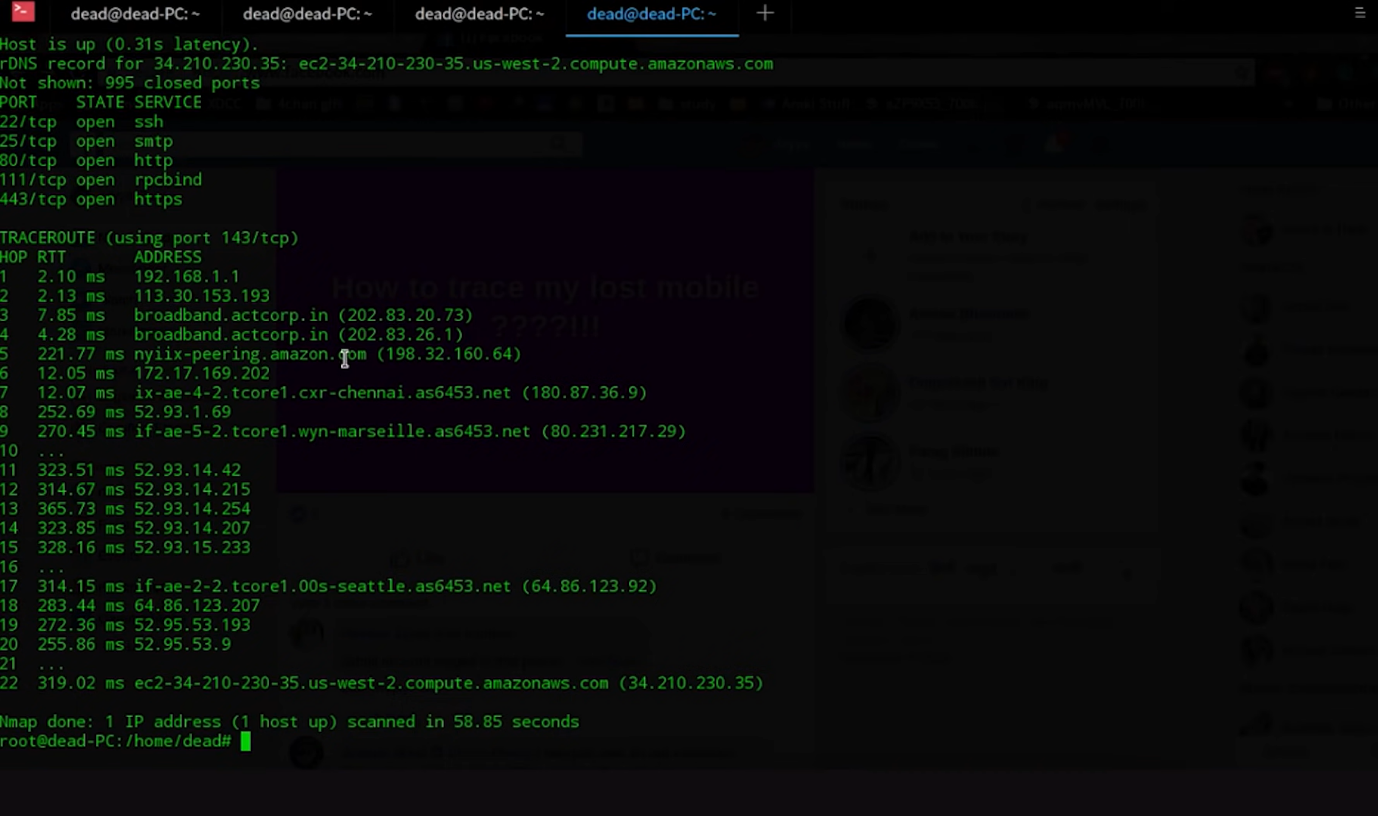
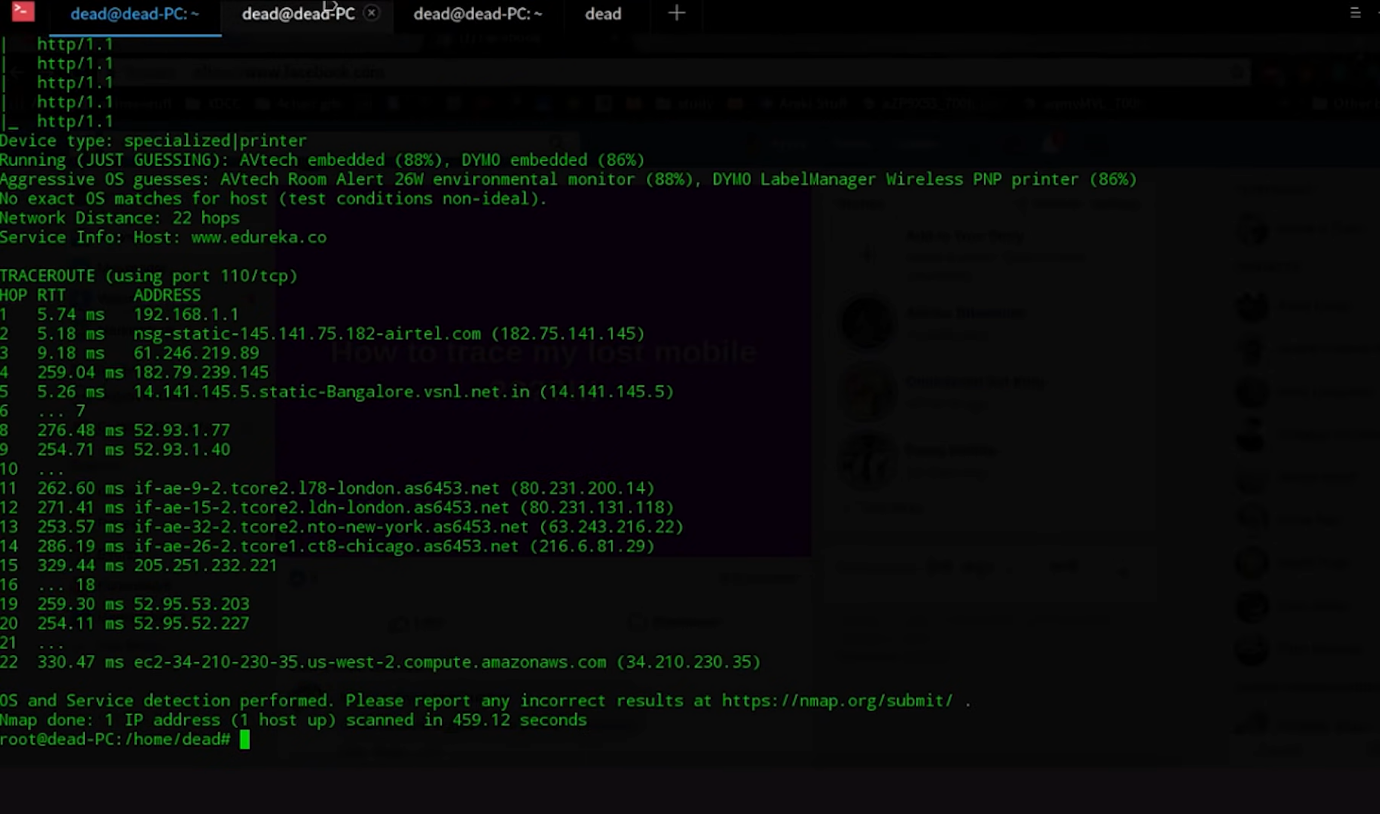
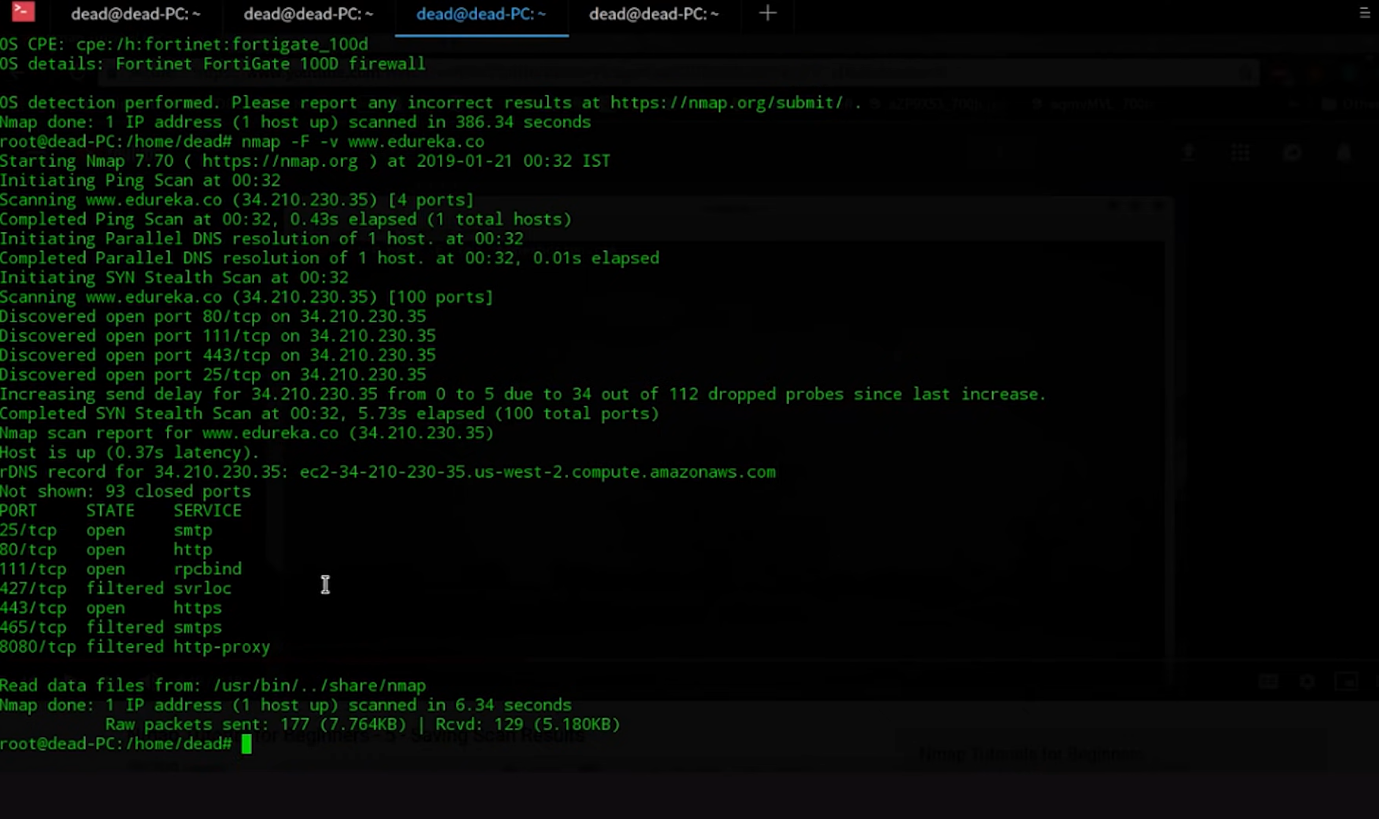
Name-Shubhankar Vinay Gokhale

Project Name : -  
Cyber Security October Major Project

1.Perform Scanning Module by using Nmap tool (Download from Internet) and scan kalilinux and  
Windows 7 machine and find the open/closed ports and services running on machine  
Hacker Machine : Windows 10  
Victim machine : Kali Linux and Windows 7

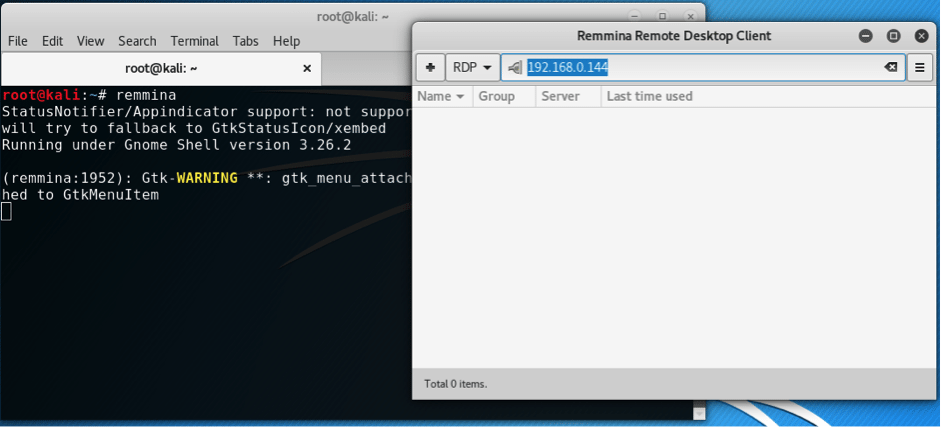






2 Test the System Security by using metasploit Tool from kali linux and hack the windows 7 / win  
dows10. Execute the commands to get the keystrokes / screenshots / Webcam and etc.,

Write a report on vulnerability issue along with screenshots how you performed and suggest the security  
patch to avoid these type of attacks  
Hacker Machine : Kali Linux  
Victim machine : Windows XP / Windows 7



Write a report on vulnerability issue along with screenshots how you performed and suggest the security  
patch to avoid these type of attacks

**Programmers and IT professionals often express confusion regarding the distinction between authorization and authentication. The use of the abbreviation auth for both terms increases the haziness that surrounds them.**

**Let’s define and clarify the distinction:**

* **Authentication: Verifying that a user is (or at least appears to be) the person they say they are.**
* **Authorization: Granting a user access to a specific resource, or permission to perform a particular action.**

**Stated another way, authentication is knowing who an entity is, while authorization is what a given entity can do. With this in mind, let’s explore 10 common internet vulnerability issues.**

## **Injection Flaws**

**Injection flaws result from a classic failure to filter untrusted input. Injection flaws can happen when we pass unfiltered data to the SQL server (SQL injection), to the browser (**[**via Cross Site Scripting**](https://www.toptal.com/security/10-most-common-web-security-vulnerabilities#CommonMistake_XSS)**), to the LDAP server (LDAP injection), or anywhere else. The problem here is that the attacker can inject commands to hijack clients’ browsers, resulting in loss of data.**

**Anything that your application receives from an untrusted source must be filtered, preferably according to a whitelist. Using a blacklist to this end is not recommended, as it is difficult to configure properly. A blacklist is also considered easy for a hacker to bypass. Antivirus software products typically provide stellar examples of failing blacklists. Pattern matching does not work.**

**Prevention: Protecting against injection is “simply” a matter of filtering our input and considering which senders can be trusted. Filtering is quite an undertaking because we need to process all input unless it can unquestionably be trusted.**

**If we filter 999 inputs in a system with 1,000 inputs, we still have one field that can be the Achilles’ heel that brings down our system.**

**Using**[**Second Order SQL Injection**](https://en.wikipedia.org/wiki/SQL_injection#Second_Order_SQL_Injection)**to inject one SQL query result into another is also considered dangerous. It could seem like a good idea because the database is trusted. But if the perimeter is not, our input could originate indirectly from a malicious source.**

**Since filtering is pretty hard to get right, it is advisable to rely on our framework’s filtering functions. They are proven to work and thoroughly scrutinized. If you do not already use a framework, consider the server security benefits of moving to one. .**

## **Broken Authentication**

**Problems that might occur during broken authentication don’t necessarily stem from the same root cause. Rolling your own authentication code is not recommended, as it is hard to get right. There are myriad possible pitfalls, and here are a few:**

1. **The URL might contain the session ID and leak it in the referer header.**
2. **Passwords might not be encrypted in storage and/or transit.**
3. **Session IDs might be predictable, making it a little too easy to gain unauthorized access.**
4. **Session fixation might be possible.**
5. **Session hijacking could occur if timeouts are not implemented correctly, or if using HTTP (no SSL security), etc.**

**Prevention: The most straightforward way to avoid the web security vulnerabilities related to broken authentication is to implement a framework. If you roll your own code, be extremely paranoid and educate yourself on the potential issues that could arise.**

## **Cross-Site Scripting (XSS)**

**An attacker sends on input JavaScript tags to your web application. When this input is returned to the user unsanitized, the user’s browser would execute it. This is a fairly widespread input sanitization failure, essentially a subcategory of**[**injection flaws**](https://www.toptal.com/security/10-most-common-web-security-vulnerabilities#CommonMistake_Injection)**). CSS can be as simple as crafting a link and persuading a user to click it, or it can be something much more sinister. For example, on page load, the script would run and be used to post your cookies to the attacker.**

**Prevention: Simply put, don’t return HTML tags to the client. This would also protect you from HTML injection, which is when an attacker injects plain HTML content (such as images or loud but invisible flash players). To implement this solution, convert all**[**HTML entities**](https://www.toptal.com/designers/htmlarrows/symbols/)**to return something else. For example, convert <script> to return &lt;script&gt;. Alternatively, you can use regular expressions to strip away HTML tags using regular expressions on < and >. But this is dangerous because some browsers may not interpret severely broken HTML. Better to convert all characters to their escaped counterparts.**

## **Insecure Direct Object References**

**This is a classic case of trusting user input and paying the price by inheriting a resultant security vulnerability. A direct object reference means that an internal object (e.g., a file or a database key) is exposed to the user, leaving us vulnerable to attack. The attacker can provide this reference, and if authorization is either not enforced or broken, the attacker gets in.**

**For example, the code has a download.php module that reads and lets the user download files, using a CGI parameter to specify the file name (e.g., download.php?file=something.txt). If the developer omitted authorization from the code, the attacker can now use it to download system files accessible to the user running PHP (e.g., the application code or random server data like backups).**

**Another example of insecure direct object reference vulnerability is a password reset function that relies on user input to determine their identity. After clicking the valid URL, an attacker could modify the username field in the URL to say something like “admin.”**

**Incidentally, I have seen both of these examples often “in the wild.”**

**Prevention: Perform user authorization properly and consistently, and whitelist the choices. More often than not, the vulnerability can be avoided altogether by storing data internally and not relying on data being passed from the client via CGI parameters. Session variables in most frameworks are well suited to this purpose.**

## **Security Misconfiguration**

**In my experience, it is common to encounter misconfigured web servers and applications. Some examples:**

1. **Running an application with debug enabled in production**
2. **Having directory listing (which leaks valuable information) enabled on the server**
3. **Running outdated software (think WordPress plugins, old PhpMyAdmin)**
4. **Running unnecessary services**
5. **Not changing default keys and passwords (which happens more frequently than you’d believe)**
6. **Revealing error handling information (e.g., stack traces) to potential attackers**

**Prevention: Have a good (preferably automated) “build and deploy” process, which can run tests on deploy. The poor man’s security misconfiguration solution is post-commit hooks, to prevent code from going out with default passwords and/or development stuff built in.**

## **Sensitive data exposure**

**This web security vulnerability is about crypto and resource protection. Sensitive data should be encrypted at all times, including in transit and at rest. No exceptions. Credit card information and user passwords should never travel or be stored unencrypted, and passwords should always be hashed. Obviously, the crypto/hashing algorithm must not be a weak one. When in doubt, web security standards recommend**[**AES (256 bits and up)**](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard)**and**[**RSA (2048 bits and up)**](https://en.wikipedia.org/wiki/RSA_numbers#RSA-2048)**.**

**It cannot be overemphasized that session IDs and sensitive data should not travel in URLs. Cookies with sensitive data should have the “secure” flag on.**

**Prevention:**

* **In transit: Use**[**HTTPS**](https://en.wikipedia.org/wiki/HTTP_Secure)**with a proper certificate and**[**PFS (Perfect Forward Secrecy)**](https://en.wikipedia.org/wiki/Forward_secrecy#Perfect_forward_secrecy)**. Do not accept anything over non-HTTPS connections. Have the “secure” flag on cookies.**
* **In storage: Reduce your exposure to this vulnerability. If you don’t need sensitive data, virtually shred it. The data you don’t have can’t be**[**stolen**](https://krebsonsecurity.com/2017/12/the-market-for-stolen-account-credentials/)**. Do not store credit card information, and you will not need to have to deal with being**[**PCI compliant**](https://www.pcicomplianceguide.org/faq/)**. Sign up with a payment processor like**[**Stripe**](https://stripe.com/)**or**[**Braintree**](https://www.braintreepayments.com/)**. Store and encrypt sensitive data, and ensure all passwords are hashed using [bcrypt](https://bcrypt.sourceforge.net/" \t "_blank). If you don’t use bcrypt, educate yourself on**[**salting**](https://en.wikipedia.org/wiki/Salt_(cryptography))**and**[**rainbow tables**](https://en.wikipedia.org/wiki/Rainbow_table)**.**

**And at the risk of stating the obvious, do not store the encryption keys near their protected data. That’s like storing your bike with a lock that has the key in it. Protect your backups with encryption and keep your keys private. And of course, don’t lose the keys!**

## **Missing Function Level Access Control**

**This is a failure that happens if proper authorization is not performed when a function is called on the server. Developers tend to assume that since the server side generates the UI, the client would not be able to access functionality that is not supplied by the server. It is not as simple as that, as an attacker can always forge a request to the “hidden” functionality. An attacker will not be deterred by the fact that the desired functionality is not easily accessible. Imagine there’s an /admin panel, and the button is only present in the UI if the user is actually an admin. Nothing keeps an attacker from discovering and misusing this functionality if authorization is missing.**

**Prevention: On the server side, authorization must always be performed.**

## **Cross-Site Request Forgery (CSRF)**

**In a CSRF—also referred to as a**[**confused deputy**](https://en.wikipedia.org/wiki/Confused_deputy_problem)**attack—a malicious third party fools the browser into misusing its authority to do something for the attacker.**

**In the case of CSRF, a third-party site uses your browser, cookies, and session to issue a request to a target site (e.g., your bank). If on one browser tab you are logged in to your bank, and if your bank is vulnerable to this type of attack, then another tab can be controlled to make your browser misuse its credentials on the attacker’s behalf, which results in the confused deputy problem. The deputy is the browser that misuses its authority (session cookies) to perform the attacker’s instructions.**

**Consider this example: Attacker Alice wants to lighten target Todd’s wallet by transferring some of his money into her account.**

**To send money, Todd accesses the following URL: https://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243 at his bank which, incidentally, is vulnerable to CSRF attacks. After Todd performs his transaction, a success page displays and the transfer is complete.**

**Alice is aware that Todd frequently visits a site she controls at https://blog.aliceisawesome.com, so Alice places the following snippet on her site: <img src=https://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243 width=0 height=0 />**

**When Todd next visits Alice’s website, his browser wrongly thinks the snippet links to an image. The browser automatically issues an HTTP GET request to fetch the picture. But instead of getting an image to display in the browser, the request instructs Todd’s bank to transfer $1,500 to Alice.**

**Incidentally, in addition to demonstrating the CSRF vulnerability, this example also demonstrates altering the server state with an**[**idempotent**](https://en.wikipedia.org/wiki/Idempotence)**(safe)HTTP GET request. This in itself a serious vulnerability. HTTP GET requests must be idempotent, meaning that they cannot alter the resource that is accessed. Never use idempotent methods to change the server state.**

**Fun fact: CSRF is also the method people used for cookie-stuffing in the past until affiliates got wiser.**

**Prevention: Store a secret token in a hidden form field, inaccessible to a third-party site. This, of course, requires you to verify the hidden field. Some sites may ask for a password before allowing you to modify sensitive settings (like a password reminder email). I suspect this could be to prevent the misuse of your abandoned sessions on public computers.**

## **Using Components With Known Vulnerabilities**

**The title says it all. I’d classify this one as more of a maintenance/deployment issue. Before incorporating new code, do some research, and possibly some auditing. Using code from a random person on**[**GitHub**](https://github.com/)**, for example, may be convenient, but it is not without risk of serious web security vulnerability.**

**I have seen many instances where sites got owned (i.e., where an outsider gains administrative access to a system), because third-party software (e.g., WordPress plugins) remained unpatched for years in production. If you think they will not find your hidden phpmyadmin installation, let me introduce you to [DirBuster](https://www.kali.org/tools/dirbuster/" \t "_blank).**

**The lesson here is that software development does not end when the application is deployed. There has to be documentation, tests, and plans on how to maintain and keep the application updated, especially if it contains third-party or open source components.**

**Prevention:**

* **Do not be a copy-paste coder. Carefully inspect the piece of code you are about to put into your software, as it might be broken or, in some cases, intentionally malicious. Web security attacks are sometimes unwittingly invited in this way.**
* **Stay up-to-date with the latest versions of everything that you trust, and have a plan to update regularly. To stay on top of new security vulnerabilities, subscribe to your products’ newsletters.**

## **Unvalidated Redirects and Forwards**

**This is yet another input filtering issue. Suppose that the target site has a redirect.php module that takes a URL as a GET parameter. Manipulating the parameter can create a URL on targetsite.com that redirects the browser to malwareinstall.com. A user would see the link as targetsite.com/blahblahblah, which looks innocuous enough to trust and click. But clicking this link could transfer the user to a malware drop (or any other malicious) page. Alternatively, the attacker might redirect the browser to targetsite.com/deleteprofile?confirm=1.**

**It is worth mentioning that stuffing unsanitized user-defined input into an HTTP header might lead to**[**header injection**](https://en.wikipedia.org/wiki/HTTP_header_injection)**, which is pretty bad.**

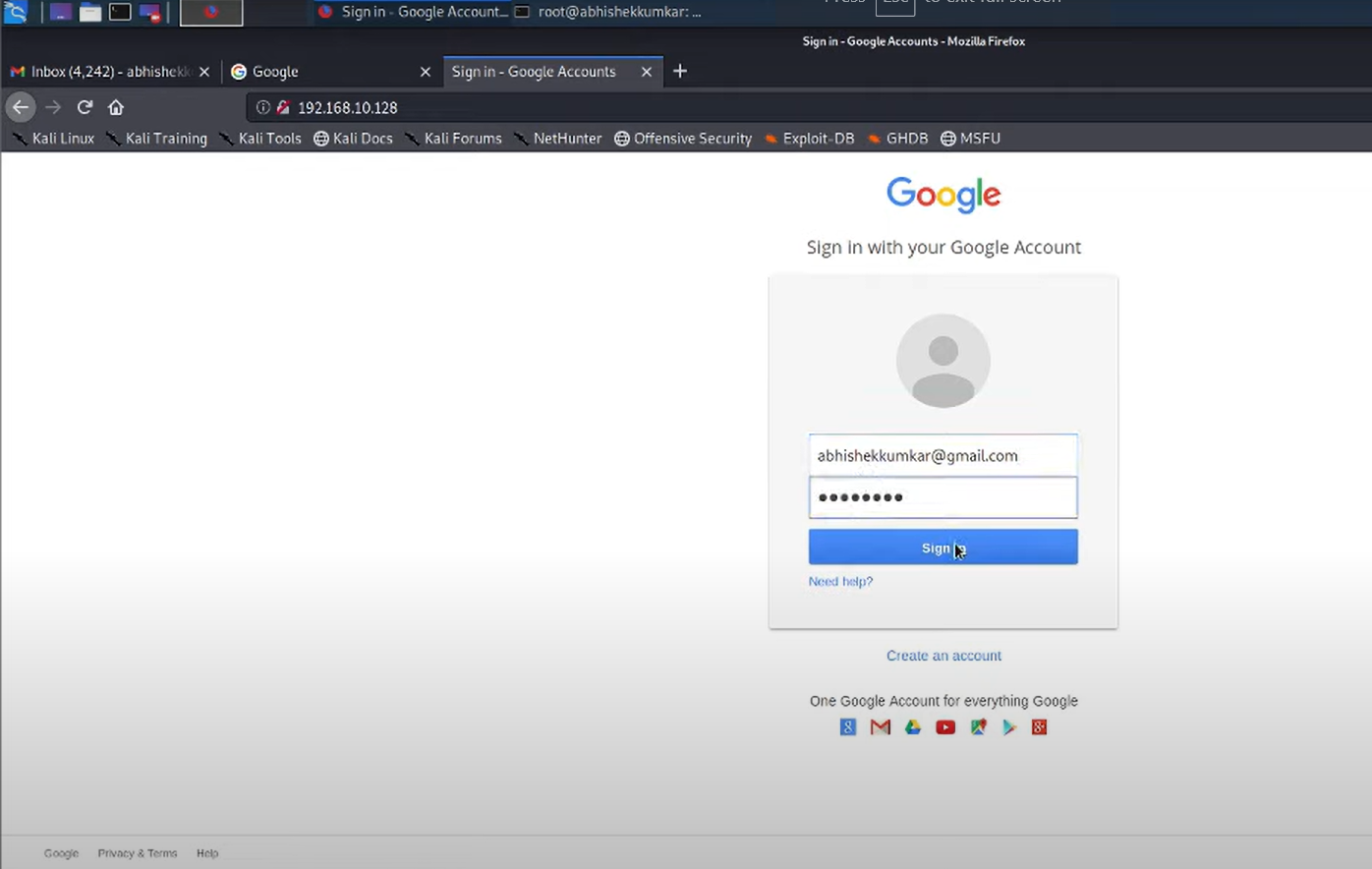
**Prevention: Options include:**

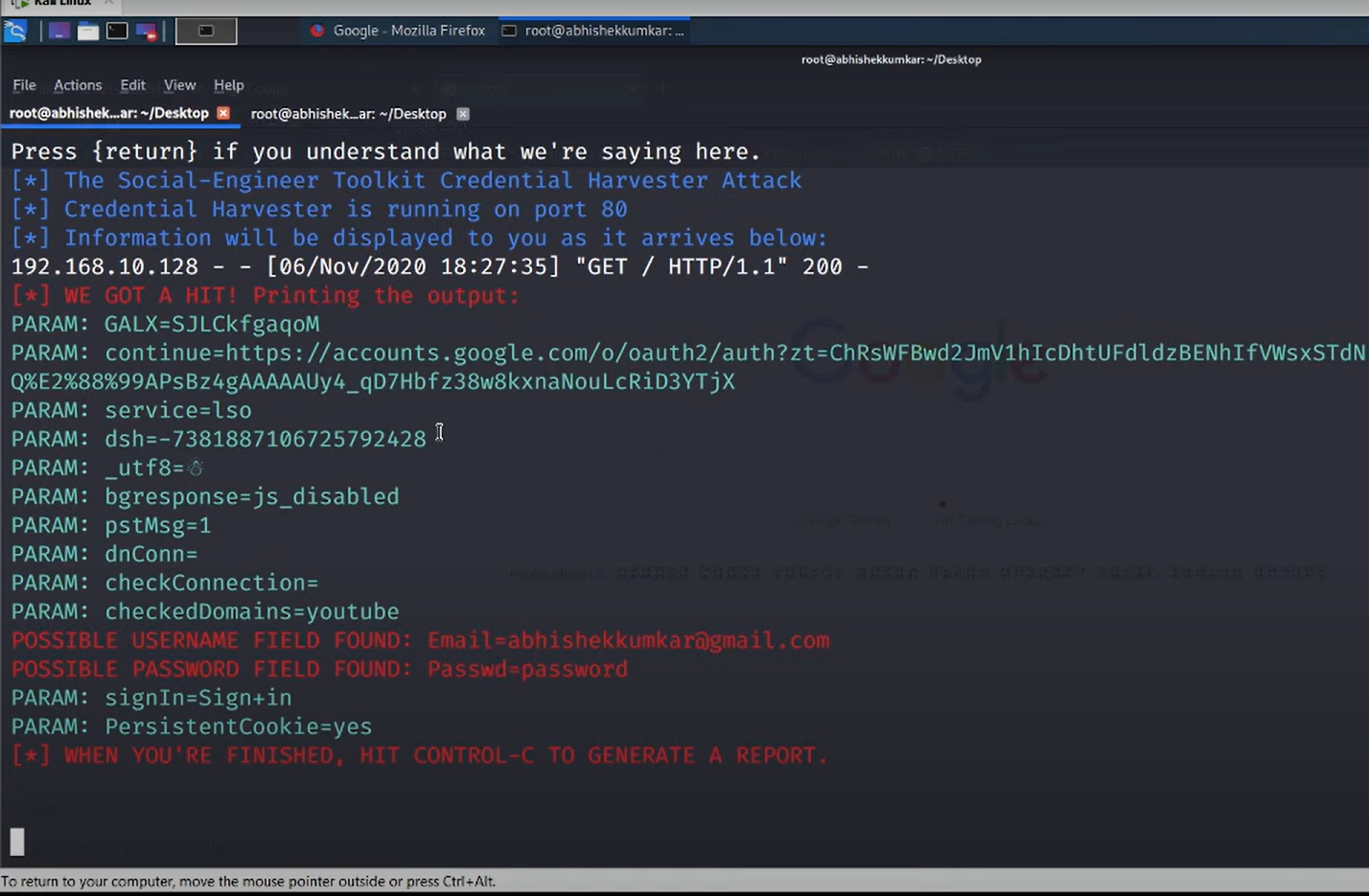
* **Don’t do redirects; these are seldom necessary.**
* **When a redirect is necessary, have a static list of valid redirect locations.**
* **Whitelist the user-defined parameter. Note this can be tricky.**

3. Use SET Tool and create a fake Gmail page and try to capture the credentials in command line and  
Hacker Machine : Kali Linux

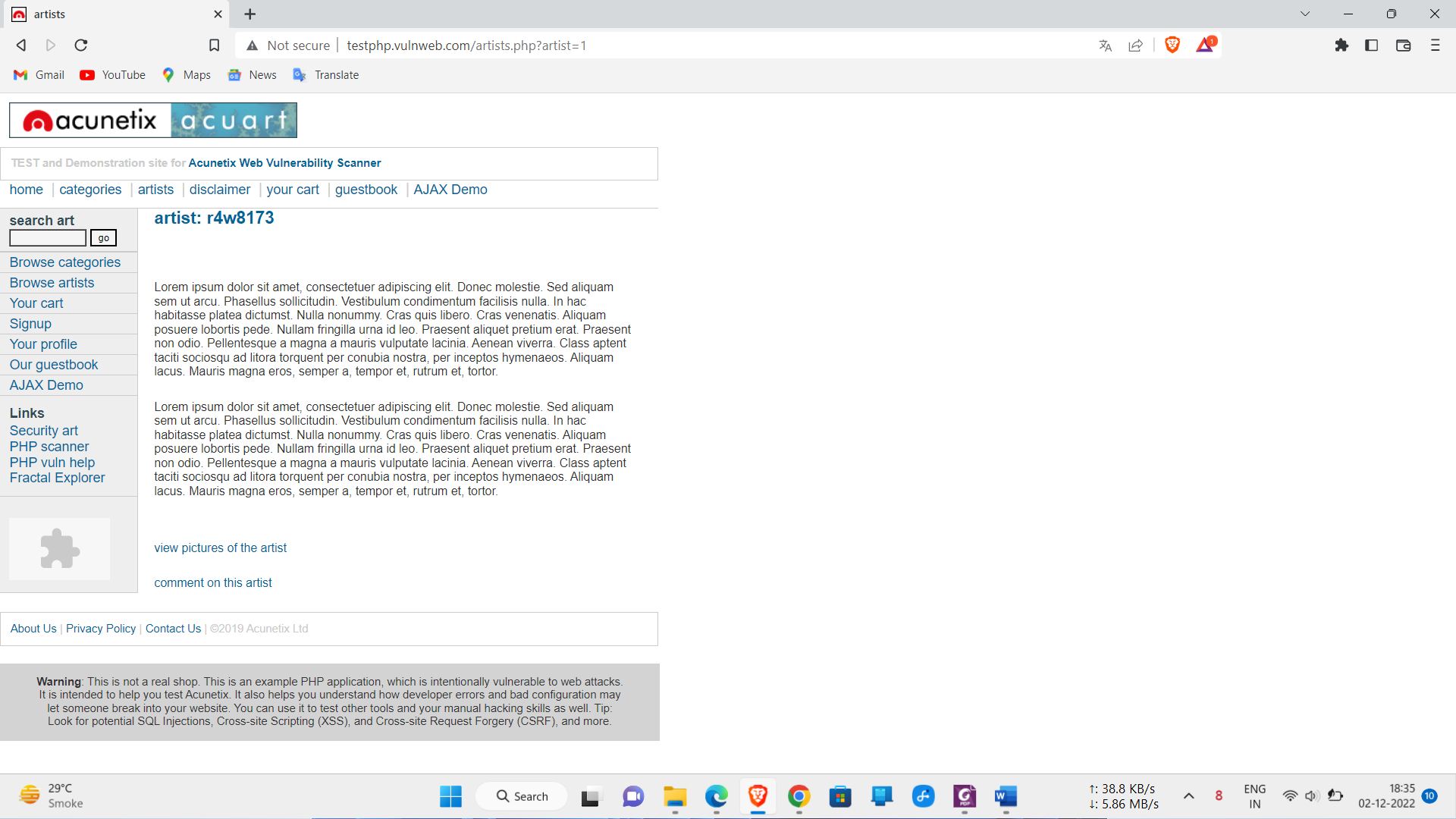
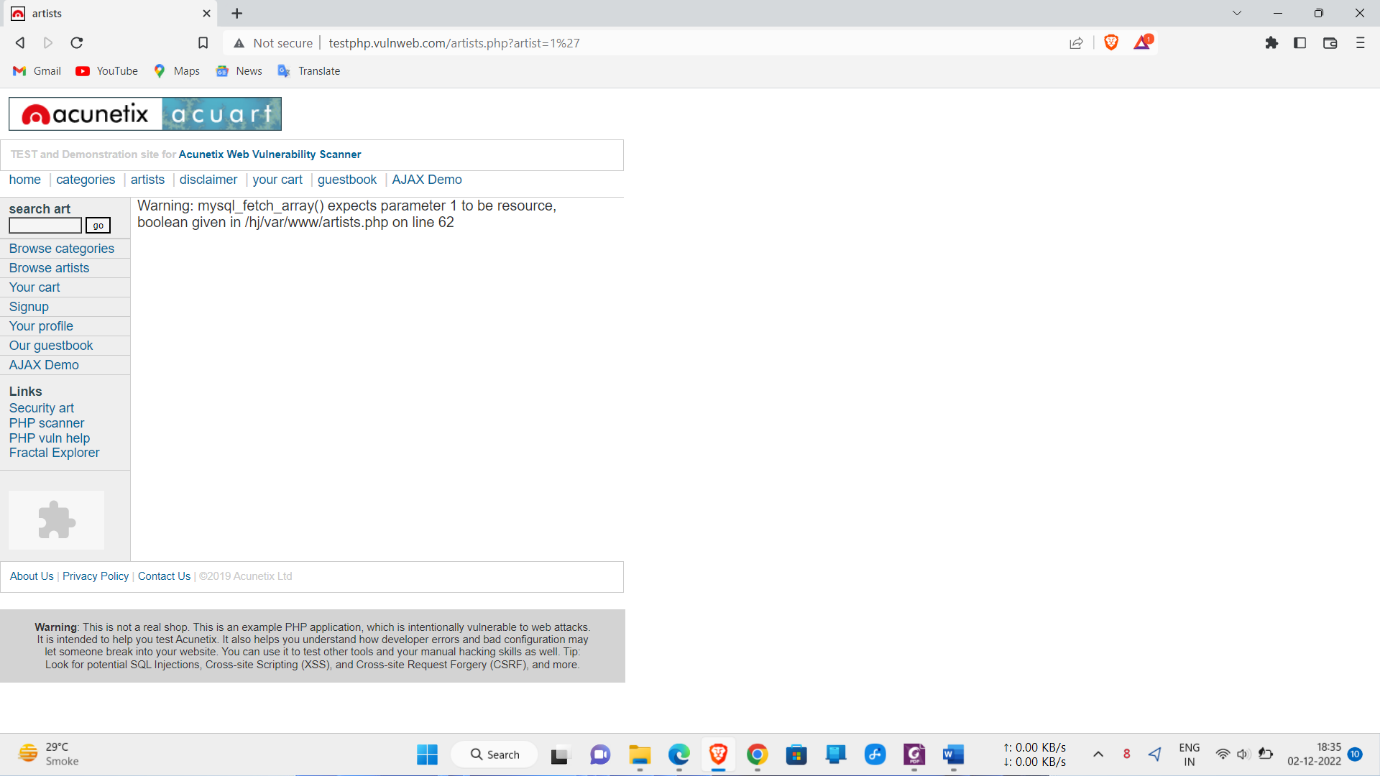
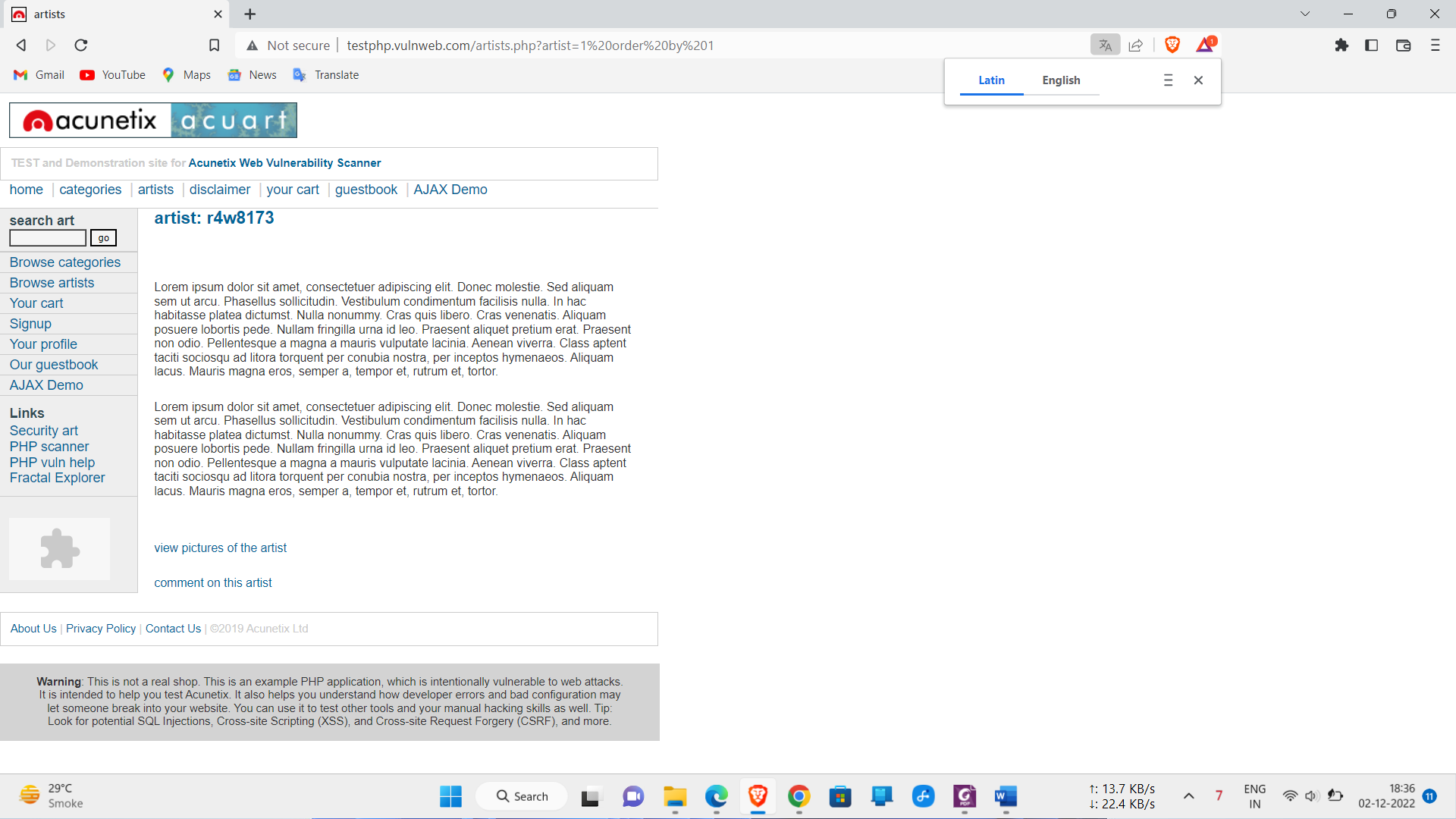
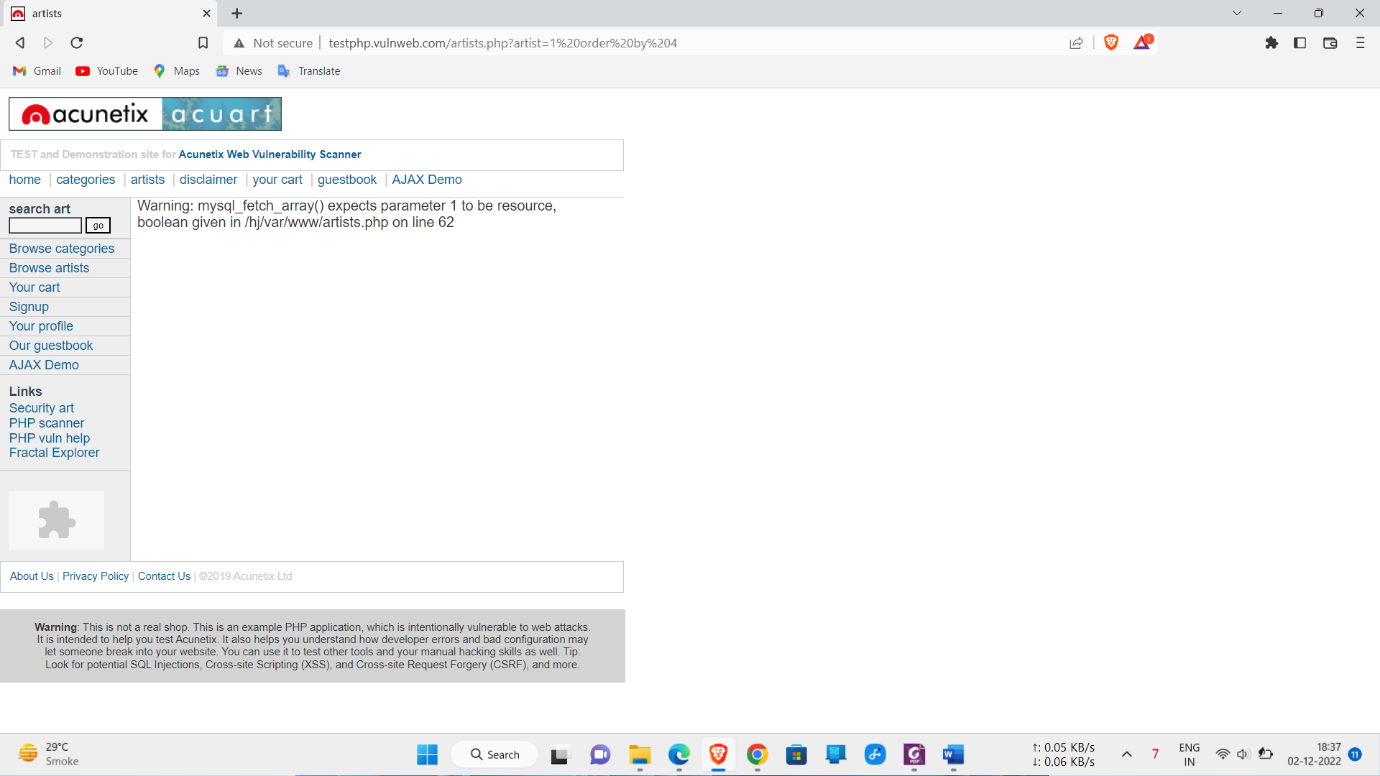
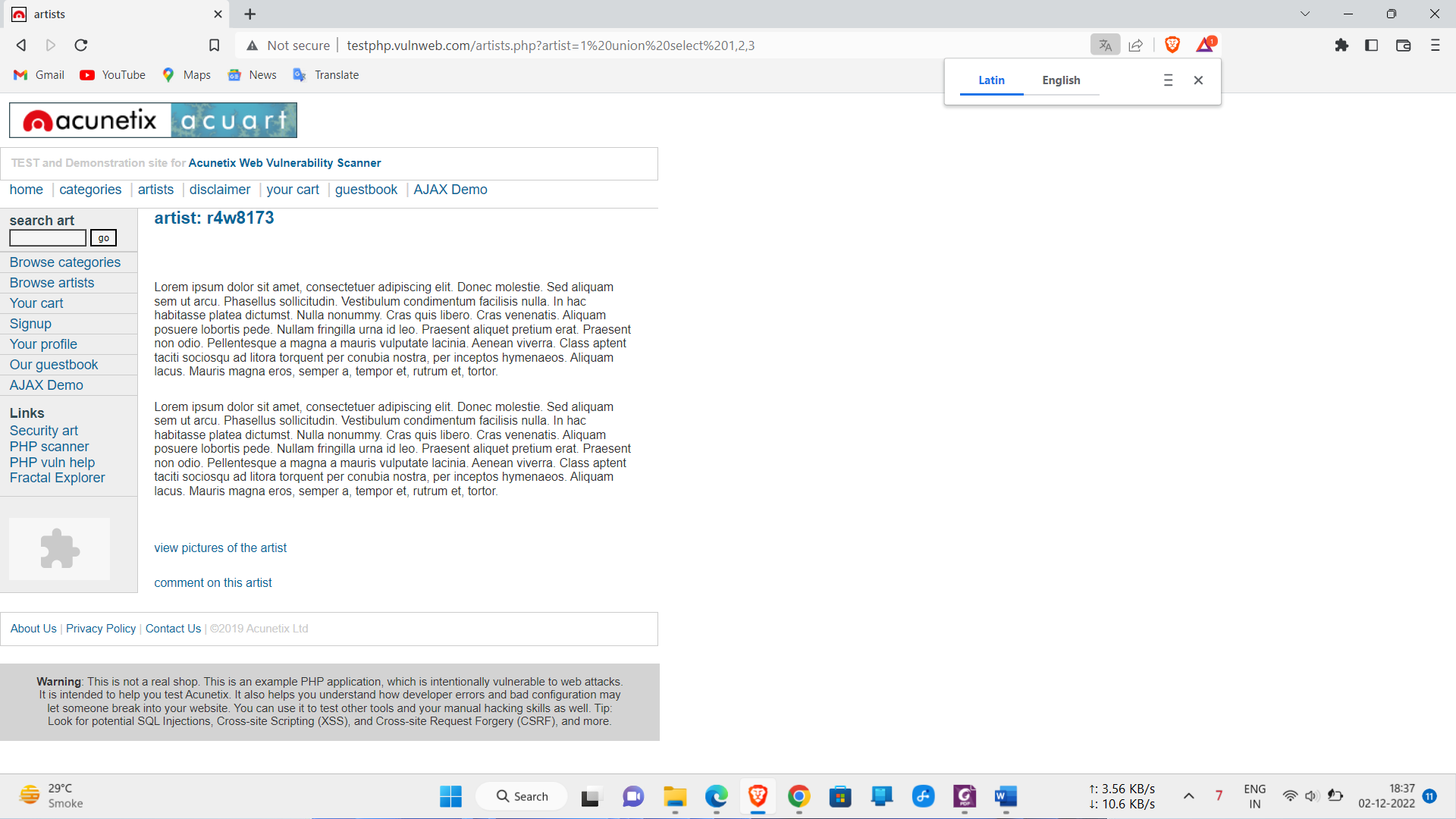
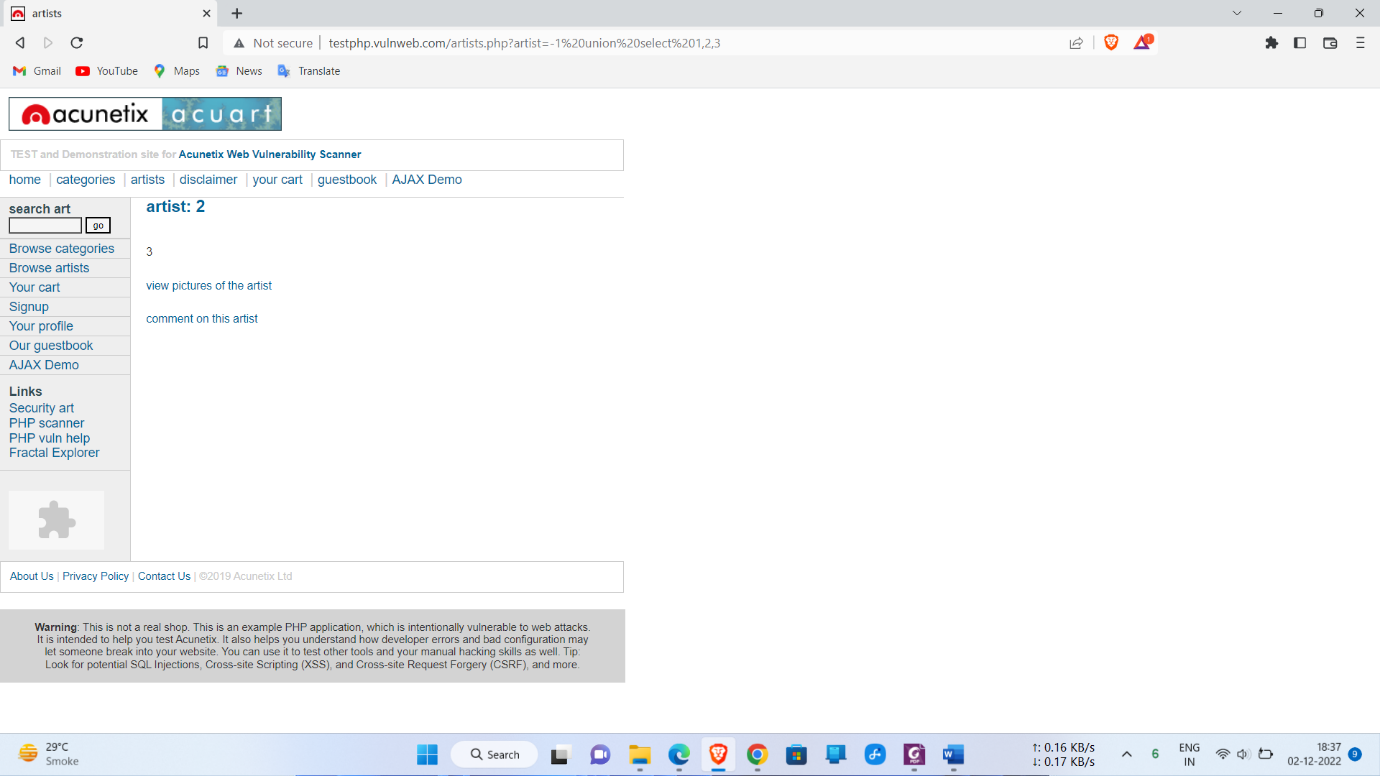
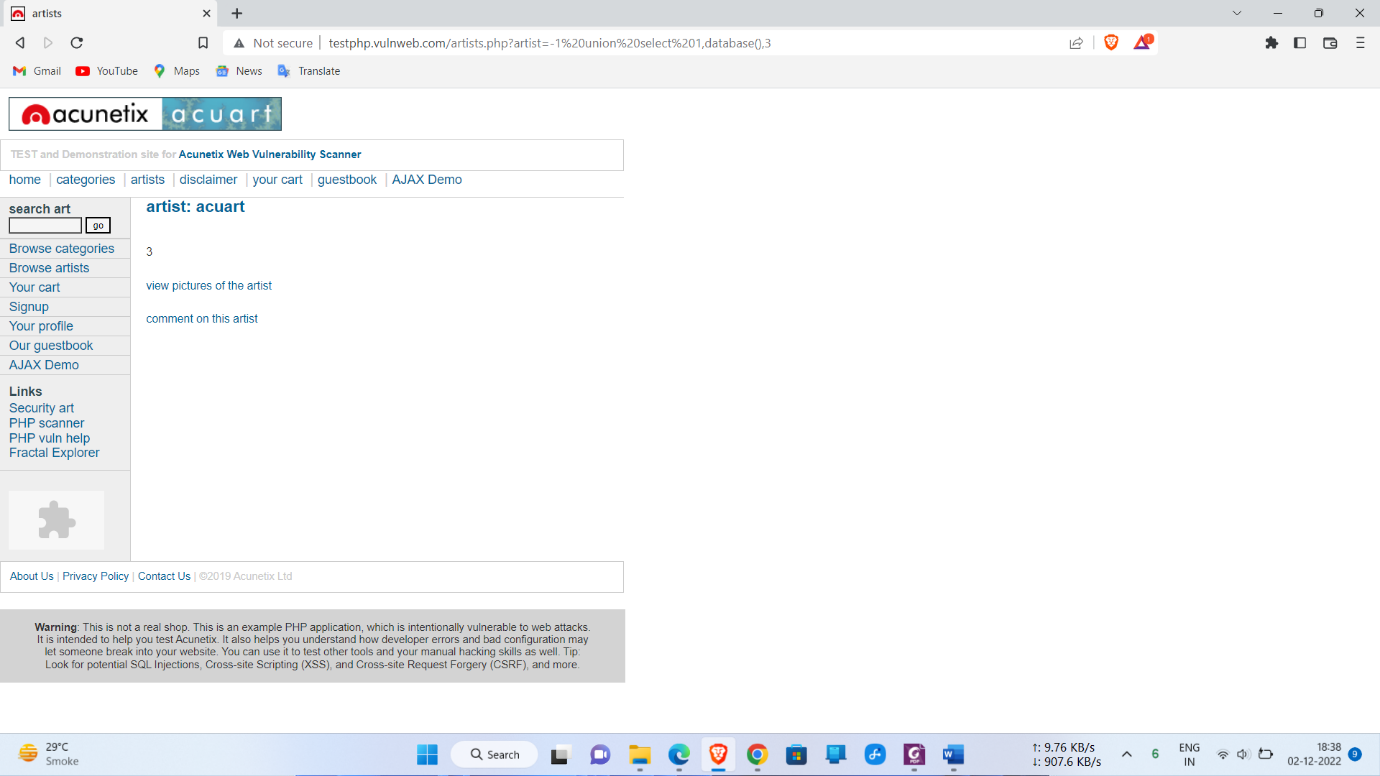
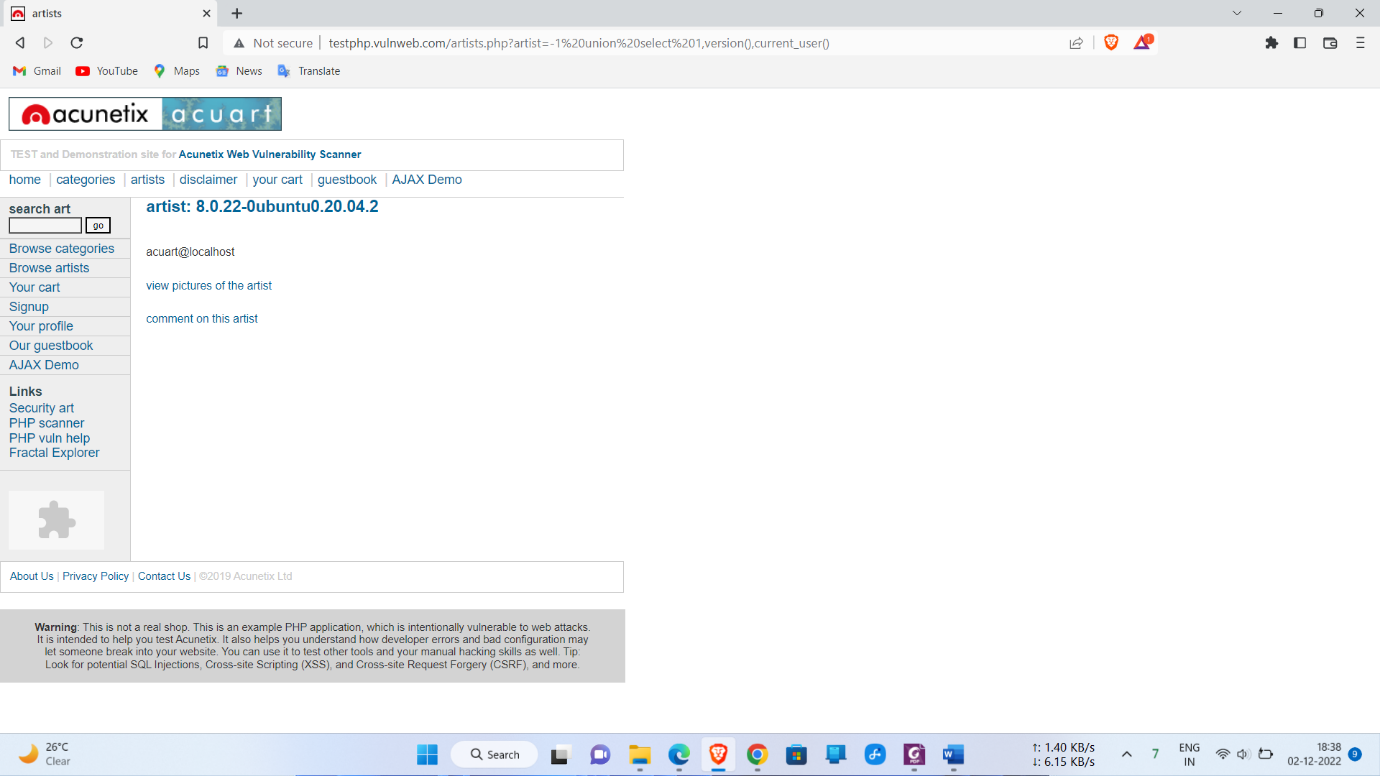
&

4. Install Social Phish tool from GitHub and try to execute the tool for phishing page and perfrom in  
lab setup only





5 Perform SQL injection Manually on http://testphp.vulnweb.com Write a report along with  
screenshots and mention preventive steps to avoid SQL injections



**Preventive steps to avoid SQL injections**

Preventing or mitigating SQL injection attacks is a lot about ensuring that none of the fields are vulnerable to invalid inputs and application execution. yours is manually impossible to actually to check every page and every application on the website, especially when updates are frequent and user-friendliness is the top priority. Nonetheless, security analysts and seasoned developers recommend a number of the subsequent points guarantee your database square measure well protected inside the confinement of the server.

**1) Continuous Scanning and Penetration Testing**

The automated [**web application scanner**](https://www.indusface.com/web-application-scanning.php/)has been the best choice to point out vulnerabilities within the web applications for quite some time now. Now, with SQL injections getting smarter in exploiting logical flaws, website security professionals should explore manual testing with the help of a security vendor.

They can authenticate user inputs against a set of rules for syntax, type, and length. It helps to audit application vulnerabilities discreetly so that you can patch the code before hackers exploit it to their advantage.

**2) Restrict Privileges**

It is more of a database management function, but enforcing specific privileges to specific accounts helps prevent blind SQL injection attacks. Begin with no privileges account and move on to ‘read-only’, ‘edit’, ‘delete’ and similar privilege levels.

Minimizing privileges to the application will ensure that the attacker, who gets into the database through the application, cannot make unauthorized use of specific data.

**3) Use Query Parameters**

Dynamic queries create a lot of troubles for security professionals. They have to deal with variable vulnerabilities in each application, which only gets graver with updates and changes. It is recommended that you prepare parameterized queries.

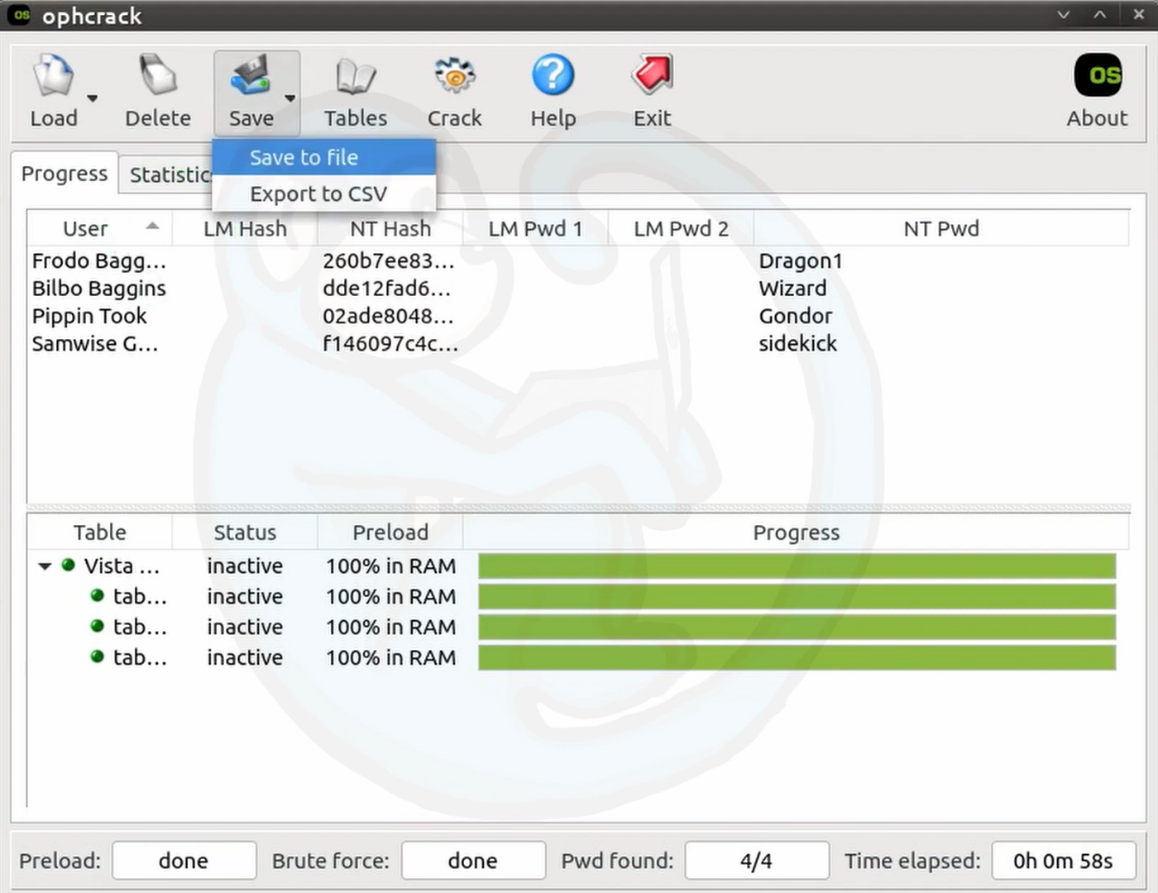
These queries are simple, easy to write, and only pass when each parameter in SQL code is clearly defined. This way, your info is supplied with weapons to differentiate between code and information inputs.

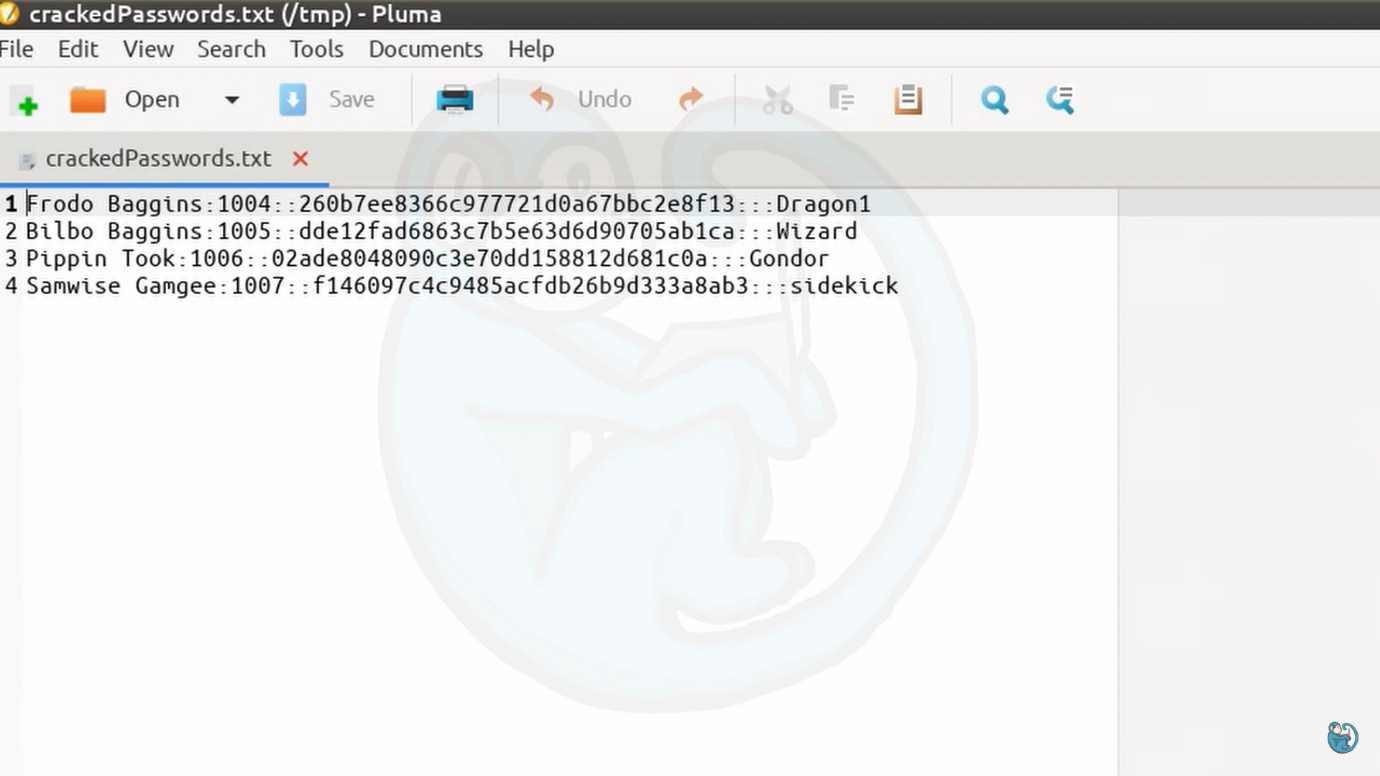
**4) Instant Protection**

A majority of organizations fail the problems like outdated code, scarcity of resources to test and make changes, no knowledge of application security, and frequent updates in the application. For these, web application protection is the best solution.

A managed [**web application firewal**l](https://www.indusface.com/web-application-firewall.php) can be deployed for immediate mitigation of such attacks. It contains custom policies to block any suspicious input and deny information breach instantly. This way, you do not have to manually look for loopholes and mend problems afterward.

6. Crack the password of windows machine by using ophcrack tool in virtual machine on windows 7  
and try get the password, along with that mention the path of SAM file in windows and and explain  
about SAM file usage and how it can be cracked by tool





**The SAM database file is stored within C:\Windows\System32\config. All of the data within the file is encrypted. The passwords hashes are stored in HKEY\_LOCAL\_MACHINE\SAM.**

### ****Introduction to SAM****

SAM is short for the Security Account Manager which manages all the user accounts and their passwords. It acts as a database. All the passwords are hashed and then stored SAM. It is the responsibility of LSA (Local Security Authority) to verify user login by matching the passwords with the database maintained in SAM. SAM starts running in the background as soon as the Windows boots up. SAM is found in **C:\Windows\System32\config** and passwords that are hashed and saved in SAM can found in the registry, just open the Registry Editor and navigate yourself to **HKEY\_LOCAL\_MACHINE\SAM**.

### ****How are Passwords stored in Windows?****

To know how passwords are saved in windows, we will first need to understand what are LM, NTLM v1 & v2, Kerberos.

### ****LM authentication****

LAN Manager (LM) authentication was developed by IBM for Microsoft’s Windows Operating Systems. The security it provides is considered hackable today. It converts your password into a hash by breaking it into two chunks of seven characters each. And then further encrypting each chunk. It is not case sensitive either, which is a huge drawback. This method coverts the whole password string into uppercase, so when the attacker is applying any attack like brute force or dictionary; they can altogether avoid the possibility of lowercase. The key it is using to encrypt is 56-bit DES which now can be easily cracked.

### ****NTLM authentication****

NTLM authentication was developed to secure the systems as LM proved to be insecure at the time. NTLM’s base is a challenge-response mechanism. It uses three components – nonce (challenge), response and authentication.

When any password is stored in Windows, NTLM starts working by encrypting the password and storing the hash of the said password while it disposes of the actual password. And it further sends the username to the server, then the server creates a 16-byte random numeric string, namely nonce and sends it to the client. Now, the client will encrypt the nonce using the hash string of the password and send the result back to the server. This process is called a response. These three components (nonce, username, and response) will be sent to Domain Controller. The Domain Controller will recover the password using hash from the Security Account Manager (SAM) database. Furthermore, the domain controller will check the nonce and response in case they match, Authentication turns out to be successful.

Working of NTLM v1 and NTML v2 is the same, although there are few differences such as NTML v1 is MD4 and v2 is MD5 and in v1 C/R Length is 56 bits + 56-bit +16 bit while v2 uses 128 bits. When it comes to C/R Algorithm v1 uses DES (ECB mode) and v2 is HMAC\_MD5. and lastly, in v1 C/R Value Length 64 bit + 64 bit + 64 bit and v2 uses 128 bits.

Now as we have understood these hashing systems, let’s focus on how to dump them. The methods we will focus on are best suited for both internal and external pen-testing. Let’s begin!

**NOTE:** Microsoft changed the algorithm on Windows 10 v1607 which replaced the RC4 cipher with AES. This change made all the extraction tools that directly access SAM to dump hashes obsolete. Some of the tools have been updated and handle the new encryption method properly. But others were not able to keep up. This doesn’t mean that they cannot be used anymore. This just means that if we face the latest Windows 10, we rather use update tools. Hence we divided this article into 2 parts. Windows 7 and Windows 10.

7.Write an Article on cybersecurity and recent attacks which you came across in media and news and  
research on that news, and explain the any topic which you learned in this course and mention  
what you learned

To achieve those goals of gaining access or disabling operations, a number of different technical methods are deployed by cybercriminals. There are always new methods proliferating, and some of these categories overlap, but these are the terms that you're most likely to hear discussed.

1. Malware
2. Phishing
3. Ransomware
4. Denial of service
5. Man in the middle
6. Cryptojacking
7. SQL injection
8. Zero-day exploits

[**Malware**](https://www.csoonline.com/article/3295877/malware/what-is-malware-viruses-worms-trojans-and-beyond.html) — Short for malicious software, malware can refer to any kind of software, no matter how it's structured or operated, that "is a designed to cause damage to a single computer, server, or computer network," [as Microsoft puts it](https://go.skimresources.com/?id=111346X1569475&xs=1&isjs=1&url=https%3A%2F%2Ftechnet.microsoft.com%2Fen-us%2Flibrary%2Fdd632948.aspx&xguid=cf93e6f57a51c6ac5a825a22c1a95405&xuuid=84d75aa68a6b168c08bcaea9a69152d9&xsessid=&xcreo=0&xed=0&sref=https%3A%2F%2Fwww.csoonline.com%2Farticle%2F3295877%2Fmalware%2Fwhat-is-malware-viruses-worms-trojans-and-beyond.html&pref=https%3A%2F%2Fwww.csoonline.com%2Farticle%2F3237324%2Fcyber-attacks-espionage%2Fwhat-is-a-cyber-attack-recent-examples-show-disturbing-trends.html&xtz=480&jv=13.12.1-stackpath&bv=2.5.1). Worms, viruses, and trojans are all varieties of malware, distinguished from one another by the means by which they reproduce and spread. These attacks may render the computer or network inoperable, or grant the attacker root access so they can control the system remotely.

[**Phishing**](https://www.csoonline.com/article/2117843/phishing/what-is-phishing-inside-this-effective-and-evolving-cyber-attack.html) — Phishing is a technique by which cybercriminals craft emails to fool a target into taking some harmful action. The recipient might be tricked into downloading malware that's disguised as an important document, for instance, or urged to click on a link that takes them to a fake website where they'll be asked for sensitive information like bank usernames and passwords. Many phishing emails are relatively crude and emailed to thousands of potential victims, but some are specifically crafted for valuable target individuals to try to get them to part with useful information.

[ Related: [15 real-world phishing examples — and how to recognize them](https://www.csoonline.com/article/3235520/phishing/15-real-world-phishing-examples-and-how-to-recognize-them.html) ]

[**Ransomware**](https://www.csoonline.com/article/3236183/what-is-ransomware-how-it-works-and-how-to-remove-it.html) — Ransomware is a form of malware that encrypts a victim's files. The attacker then demands a ransom from the victim to restore access to the data upon payment. Users are shown instructions for how to pay a fee to get the decryption key. The costs can range from a few hundred dollars to thousands, and are typically payable to cybercriminals in cyptocurrency.

[**Denial of service**](https://www.csoonline.com/article/3648530/ddos-attacks-definition-examples-and-techniques.html)— A denial of service attack is a brute force method to try stop some online service from working properly. For instance, attackers might send so much traffic to a website or so many requests to a database that it overwhelms those systems ability to function, making them unavailable to anybody. A [**distributed denial of service (DDoS)**](https://www.csoonline.com/article/3222095/ddos-explained-how-denial-of-service-attacks-are-evolving.html) attack uses an army of computers, usually compromised by malware and under the control of cybercriminals, to funnel the traffic towards the targets.

[**Man in the middle**](https://www.csoonline.com/article/3340117/what-is-a-man-in-the-middle-attack-how-mitm-attacks-work-and-how-to-prevent-them.html)— A man in the middle attack (MITM) is a method by which attackers manage to interpose themselves secretly between the user and a web service they're trying to access. For instance, an attacker might set up a Wi-Fi network with a login screen designed to mimic a hotel network; once a user logs in, the attacker can harvest any information that user sends, including banking passwords.

[**Cryptojacking**](https://www.csoonline.com/article/3253572/internet/what-is-cryptojacking-how-to-prevent-detect-and-recover-from-it.html)— Cryptojacking is a specialized attack that involves getting someone else's computer to do the work of generating cryptocurrency for you (a process called mining in crypto lingo). The attackers will either install malware on the victim's computer to perform the necessary calculations, or sometimes run the code in JavaScript that executes in the victim's browser.

[**SQL injection**](https://www.csoonline.com/article/3257429/application-security/what-is-sql-injection-this-oldie-but-goodie-can-make-your-web-applications-hurt.html)— SQL injection is a means by which an attacker can exploit a vulnerability to take control of a victim's database. Many databases are designed to obey commands written in the Structured Query Language (SQL), and many websites that take information from users send that data to SQL databases. In a SQL injection attack, a hacker will, for instance, write some SQL commands into a web form that's asking for name and address information; if the web site and database aren't programmed correctly, the database might try to execute those commands.

[**Zero-day exploits**](https://www.csoonline.com/article/3284084/cyber-attacks-espionage/what-is-a-zero-day-exploit-a-powerful-but-fragile-weapon.html)— Zero-days are vulnerabilities in software that have yet to be fixed. The name arises because once a patch is released, each day represents fewer and fewer computers open to attack as users download their security updates.  Techniques for exploiting such vulnerabilites are often bought and sold on the [dark web](https://www.csoonline.com/article/3249765/data-breach/what-is-the-dark-web-how-to-access-it-and-what-youll-find.html) — and are sometimes discovered by government agencies that controversially may use them for their own hacking purposes, rather than releasing information about them for the common benefit.

## Recent cyber attacks

Deciding which cyber attacks were the worst is, arguably, somewhat subjective. Those that made our list did so because they got a lot of notice for various reasons — because they were widespread, perhaps, or because they were signals of a larger, scary trend.

Without further ado, here are some of the most notable cyber attacks in recent history and what we can learn from them:

1. Capitol One breach
2. The Weather Channel ransomware
3. U.S. Customs and Border Protection/Perceptics
4. Citrix breach
5. Texas ransomware attacks
6. WannaCry
7. NotPetya
8. Ethereum
9. Equifax
10. Yahoo
11. GitHub

**Capitol One breach**

In July of 2019, online banking giant Capitol One realized that its data had been hacked. Hundreds of thousands of credit card applications, which included personally identifying information like birthdates and Social Security numbers, were exposed. No bank account numbers were stolen, but the sheer scale was extremely worrying. Things followed the usual script, with Capitol One making [shamefaced amends and offering credit monitoring](https://www.capitalone.com/facts2019/) to those affected.

But then things took a turn for the unusual. The stolen data never appeared on the [dark web](https://www.csoonline.com/article/3249765/what-is-the-dark-web-how-to-access-it-and-what-youll-find.html), nor did the hack look like a Chinese espionage operation like the [Equifax](https://www.csoonline.com/article/3444488/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html) and [Marriott](https://www.csoonline.com/article/3441220/marriott-data-breach-faq-how-did-it-happen-and-what-was-the-impact.html) breaches. In fact, the attack was perpetrated by an American named Paige Thompson, aka Erratic. Thompson had previously worked for Amazon, which gave her the background necessary to recognize that Capitol One's AWS server had been badly misconfigured in such a way to leave it quite vulnerable. It initially seemed that Thompson's theft of the data was in the tradition of [freelance white-hat hacking and security research](https://www.theverge.com/2019/7/31/20748886/capital-one-breach-hack-thompson-security-data): she made little attempt to hide what she was doing, never tried to profit from the data, and in fact was caught because she posted a list of Capitol One's breached directories — but no actual data — on her GitHub page. But attempts to understand her motivation in the wake of her arrest [were increasingly difficult](https://www.csoonline.com/article/3433244/capital-one-hack-shows-difficulty-of-defending-against-irrational-cybercriminals.html), and it's possible that she was, true to her chosen nickname, erratic, if not undergoing a serious mental health crisis.

**The Weather Channel ransomware**

The Weather Channel may not seem like a crucial piece of infrastructure, but for many people it's a lifeline — and in April 2019, during a stretch of tornado strikes across the American south, many people were tuning in. But one Thursday morning the channel ceased live broadcasting for [nearly 90 minutes](https://www.theverge.com/2019/4/19/18507869/weather-channel-ransomware-attack-tv-program-cable-off-the-air), something almost unheard of in the world of broadcast television.

It turns out The Weather Channel had fallen victim to a ransomware attack, and while there's been no confirmation of the attack vector, [rumors are that it was via phishing attack](https://www.cybersecurity-insiders.com/ransomware-attack-disrupts-the-weather-channel-for-90-minutes/), one of the most common causes of [ransomware](https://www.csoonline.com/article/3236183/what-is-ransomware-how-it-works-and-how-to-remove-it.html) infection. The attack demonstrated that the boundary between "television" and "the internet" has more or less been erased, as any TV operation like The Weather Channel would be entirely reliant on internet-based services to operate. It also demonstrated one way to beat ransomware. The Weather Channel didn't fork over any bitcoin; rather, they had good backups of the affected servers and were able to get back online in less than two hours.

**U.S. Customs and Border Protection/Perceptics**

The sequence was sadly not that unusual: a hacker breaches a company's servers, gets access to sensitive data, and then demands a ransom. When the executives fail to pay up, the material begins to find its way to the dark web for sale, where the scope of its importance become recognized.

The data turned out to be very important indeed: it was [stolen from the U.S. Customs and Border Protection agency](https://www.washingtonpost.com/technology/2019/06/10/us-customs-border-protection-says-photos-travelers-into-out-country-were-recently-taken-data-breach/) (CBP), and the irony that the agency dedicated to protecting the U.S. borders couldn't protect its own data wasn't lost on anyone. In fact, much of the blame lay on Perceptics, a contractor that provides all the license plate scanners for the border agency, as well as to a host of other U.S. and Canadian government departments. The stolen photos of [cars and drivers](https://www.vice.com/en_us/article/43j5wm/here-are-images-of-drivers-hacked-from-a-us-border-protection-contractor-on-the-dark-web-perceptics) had actually been copied from CBP's computers to Perceptics' own servers, in violation of government policy; Perceptics was then hacked, and the data publicized by the attacker "[Boris Bullet-Dodger](https://www.theregister.co.uk/2019/05/23/perceptics_hacked_license_plate_recognition/)" when ransom negotiations with execs broke down. The case brought up questions about government-contractor relations and the wisdom of allowing the collection of biometric data. While Perceptics' relationship with CBP was suspended in the wake of the attack, the government eventually agreed to [keep doing business with the company](https://www.washingtonpost.com/technology/2019/10/10/surveillance-contractor-that-violated-rules-by-copying-traveler-images-license-plates-can-continue-work-with-cbp/).

**Citrix breach**

When an organization being breached is itself in the cybersecurity business, that's enough to make everyone nervous — but it's also a cautionary tale about how even security vendors can have a hard time establishing a security mindset internally.

Take Citrix, for example. The company makes VPNs, which help secure millions of internet connections, and has extensive dealings with the U.S. government. But it still fell victim to a ["password spraying" attack in March of 2019](https://www.cyberscoop.com/hackers-used-password-spraying-breach-citrix-investigation-confirms/) — essentially, an attack where a hacker attempts to gain access to a system via brute force, by rapidly attempting to login with simple and frequently used passwords (think "password123" and the like). In all likelihood, the attack came from a group associated with the [Iranian government](https://www.cpomagazine.com/cyber-security/massive-citrix-data-breach-thought-to-be-the-work-of-iranian-hackers/). Fortunately, the attackers didn't get very far into Citrix's systems — but the company still promised a revamp of its internal security culture.

**Texas ransomware attacks**

In August of 2019, computer systems in 22 small Texas towns were [rendered useless by ransomware](https://www.nytimes.com/2019/08/20/us/texas-ransomware.html), leaving their governments unable to provide basic services like issuing birth or death certificates. How did a single attacker, using the [REvil/Sodinokibi ransomware](https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/texas-municipalities-hit-by-revil-sodinokibi-paid-no-ransom-over-half-resume-operations), manage to hit so many different towns? There was a single point of weakness: an IT vendor who provided services to all of these municipalities, all of which were too small to support a full-time IT staff.

But if that sort of collective action opened a weakness, there was a power in collaboration as well. Rather than giving in and paying the [$2.5 million ransom demanded](https://www.npr.org/2019/08/20/752695554/23-texas-towns-hit-with-ransomware-attack-in-new-front-of-cyberassault), the towns teamed up with the Texas state government's Department of Information Resources. The agency led a [remediation effort](https://dir.texas.gov/View-About-DIR/Article-Detail.aspx?id=213) that had the cities back on their feet within weeks, in contrast with places like Baltimore, where [systems were offline for months](https://en.wikipedia.org/wiki/2019_Baltimore_ransomware_attack).

**WannaCry**

[WannaCry](https://www.csoonline.com/article/3227906/wannacry-explained-a-perfect-ransomware-storm.html) was a ransomware attack that spread rapidly in May of 2017. Like all ransomware, it took over infected computers and encrypted the contents of their hard drives, then demanded a payment in Bitcoin in order to decrypt them. The malware took particular root in computers at facilities run by the United Kingdom's NHS.

Malware isn't anything new, though. What made WannaCry significant and scary was the means it used to propagate: it exploited a vulnerability in Microsoft Windows using code that had been secretly developed by the United States National Security Agency. Called EternalBlue, the exploit had been [stolen and leaked by a hacking group called the Shadow Brokers](https://medium.com/@shadowbrokerss/dont-forget-your-base-867d304a94b1). Microsoft had already patched the vulnerability a few weeks before, but many systems hadn't upgraded. Microsoft was furious that the U.S. government had built a weapon to exploit the vulnerability rather than share information about the hole with the infosec community.

**NotPetya**

Petya was just another piece of ransomware when it started circulating via phishing spam in 2016; its main claim to fame was that it encrypted the master boot record of infected machines, making it devilishly difficult for users to get access to their files.

Then, abruptly in June of 2017, a much [more virulent version of the malware started spreading](https://www.csoonline.com/article/3204148/security/notpetya-and-shadow-brokers-july-vip-service-mystery-gift-dump-of-the-month-club.html). It was different enough from the original that it was dubbed *[NotPetya](https://www.csoonline.com/article/3233210/ransomware/petya-ransomware-and-notpetya-malware-what-you-need-to-know-now.html)*; it originally propagated via compromised Ukrainian accounting software and spread via the same EternalBlue exploit that WannaCry used. NotPetya is widely believed to be a cyberattack from Russia against Ukraine, though Russia denies it, opening up a possible era of states using weaponized malware.

**Ethereum**

While this one might not have been as high-profile as some of the others on this list, it deserves a spot here due to the sheer amount of money involved. Ether is a Bitcoin-style cryptocurrency, and $7.4 million in Ether was [stolen from the Ethereum app platform in a manner of minutes in July](https://www.csoonline.com/article/3209130/security/hacker-allegedly-stole-7-4-million-worth-of-ether-in-three-minutes.html). Then, just weeks later came [a $32 million heist](http://www.businessinsider.com/report-hackers-stole-32-million-in-ethereum-after-a-parity-breach-2017-7). The whole incident raised questions about the security of blockchain-based currencies.

In this course I learned Malware, Phishing , Ransomware and SQL injection attacks