seq=228 Ack=2928 Win=130720 Len=0 TSval=563269799 TSecr=2214677360
629	09:18:12.310555	172.16.24.9	59920	54.169.255.220	443	TLSv1.2	131	Client Key Exchange

```
▶ Frame 626: 379 bytes on wire (3032 bits), 379 bytes captured (3032 bits) on interface 4
▶ Point-to-Point Protocol
▶ Internet Protocol Version 4, Src: 54.169.255.220, Dst: 172.16.24.9
▶ Transmission Control Protocol, Src Port: 443 (443), Dst Port: 59920 (59920), Seq: 2605, Ack: 228, Len: 323
▶ [2 Reassembled TCP Segments (338 bytes): #625(24), #626(314)]
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 333
    ▼ Handshake Protocol: Server Key Exchange
        Handshake Type: Server Key Exchange (12)
        Length: 329
      ▼ EC Diffie-Hellman Server Params
          Curve Type: named_curve (0x03)
          Named Curve: secp256r1 (0x0017)
          Pubkey Length: 65
          Pubkey: 0449134b0de5a88444be6509b811ebbb29093fc34261d070...
        ▶ Signature Hash Algorithm: 0x0401
          Signature Length: 256
          Signature: 1266c33ddb57186d549947e197b6f4764e51480cd0a4a944...
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 4
    ▼ Handshake Protocol: Server Hello Done
        Handshake Type: Server Hello Done (14)
        Length: 0
```



Under the Hood: #4 Encryption with Session Key

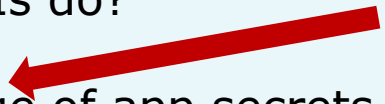



```
openssl s_client -connect amisafe.secops.in:443 -ssl3

openssl s_client -connect amisafe.secops.in:443 -tls1

openssl s_client -connect amisafe.secops.in:443 -tls1.1

openssl s_client -connect amisafe.secops.in:443 -tls1.2
```

Tools like **OpenSSL** can be used check the SSL/TLS negotiations.

ASP.NET Extensible Security APIs

- Core cryptography extensibility
- Key management extensibility
- Miscellaneous APIs
- What do these APIs do?
  - Enforce SSL
  - Allow safe storage of app secrets during development
  - Azure Key Vault configuration provider
  - Provide Anti-Request Forgery
  - Preventing Open Redirect Attacks
  - Preventing Cross-Site Scripting
  - Enabling Cross-Origin Requests
  - Safe Storage of App Secrets



https://docs.microsoft.com/en-us/aspnet/core/security/data-protection/extensibility/

Enforcing SSL in an ASP.NET Core app

- The RequireHttpsAttribute is used to require SSL. You can decorate controllers or methods with this attribute or you can apply it globally as shown below:

C# Copy

```
// Requires using Microsoft.AspNetCore.Mvc;
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<MvcOptions>(options =>
    {
        options.Filters.Add(new RequireHttpsAttribute());
    });
```

Add the following code to `ConfigureServices` in `Startup`:

The highlighted code above requires all requests use `HTTPS`, therefore HTTP requests are ignored. The following highlighted code redirects all HTTP requests to HTTPS:

C# Copy

```
// Requires using Microsoft.AspNetCore.Rewrite;
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFa
{
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();

    var options = new RewriteOptions()
        .AddRedirectToHttps();

    app.UseRewriter(options);
```

Enforcing SSL Via IIS

- IIS Management Console HTTP 308 to https is the only method to enforce **(IIS web site option. Require SSL)** using permanent redirect, without a header and request type change.
- You can use **IIS admin console for HTTP 308 Permanent Redirect method** as well as your redirect-only (empty) web site with default HTTP binding redirects to you HTTPS-only ASP.NET Core web site with HTTPS binding, using permanent HTTP 308,

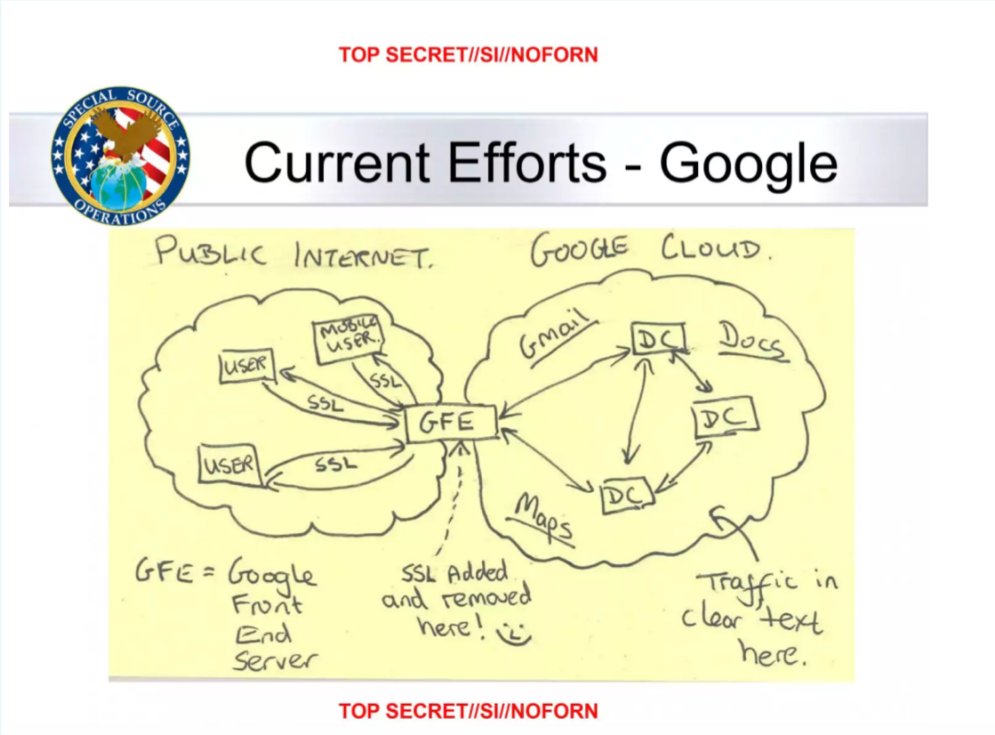
```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<system.webServer>
  <handlers>
    <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModule"
resourceType="Unspecified" />
  </handlers>
  <aspNetCore processPath="dotnet" arguments=".\deafaultdotnetcore20.dll"
stdoutLogEnabled="false" stdoutLogFile=".\logs\stdout" />
      <httpRedirect enabled="true" destination="https://www.experimentalcommunity.org"
httpResponseStatus="PermRedirect" />
        <defaultDocument enabled="false" />
  </system.webServer>
</configuration>
```

You can try to GET **http://www.experimentalcommunity.org** and redirected immediately with HTTP 308

SSL Vulnerabilities …

Known Government SSL Attacks

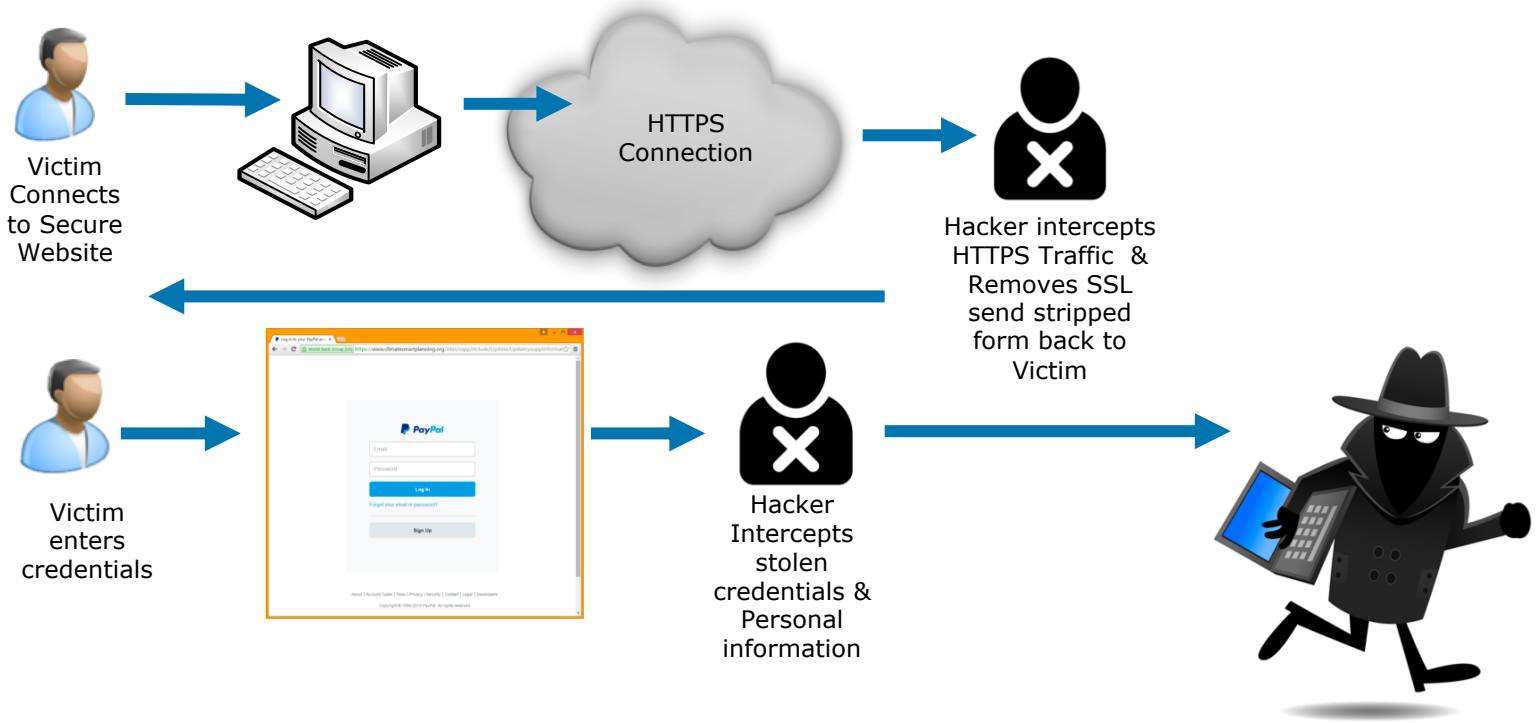
- NSA Experiment for massive SSL/TLS Decryption
- Canadian Document from CES on TLS Trends
- Details on how NSA uses the SCARLETFEVER program to attack Secure Sockets Layer (SSL) / Transport Layer Security
- (TLS)Analysis from SSL/TLS Connections through GCHQ in the flying pig database



Current known SSL Attacks

- BEAST (Browser Exploit Against SSL/TLS) ...
- BREACH (Browser Reconnaissance & Exfiltration via Adaptive Compression of Hypertext) ...
- CRIME (Compression Ratio Info-leak Made Easy) ...
- FREAK (Factoring Attack on RSA-EXPORT Keys) ...
- (Open SSL) Heartbleed Bug. ...
- SSL Strip
- Insecure TLS Renegotiation. ...
- Logjam Attack. ...
- RC4 Cipher Enabled.

Example: SSL Strip Man in the Middle SSL Attack



Demo

SSL Strip

Authentication Vulnerabilities …

Authentication Protocols

Communication or cryptographic protocols designed to transfer authentication data between two entities. Typically Client & Server.

- Authentication methods typically authenticate either using a token, secret key or digital certificate. There are many examples including:
  - PAP - Password Authentication Protocol (Clear Text)
  - CHAP - Challenge-handshake authentication protocol
  - EAP - Extensible Authentication Protocol
  - SAML
  - OAuth 2.0
  - OpenID Connect
  - Kerberos (protocol)
  - Plus many more …

Underneath the Technology
OAuth 2.0

Oauth 2.0: What is it?

- **OAuth 2** allows applications to obtain limited access to user accounts on a HTTPS service, such as Facebook, Office 365,Google, Netflix and others.
- **OAuth 2** An open-web specification for organizations to access protected resources on each other's web sites
- **OAuth 2** provides authorisation flows for web and desktop applications, and mobile devices
- OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices.
- This specification is being developed within the IETF OAuth WG

OAuth 2.0: Pro's & Cons

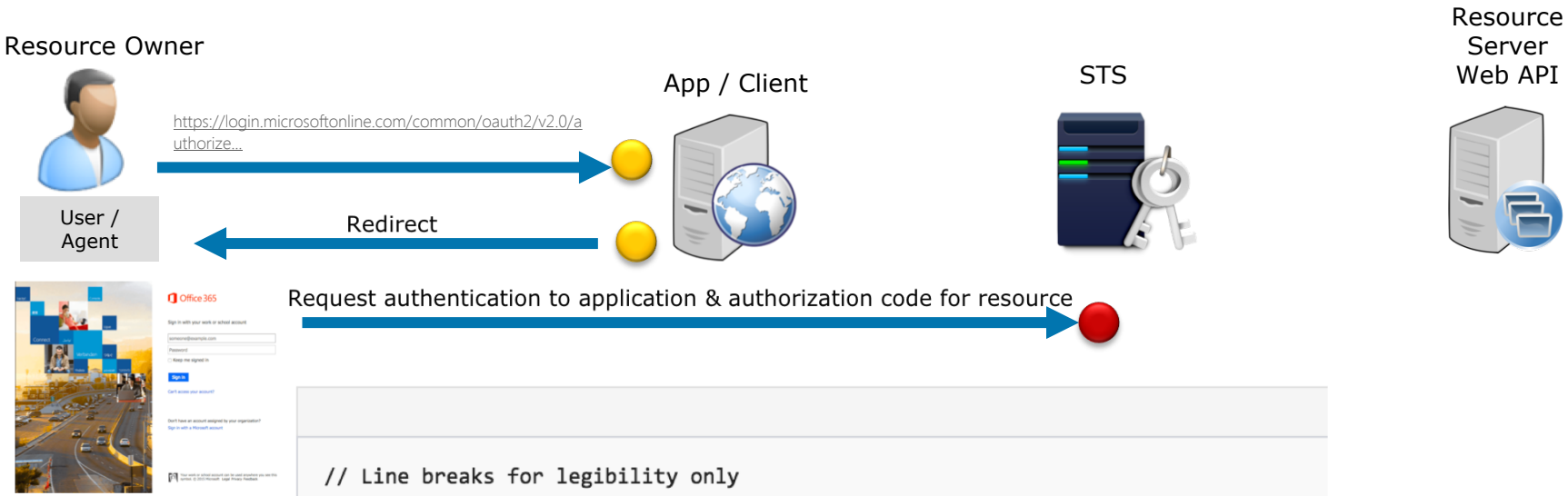
- The Good
  - Easier to implement than its predecessor OAuth 1.0 (Did not scale well)
  - Uses short lived tokens
  - Uses encapsulated tokens
- The Bad
  - No Signature. Relies solely on SSL/TLS (Bearer Tokens)
  - No built in Security
  - Can be dangerous if improperly implemented
  - Heavy burden on the Client
- The Ugly
  - Oauth 2.0 is not a protocol, it's a framework RFC 6749
  - Not Interoperable
  - Lots of misunderstanding

OAuth 2.0: So what are the Key Components

- **The Third-Party Application: "Client"**
- The client is the application that is attempting to get access to the user's account. It needs to get permission from the user before it can do so.
- **The API: "Resource Server"**
- The resource server is the API server used to access the user's information.
- **The Authorization Server**
- This is the server that presents the interface where the user approves or denies the request. In smaller implementations, this may be the same server as the API server, but larger scale deployments will often build this as a separate component.
- **The User: "Resource Owner"**
- The resource owner is the person who is giving access to some portion of their account.

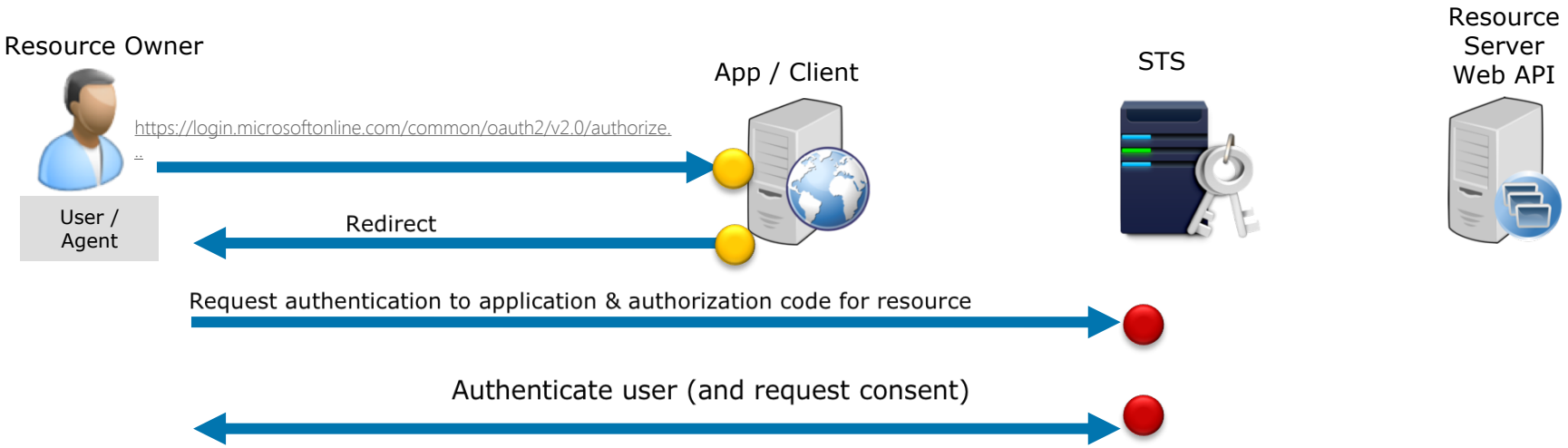
https://aaronparecki.com/oauth-2-simplified/

OAuth 2.0 Authorization Process

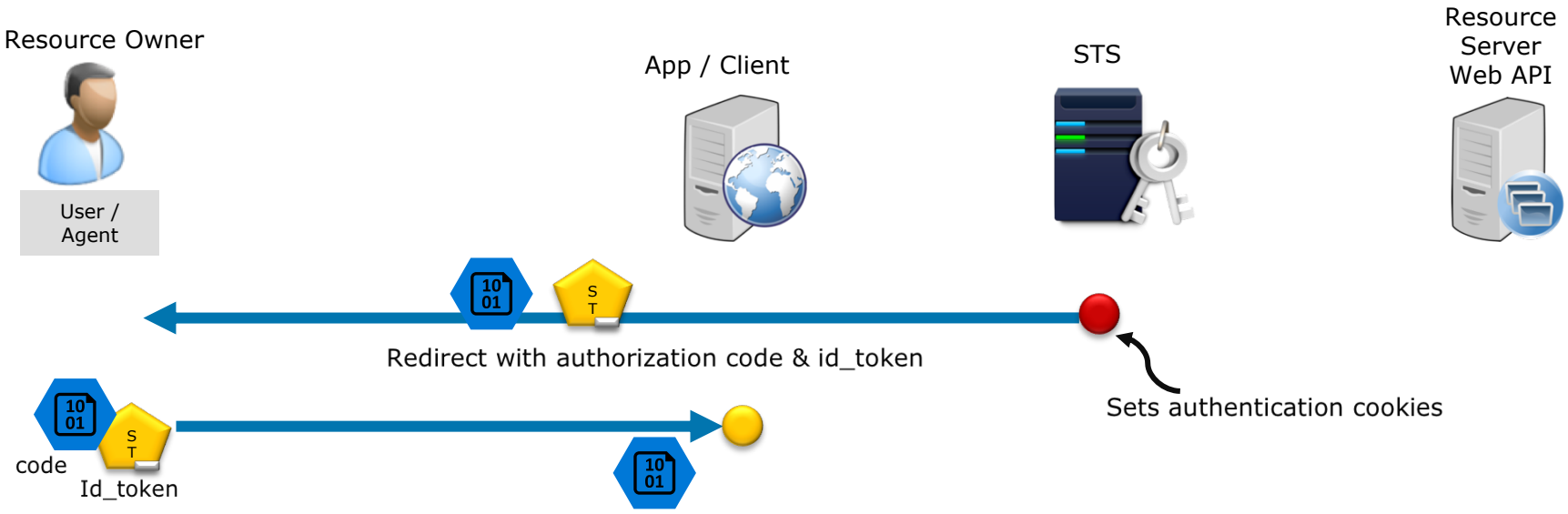



```
// Line breaks for legibility only

https://login.microsoftonline.com/{tenant}/oauth2/v2.0/authorize?
client_id=6731de76-14a6-49ae-97bc-6eba6914391e
&response_type=code
&redirect_uri=http%3A%2F%2Flocalhost%2Fmyapp%2F
&response_mode=query
&scope=openid%20offline_access%20https%3A%2F%2Fgraph.microsoft.com%2Fmail.read
&state=12345
```

OAuth 2.0 Authorization Process



OAuth 2.0 Authorization Process



OAuth 2.0 Authorization Process

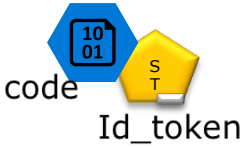
Resource Owner

User / Agent

App / Client

STS

Resource Server Web API



Copy

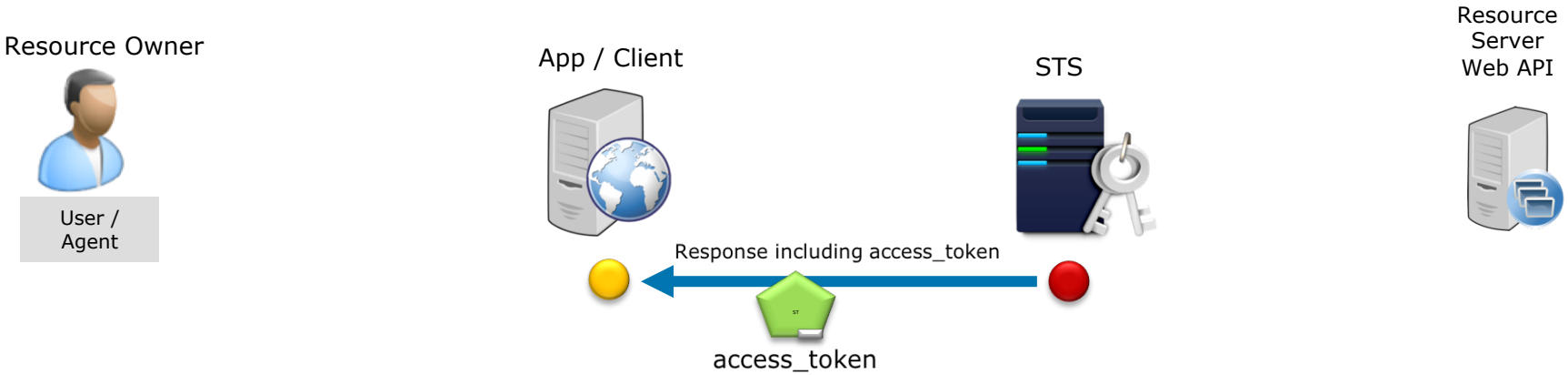
```
// Line breaks for legibility only

POST /{tenant}/oauth2/token HTTP/1.1
Host: https://login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code
&client_id=2d4d11a2-f814-46a7-890a-274a72a7309e
&code=AwABAAAAvPM1KaPlrEqdFSBzjqfTGBCmLdgfSTLEMPGYuNHSUYBrqqf_ZT_p5uEAEJJ_nZ3UmphWygRNy2C3jJ239gV_DBnZ2syeg95Ki-3
&redirect_uri=https%3A%2F%2Flocalhost%2Fmyapp%2F
&resource=https%3A%2F%2Fservice.contoso.com%2F
&client_secret=p@ssw0rd

//NOTE: client_secret only required for web apps
```

Authorize endpoint

Token endpoint

Web / API endpoint

OAuth 2.0 Before you begin?

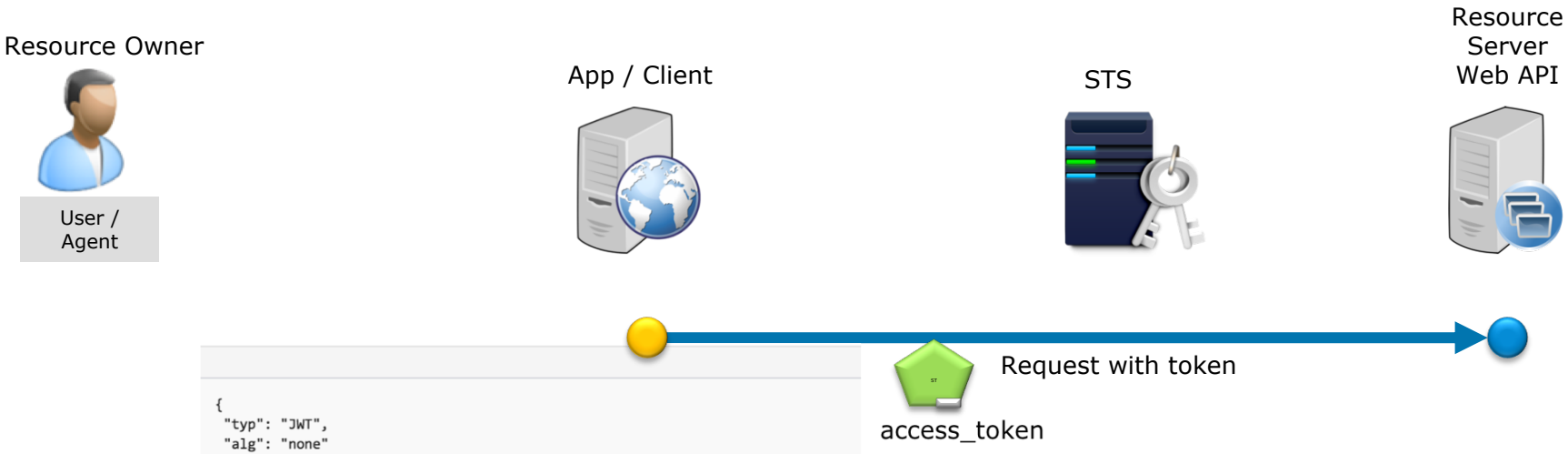


Copy

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ik5HVEZ2ZEstZnl0aEV1THdqcHdBSk9NOW4tQSJ9.eyJhdWQiC
  "token_type": "Bearer",
  "expires_in": "3600",
  "expires_on": "1388444763",
  "resource": "https://service.contoso.com/",
  "refresh_token": "AwABAAAAvPM1KaPlrEqdFSBzjqfTGAMxZGUTdM0t4B4rTfgV29ghDOHRc2B-C_hHeJaJICqjZ3mY2b_YNqmf9SoAylD1F
  "scope": "https%3A%2F%2Fgraph.microsoft.com%2Fmail.read",
  "id_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJhdWQiOiIyZDRkMTFhMi1mODE0LTQ2YTctODkwYS0yNzRhNzJhNzMwOWUiLC
}
```

- Authorize endpoint
- Token endpoint
- Web / API endpoint

OAuth 2.0 Before you begin?



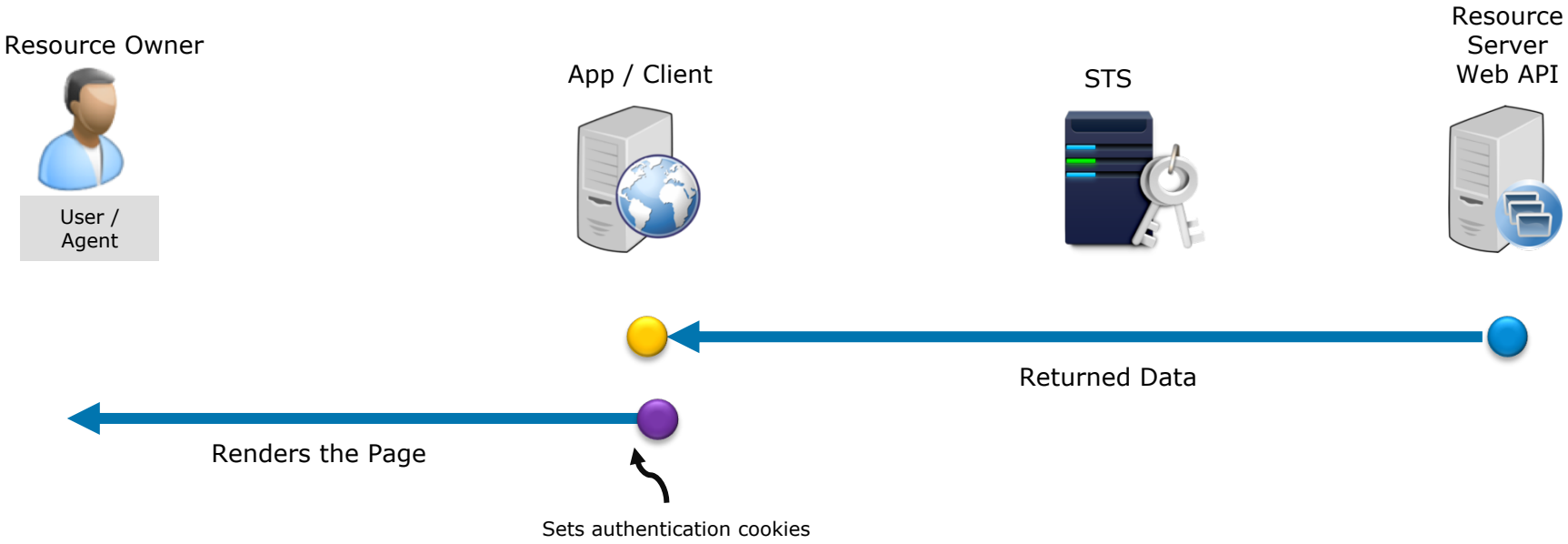

```
{
 "typ": "JWT",
 "alg": "none"
}.
{
 "aud": "2d4d11a2-f814-46a7-890a-274a72a7309e",
 "iss": "https://sts.windows.net/7fe81447-da57-4385-becb-6de57f21477e/",
 "iat": 1388440863,
 "nbf": 1388440863,
 "exp": 1388444763,
 "ver": "1.0",
 "tid": "7fe81447-da57-4385-becb-6de57f21477e",
 "oid": "68389ae2-62fa-4b18-91fe-53dd109d74f5",
 "upn": "frank@contoso.com",
 "unique_name": "frank@contoso.com",
 "sub": "JWvYdCWPhhlpS1Zsf7yYUxShUwtUm5yzPmw_-jX3fHY",
 "family_name": "Miller",
 "given_name": "Frank"
}.
```

Authorize endpoint

Token endpoint

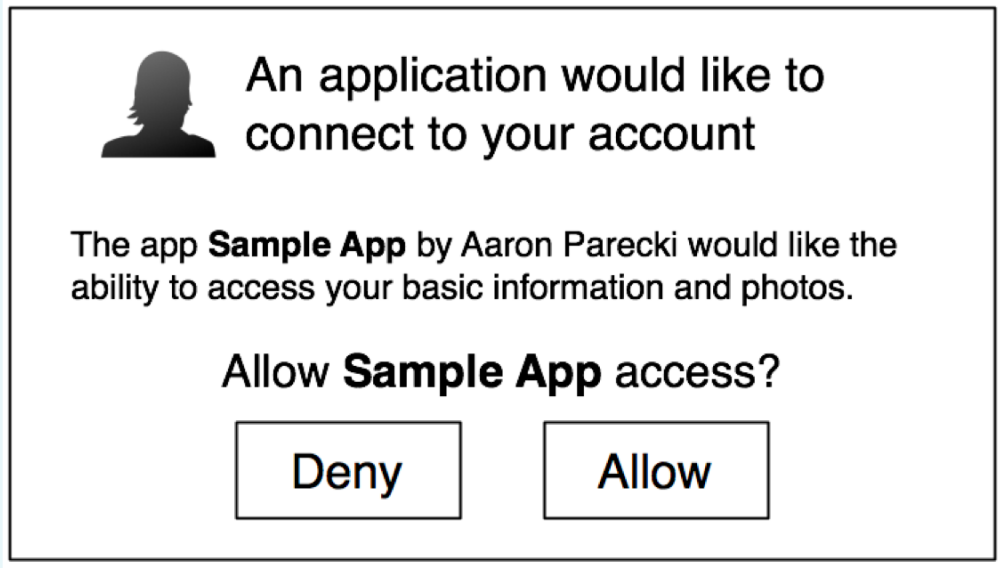
Web / API endpoint

OAuth 2.0 Before you begin?



Oauth 2.0 Required Consent

- Some apps may require user or even admin privileges



Setting an Azure Token Lifetime Policy Via PowerShell

- Deploy latest version of PowerShell

  Connect-AzureAD –Confirm

  Get-AzureADPolicy

  Set-AzureADPolicy -Id <ObjectId FROM GET COMMAND> -DisplayName "OrganizationDefaultPolicyUpdatedScenario" -Definition @('{"TokenLifetimePolicy":{"Version":1,"MaxAgeSingleFactor":"2.00:00:00"}}')

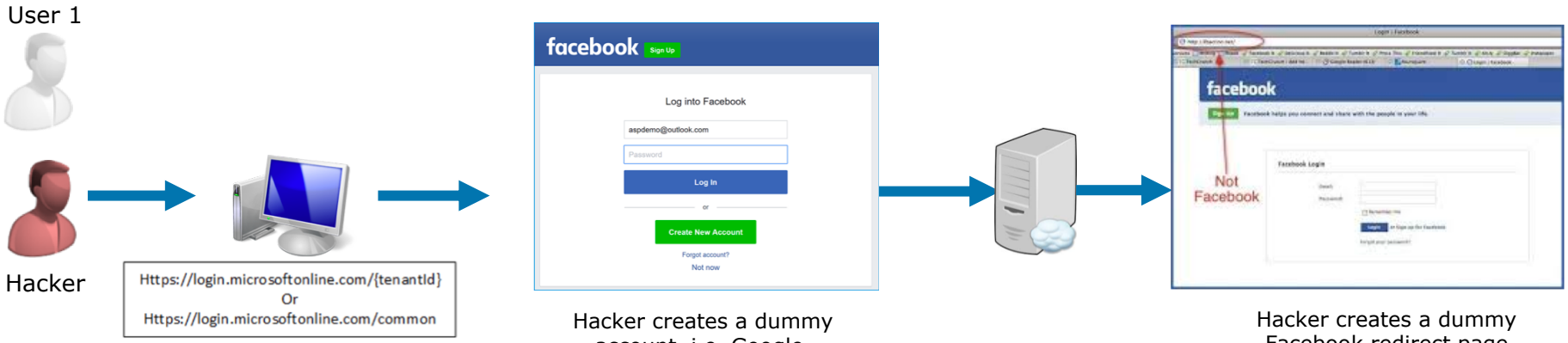
https://docs.microsoft.com/en-us/azure/active-directory/active-directory-configurable-token-lifetimes

OAuth 2.0
Potential Vulnerabilities

OAuth 2.0 Potential Vulnerabilities

1. Lack Of Data Confidentiality and Server Trust
   a) Brute Force Attacks Against the Server
   b) Lack of Server Trust
   c) Man in the Middle attack (Chinese Hack)
2. Insecure Storage of Secrets
   a) Shared secrets on the Server
   b) Secrets on cross-platform clients

OAuth 2.0 Attacking the Connect request



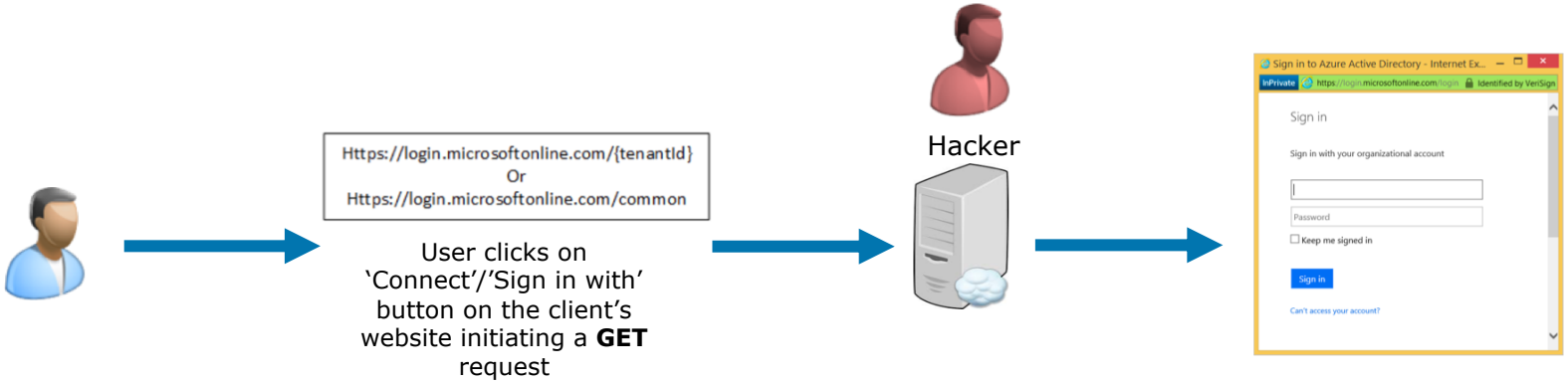
Hacker creates a dummy account. i.e. Google, Facebook etc.

Hacker creates a dummy Facebook redirect page

- Logging out the user on Provider(using CSRF).
- Logging in the user on Provider with the credentials of his/her dummy account(using CSRF).
- Spoofing the 1st request to connect the *Provider* account with *Client*. This can be easily done, as it is just another *GET* request. It is preferred to do this within an iframe so that the victim is

Response_type: *code*
client_id: The *id* issued to the client.
redirect_uri (FAKE URI): The URI where the authorization server will redirect the response to.
scope(optional): The scope to be requested.
state(recommended): An opaque value to maintain state between the request and callback.

OAuth 2.0 Attacking the ‘Connect’ request



- When the victim visits the attacker’s page, he/she is logged out of *Provider* and then gets signed in as a *dummy account*.
- The ‘Connect’ request is then issued which results in the attacker’s dummy account to be connected with the victim’s account on *Client*.
- Note that the victim will not be asked for granting access to the client as the attacker has already approved it.
- Now, the attacker can log in to the victim’s account on Client by signing in with the dummy account on Provider.

Oauth 2.0 Attacking ‘redirect_uri’

1. The attacker is able to leak data(say through XSS) from a page on the client’s domain `https://client.com/vuln`.
2. The attacker injects JavaScript code(if XSS) on that page that sends the URL loaded in the browser(with parameters as well as fragments) to the attacker
3. The attacker creates a webpage that forces the user to visit a malicious link such as `https://provider.com/oauth/authorize?client_id=CLIENT_ID&response_type=code&redirect_uri=https%3A%2F%2Fclient.com%2Fvuln`

Oauth 2.0 Attacking ‘redirect_uri’

4. When the victim loads this link, the user-agent is redirected to `https://client.com/vuln?code=CODE`.

   `CODE` is then sent to the attacker.

5. The attacker can use this code at his/her end to issue an access token by passing it to the authentic *redirect_uri* such as `https://client.com/oauth/callback?code=CODE`.

- This attack is even more dangerous if the authorization server supports the *Implicit grant*. By passing the `response_type=token` the attacker can steal the token directly.

Mitigation: The authorization server MUST:

- Ensure that the “redirect_uri” parameter is present
- If the “redirect_uri” parameter was included in the initial authorization request as described, and if included ensure that their values are identical.

OAuth 2.0 Security Best Practices

- **Always use SSL**. OAuth 2.0 security depends solely on SSL and using OAuth 2.0 without is like sending a password in a plaintext is insecure
- **Always check the SSL certificate** Validate both Source & Authenticity
- **Always use refresh tokens** and make access tokens short-lived
  - They can be used to avoid access tokens living forever
  - They are revocable. When the user changes the password, the token can be revoked
  - They can be used for updating access token content
- **Choose the lifetime for access tokens** and refresh tokens properly (e.g. Financial systems)

Operating System Vulnerabilities …

OS Vulnerabilities …

- Every OS has variabilities, bugs
- Often built on old technologies, meaning thaty old flaws remain
- The point is that if you can get physical Access to a device, then it can be hacked.
- All modern OSes have Key Escrow. Giving an employer / law enforcement the ability to back door in.

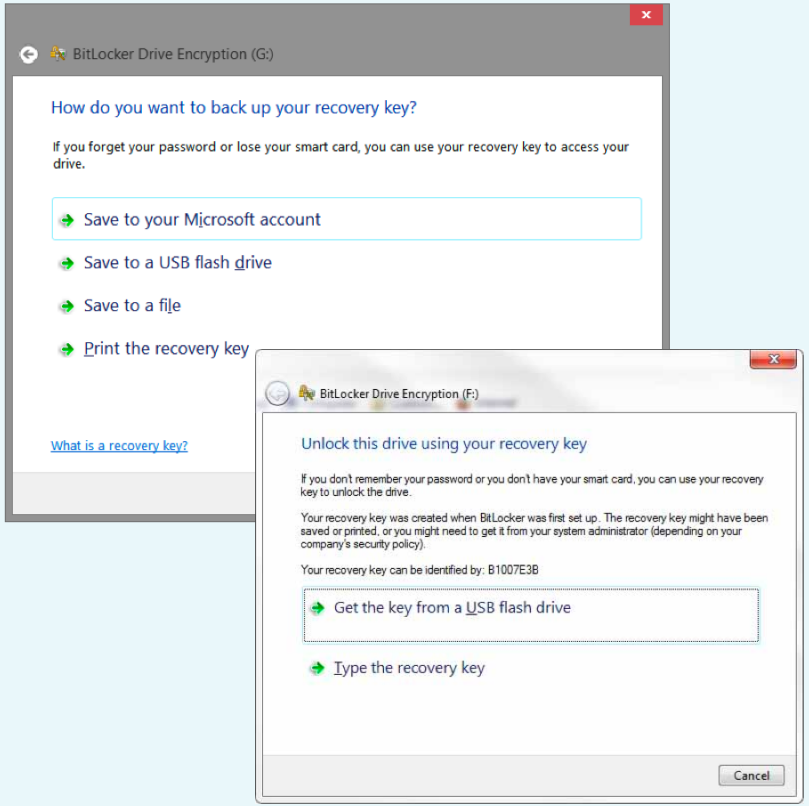
The Great Windows NSA Scandal

```
Type Bits/KeyID Date User ID
pub 1024/346B5095 1999/09/06 Microsoft's CAPI key <postmaster@microsoft.com>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQCPAzfTc8YAAAEEALJz4nepw3XHC7dJPlKws2li6XZiatYJ
2WlRggxFfHtMSJO9FJ3ieaOfbskm01RNs0kfoumvG/gmCzsI
F8C4d+r32p0C3V+FcoVOXJDpsQz7rq+Lj+HfUEe8GIKaUxSZ
AAG0L01pY3Jvc29mdCdzIENBUEkga2V5IDxwb3N0bWFzdGVy
b20+iQEVAwUQN9Nz5j57yqgoskVRAQFr/gf8DGm1hAxWBmx/
yI5mub0ie1HRLExP7lVJezBTyRryV3tDv6U3OIP+KZDthdXk
xl6Q7m7oB76SKfNaWgosZxqkE5YQrXXGsn3oVZhV6yBALekW
NvMTYRUz+MdrRFcEFDhFntblI8NlQenlX6CcnnfOkdR7ZKyI
TSjBT0swYbHX+3EVt8n2nwxWb2ouNmnm9H2gYfXHikhXrwtj
Rp+crFOB32sTO1ib2sr7GY7CZUwOpDqRxo8KmQZyhaZqz1x6
=ms8C
-----END PGP PUBLIC KEY BLOCK-----
```

```
Type Bits/KeyID Date User ID
pub 1024/51682D1F 1999/09/06 NSA's Microsoft CAPI key <postmaster@nsa.gov>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQCPAzfTdH0AAAEEALqOFf7jzRYPtHz5PitNhCYVryPwZZJk2B7cNaJ9OqRQiQoi
e1YdpAH/OQh3HSQ/butPnjUZdukPB/0izQmczXHoW5f1Q5rbFy0y1xy2bCbFsYij
4ReQ7QHrMb8nvGZ7OW/YKDCX2LOGnMdRGjSW6CmjK7rW0veqfoypgF1RaC0fABEB
AAG0LU5TQSdzIE1pY3Jvc29mdCBDQVBJIGtleSA8cG9zdG1hc3RlckBuc2EuZ292
PokBFQMFEDfTdJE+e8qoKLJFUQEBHnsH/ihUe7oq6DhU1dJjvXWcYw6p1iW+0euR
YfZjwpzPotQ8m5rC7FrJDUbgqQjoFDr++zN9kD9bjNPVUx/ZjCvSFTNu/5X1qn1r
it7IHU/6Aem1h4Bs6KE5MPpjKRxRkqQjbW4f0cgXg6+LV+V9cNMylZHRef3PZCQa
5DOI5crQ0IWyjQCt9br07BL9C3X5WHNNRsRIr9WiVfPK8eyxhNYl/NiH2GzXYbNe
UWjaS2KuJNVvozjxGymcnNTwJltZK4RLZxo05FW2InJbtEfMc+m823vVltm9l/f+
n2iYBAaDs6I/0v2AcVKNy19Cjncc3wQZkaiIYqfPZL19kT8vDNGi9uE=
=PhHT
-----END PGP PUBLIC KEY BLOCK-----
```

Windows Key Escrow

- Administrator Account reset password feature
- EFS Key Escrow
- MBAM - Bitlocker Key escrow features, numerous
- Certificate Key escrow
- Developer escrow Features



Hardware Vulnerabilities …

Meltdown

Meltdown breaks the most fundamental isolation between user applications and the operating system. This attack allows a program to access the memory, and thus also the secrets, of other programs and the operating system.

If your computer has a vulnerable processor and runs an unpatched operating system, it is not safe to work with sensitive information without the chance of leaking the information. This applies both to personal computers as well as cloud infrastructure.

Spectre

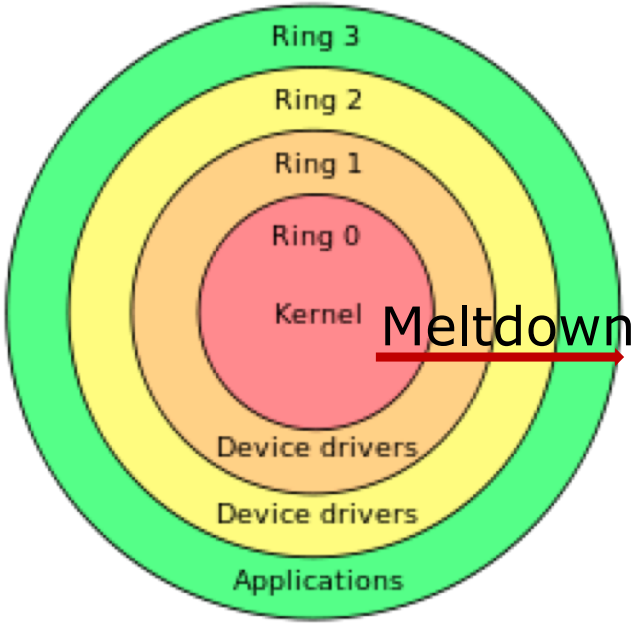
Spectre breaks the isolation between different applications. It allows an attacker to trick error-free programs, which follow best practices, into leaking their secrets. In fact, the safety checks of said best practices actually increase the attack surface and may make applications more susceptible to Spectre

Spectre is harder to exploit than Meltdown, but it is also harder to mitigate.

nic

The Broken Rings of Protection

- On January 3rd 2018, Intel, AMD and ARM Holdings, as well as a number of OS Vendors reported a series of vulnerabilities
- Variant 1: bounds check bypass (CVE-2017-5753)
- Variant 2: branch target injection (CVE-2017-5715)
- Variant 3: rogue data cache load (CVE-2017-5754)



Both Kernel Mode & User Mode now theoretically exploitable

Meltdown & Spectre

- Meltdown and Spectre vulnerabilities negatively impact the security of virtually every computer in the world today
- Allows attackers to gain control of a computer's processor and steal data located on that computer
- Meltdown affects a wide range of systems, this includes devices running any but the most recent and patched versions of iOS, Linux, MAC OS, or Windows
- Many servers and cloud services were impacted, as well as the majority of smart devices and embedded devices using based processors (mobile devices, smart TVs and others), including a wide range of networking equipment

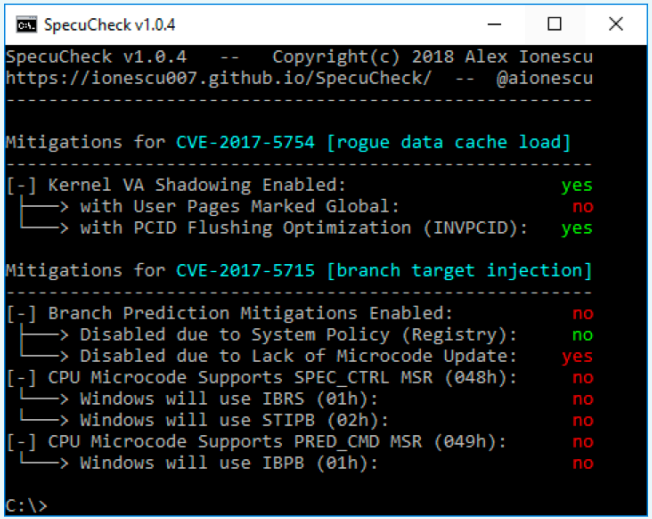
How does the Meltdown exploit work?

- Meltdown exploits a race condition, inherent in the design of many modern CPUs
- Occurs between memory access and privilege checking during the instruction process
- Along with a cache side-channel attack, allows a process to bypass the normal privilege checks that isolate the exploit process from accessing data belonging to the operating system and other running processes
- Allows an unauthorized process to read data from any address that is mapped to the current process' memory space
- Data from an unauthorized address will almost always be temporarily loaded into the cache during speculative execution—from which the data can be recovered irrespective of Permissions.

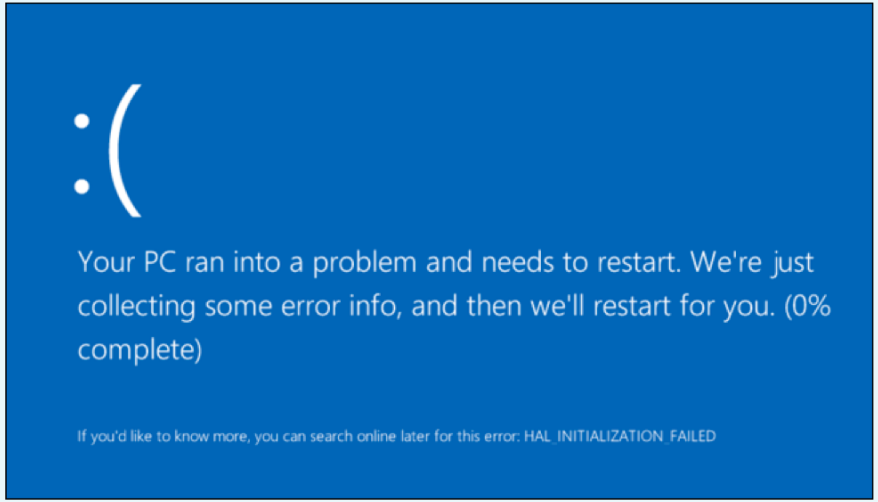
How do I know if I’m affected?

https://github.com/topics/meltdown

Specucheck PowerShell Script



Antivirus BSOD




```
Key="HKEY_LOCAL_MACHINE" Subkey="SOFTWARE\Microsoft\Windows\CurrentVersion\QualityCompat" Value="cadca5fe-87d3-4b96-b7fb-a231484277cc" Type="REG_DWORD”
```

What can I do?

```
Install-Module SpeculationControl
```

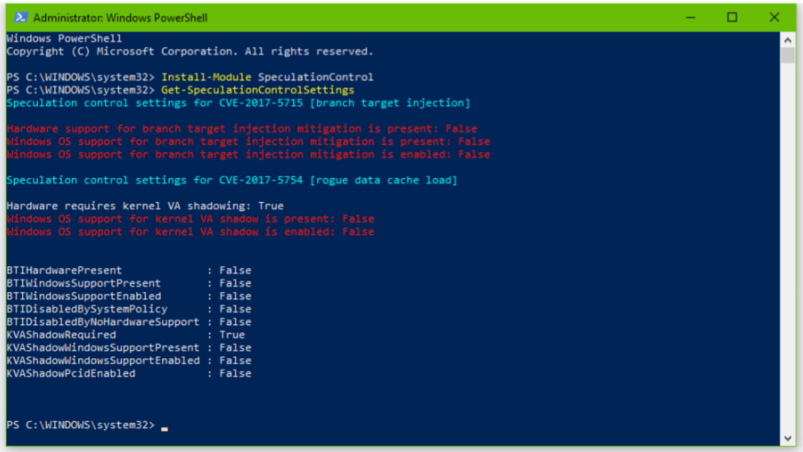
↓

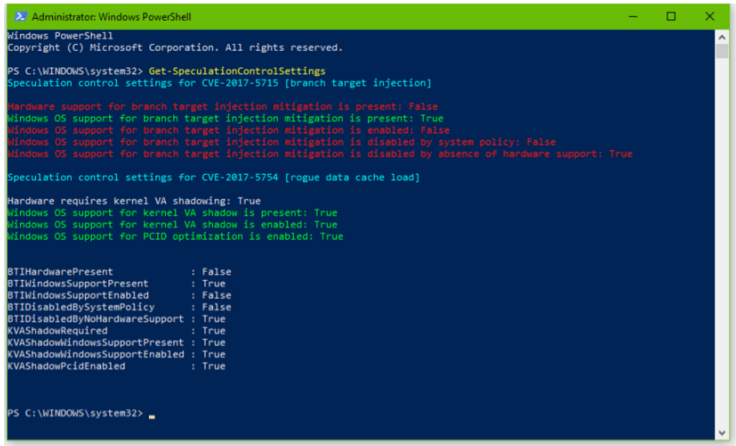
```
Set-ExecutionPolicy Bypass
```

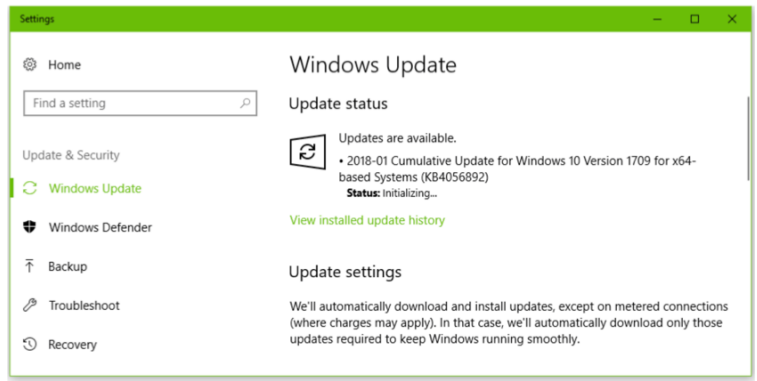
↓

```
Get-SpeculationControlSettings
```

↓

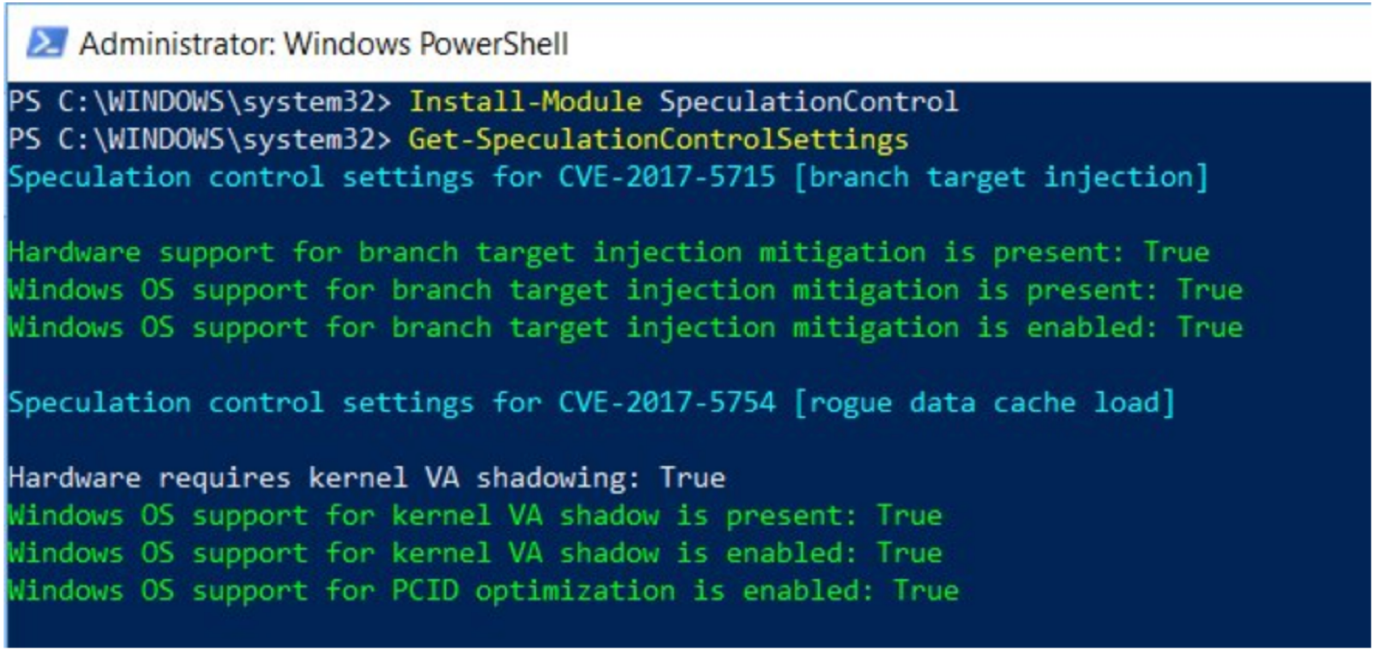






https://www.bleepingcomputer.com/news/microsoft/how-to-check-and-update-windows-systems-for-the-meltdown-and-spectre-cpu-flaws/

Success!



Important! → `Set-ExecutionPolicy Restricted`

Mitigations

	Google Project Zero (GPZ) Research Title	Details
Variant One	Bounds Check Bypass	Resolved by software / OS updates to be made available by system vendors and manufacturers. Negligible performance impact expected.
Variant Two	Branch Target Injection	Differences in AMD architecture mean there is a near zero risk of exploitation of this variant. Vulnerability to Variant 2 has not been demonstrated on AMD processors to date.
Variant Three	Rogue Data Cache Load	Zero AMD vulnerability due to AMD architecture differences.

2018: The year ahead

1. Cybercriminals Get More Sophisticated—Developing Multiple New Ways to Attack Their Victims
2. Malware Invades Hardware at Increasing Rates
    1. Firmware in 26 Android devices & fake Apple firmware
3. The Cyberattack Surface Expands Exponentially
4. Giant Leap in Adoption of Security Oriented AI&ML
    1. Internet Connected Devices, IoT, Cloud infrastructure etc.
5. Cybersecurity Augments Prevention with Resiliency
    1. Assumption that data breaches are inevitable, and instead of breach prevention, organizations will invest in breach containment & rapid recovery

Session Review

- Latest vulnerabilities
- Venerability Categories
- Exploring the technology
  - What it is?
  - How it works?
  - How it could be exploited
  - Countermeasures
- Review

Andy Malone

(Scotland, UK)

- Thanks for Attending
- Meet the Author event @NIC
- Purchase your signed copy at the Glasspaper Booth
- Author off the Sc-Fi Thrillers
- “The Seventh Day” & Shadows Rising”

Resources

Slides and demos from the conference will be available at github.com/nordicinfrastructureconference/2018 (bit.ly/2y7JhA3)