SSI
(Computer systems security)

Common Vulnerabilities and Exposures (CVE)

Exercise sheet 1

João Azevedo

University of Minho, Braga, PT
a85227@alunos.uminho.pt

Abstract. Esta ficha de exercício tem por objectivo principal apresentar a identificação padrão de vulnerabilidades e exposições publicamente conhecidas, assim como a sua importância nas atividades relacionadas com a segurança de sistemas informáticos. Espera-se, com este trabalho, promover o conhecimento de ferramentas de apoio a ações proativas de segurança.

Keywords: Security · Vulnerabilities · Exploits · CVE · CVSS

1 Respostas às questões colocadas

Q1: *"Escolha três aplicações tipicamente usadas em seu computador pessoal, pesquise pela existência de vulnerabilidades conhecidas e meios de explorá-las. Descreva detalhadamente as suas descobertas, incluindo as imagens de suas pesquisas e a descrição das informações nelas contidas."*

R: Todas as aplicações escolhidas e versões respetivas foram obtidas a partir do sistema operativo *Linux Ubuntu 20.04.1 LTS*:

1. **Node.js** (*v10.19.0*)
   - **Descrição**: *Software open-source*, *cross-platform*, e um *runtime* de *JavaScript* para execução de código *frontend* e/ou *backend*.
   - **Vulnerabilidades** (*CVEs*):
     - *CVE-2020-8251*: Para versões inferiores à v14.11.0, estando a minha incluída, o Node.js era vulnerável a ataques de negação de serviço (DoS) em atrasos no envio de pedidos (*requests*) que poderiam tornar servidores incapazes de aceitar novas conexões.

– **Análise de gravidade** (*CVSS **v3.1***):

O *standard CVSS* para esta vulnerabilidade[1], obtido através do **NIST**(1) mostra-nos os grupos de métricas que podem ser consultadas na imagem seguinte:

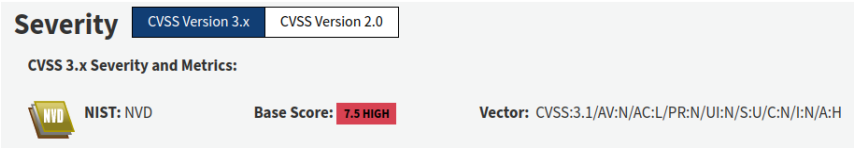

Fig. 1. Análise das métricas de gravidade para o *CVSS v3.1*.

Temos então um *score* de 7.5 (*high*) cujo vetor de ataque é a rede e a complexidade do mesmo é baixa, sem serem necessários quaisquer privilégios nem interação com o utilizador. Não compromete a integridade nem confidencialidade dos dados mas tem como consequência principal diminuir a disponibilidade do serviço para utilizadores legítimos.

Fazendo uma análise estatística das vulnerabilidades expostas para esta aplicação podemos ver que a maioria (**42%**) pertence a ataques de negação de serviço (**DoS**):

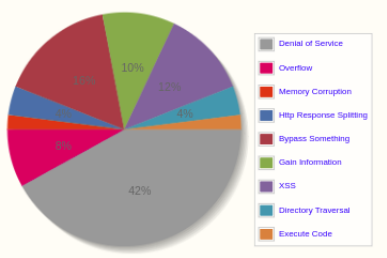

Fig. 2. Gráfico circular de vulnerabilidades para o Node.js.

– ***Exploits***: Não existem nenhuns *exploits* disponíveis para esta vulnerabilidade.

[1] Publicado a 18 de Setembro de 2020: https://nvd.nist.gov/vuln/detail/CVE-2020-8251 (National Institute of Standards and Technology)

2. **Skype Technologies** (*v8.65.0.78*)
   - **Descrição**: Aplicação especializada em fornecer serviços de chamadas de vídeo e áudio entre dispositivos.
   - **Vulnerabilidades** (*CVEs*):

     Devo salientar que a pesquisa filtrada para o Skype especificamente em sistemas Linux levou a muito poucos resultados, no entanto o caso seguinte, remetendo a **2004**, é interessante na medida em que introduz um novo tipo de vulnerabilidade.

     - *CVE-2004-1778*: O Skype para Linux, nas versões 0.92.0.12 e 1.0.0.1 e possivelmente outras, permitia a todos os tipos de utilizadores alterarem uma diretoria da aplicação que permitia modificar ficheiros de linguagens, manuais e outras informações podendo introduzir ataques de Engenharia Social e levar o *end user* a tomar decisões manipuladas pelo atacante.
   - **Análise de gravidade** (*CVSS **v2.0***):
     O *standard CVSS* disponível para esta vulnerabilidade[2] apenas está descrito para a versão 2.0, obtido através do **CVE details**(2), mostrando-nos os grupos de métricas que podem ser consultadas na imagem seguinte:

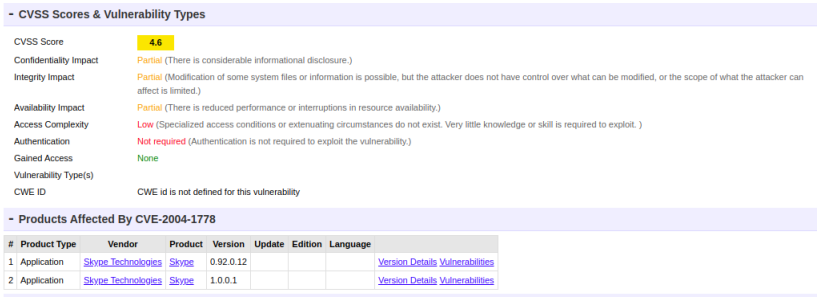

Fig. 3. Análise das métricas de gravidade para o *CVSS v2.0*.

Temos então um *score* de 4.6 (*medium*) cujo foco principal desta vulnerabilidade incide na consistência e integridade dos dados que não é inteiramente garantida. O acesso a este método é simples e não exige nenhum tipo de privilégio, visto que a informação pode ser acedida/modificada por todos.

- ***Exploits***: A alteração das permissões resolve esta vulnerabilidade pelo que não existem *exploits* e a versão atual é a 8.x.x.x.

[2] Publicado a 22 de Dezembro de 2004: https://www.cvedetails.com/cve-details.php?t=1&cve_id=CVE-2004-1778

3. **Zoom** (*v5.2.458699.0906*)

   – **Descrição**: Aplicação especializada em fornecer serviços de chamadas de vídeo e áudio entre dispositivos.

   – **Vulnerabilidades** (*CVEs*):

   Desta vez a pesquisa seguiu o caminho inverso, i.e., procurar por um possível *exploit* e analisar a vulnerabilidade (*CVE ID*) correspondente. Desta vez temos um novo tipo de ataque que comprometeu o funcionamento do sistema *Zoom* em versões mais antigas, através de um *Stack-based Buffer Overflow*.

   - *CVE-2017-15048*: No *Launcher* do *Zoom*, antes da versão 2.0.115900.120, existia um crescimento descontrolado de um *input* do utilizador, i.e., não existia verificação do tamanho do *buffer* de destino para guardar os dados, o que permitiria ao atacante executar o código que desejasse.

   – **Análise de gravidade** (*CVSS* ***v3.x***):
   O *standard CVSS* disponível para esta vulnerabilidade, obtido através do **NIST** [3], mostra os grupos de métricas que podem ser consultadas na imagem seguinte:

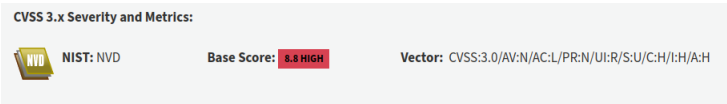

Fig. 4. Análise das métricas de gravidade para o *CVSS v3.0*.

Temos então um *score* de 8.8 (*high*), o mais elevado destes exemplos que mostrei. O atacante usaria portanto a rede para o ataque sem muita complexidade e sem necessidade de alguns privilégios adicionais. Aqui como o ataque é produzido através da inserção de comandos na *shell*, o *User interaction* é necessário mas sem prejudicar qualquer outra aplicação para além do *scope* principal. Neste caso temos um impacto elevado na confidencialidade, integridade e disponibilidade, porque a partir do momento que o atacante executa código próprio tudo está comprometido.

[3] Publicado a 19 de Dezembro de 2017: https://nvd.nist.gov/vuln/detail/CVE-2017-15048

– ***Exploits***: Como referi acima, existe sim um *exploit*(3), no entanto, o erro já foi resolvido para versões subsequentes, existe inclusivé um relatório de comunicações entre o autor do *exploit* e a empresa em questão, levando a um *update* para a v2.0.115900.1201.
Para efeitos ilustrativos, na imagem seguinte temos um excerto dos passos a realizar para reproduzir o dado *exploit*[4] (autor: *conviso*):

Fig. 5. Excerto dos comandos para reproduzir o *exploit*.

Q2: *"Em 2014 foi descoberta uma falha de programação na biblioteca de criptografia open source OpenSSL que ficou publicamente conhecida como Heartbleed. Esta falha foi identificada com CVE-2014-0160. Use esta identificação para descrever detalhadamente esta falha, incluindo (mas não apenas) as versões afetadas, os eventuais exploits existentes, vectores de ataque, impacto e soluções. Use as imagens de suas consultas e outros recursos utilizados para justificar suas conclusões."*

R: Recorrendo, por exemplo, à base de dados do *NIST* utilizado anteriormente, obtemos uma série de informações associadas a essa vulnerabilidade:

– ***ID da Vulnerabilidade:*** *CVE-2014-0160*

- Versões afetadas: v1.0.1 até v1.0.1f (resolvido na versão 1.0.1g em **2014**);

- Descrição: Os protocolos de segurança **TLS** (*Transport Layer Security*) e **DTLS** (*Datagram TLS*) implementados no *ToolKit OpenSSL* não estavam desenhados para suportar de forma correta a extensão a pacotes do tipo *Heartbeat*(4).

[4] Exploit DB, sources and description: https://www.exploit-db.com/exploits/43355

Para vias de contextualização, a extensão de pacotes *Heartbeat* fornecia um novo protocolo *TLS/DTLS* que permitia a utilização da *flag* **keep-alive** sem ter de renegociar a ligação (e outras funcionadidades presentes no *link* referenciado anteriomente), *flag* essa usada para verificar se o destino ainda estava *online*. A imagem seguinte ilustra o processo envolvido no ataque[5]:

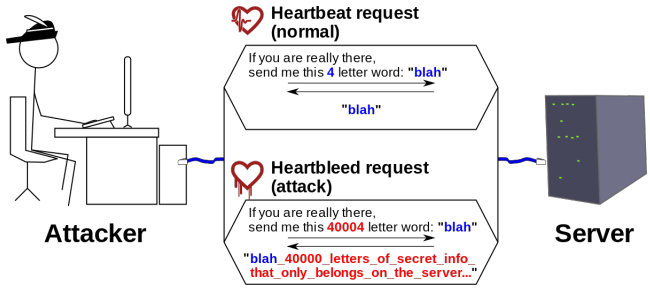

Fig. 6. Representação ilustrativa do ataque.

A utilização desta versão, com o conhecido *"The Heartbleed Bug"*[6], permitia a alguém na rede ler a memória de sistemas protegidos por *software* que usava *OpenSSL* para comunicações do tipo cliente-servidor. Os dados comprometidos envolviam chaves secretas, nomes e *passwords* de utilizadores e o próprio conteúdo enviado, obtidos não através das comunicações TLS/DTLS mas sim nas verificações paralelas de que o destino ainda estava *online*, permitindo ao atacante acrescentar informação no *Hearbeat request* que expunha dados da memória.

- Análise de gravidade (*CVSS* ***v3.1***):

  O*standard CVSS* para esta vulnerabilidade, obtido através do **NIST** [7] mostra-nos os grupos de métricas que podem ser consultadas na imagem seguinte:

[5] Fonte do diagrama: https://commons.wikimedia.org/wiki/File:Heartbleed_bug_explained.svg
[6] *"The Heartbleed Bug"*: https://heartbleed.com/
[7] Publicado a 7 de Abril de 2020: https://nvd.nist.gov/vuln/detail/CVE-2014-0160 (National Institute of Standards and Technology)

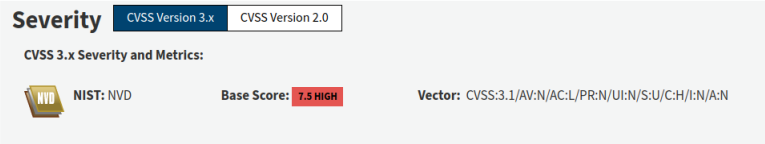

Fig. 7. Análise das métricas de gravidade para o *CVSS v3.1*.

O *score* base para esta vulnerabilidade é elevado (7.5) na escala considerada e o vetor de ataque é portanto a rede, a complexidade do mesmo é baixa, não sendo necessários quaisquer privilégios para o envio das comunicações mal intencionadas e, deste modo, a interação nem envolve qualquer tipo de *UI*.
Como o atacante obtém informações confidenciais (que à partida só deveriam ser acedidas por utilizadores legítimos) não encriptadas, a confidencialidade têm um impacto negativo elevado. Não existindo nenhuma alteração aos dados nem negação de serviço, estas duas últimas métricas não são afetadas.

- ***Exploits***: Existem uma série de *exploits* disponíveis na base de dados *ExploitDB*, segue uma lista disponível na página web desse sistema:

Fig. 8. *Exploits* disponíveis em *exploit-db.com*.

Por fim, as soluções existentes para este problema podem passar por atualizar o *software* para a versão 1.0.1g ou então, caso isso não seja possível, recompilar o *software OpenSSL* com a opção *"-DOPENSSL_NO_HEARTBEATS"*.

Q3: *"Assim como diversas corporações, a Mozilla Foundation divulga informações sobre vulnerabilidades as quais os seus produtos foram expostos através do seu Security Advisories. Em 02 de setembro de 2020, a companhia disponibilizou uma atualização do seu browser, i.e., Firefox for Android 80. Esta versão*

resolve uma série de vulnerabilidades listadas no relatório MFSA 2020-39(5). Descreva detalhadamente três vulnerabilidades listadas neste relatório."

R: O relatório disponibilizado pela *Mozilla Foundation* informa-nos da correção de 7 vulnerabilidades para o produto **Firefox for Android 80**, entre as quais:

1. **CVE-2020-15671**[8] (reportado por *Karol Frejlich*):
   - Descrição: Em certas condições, quando se inseria uma *password* utilizando o *Firefox* para *Android*, uma *race condition* ocorria quando o *InputContext*, classe que controlava o *scope* de *input*, não era corretamente colocado para um campo de *input* especial e, assim, a *password* era interpretada como uma *String* normal sendo guardada no dicionário do *keyboard*.
   - Métricas: Na escala *CVSS 3.1* o *score* é baixo (3.1), muito porque a *race condition* é mais difícil de acontecer, daí uma complexidade alta de ataque. Não são necessários privilégios mas sim interação do utilizador. As *passwords* ficando guardadas no dicionário do teclado existindo assim algum impacto na confidencialidade mas o resto mantém-se intacto.
2. **CVE-2020-15664**[9] (reportado por *Kaizer Soze*):
   - Descrição: Quando o *browser* utiliza a sua página *default - about:blank* (p. exe. quando está a carrega uma outra página), o atacante conseguia guardar uma referência da função *Javascript* **eval(String)** (execução de scripts js em formato *String*) para, através de uma página mal intencionada, ter acesso ao objeto **InstallTrigger** usado para ativar o *prompt* de uma extensão/*add-on* que, combinados com a confusão do utilizador (provavelmente um tipo de engenharia social) poderiam levar à instalação de algo malicioso.
   - Métricas: Na escala *CVSS 3.1* o *score* é médio (6.5). Através da rede, sem muita complexidade e, inclusivé, sem privilégios alguns, o atacante poderia levar o utilizador a interagir com a página através do tal *prompt* comprometendo possivelmente a integridade da informação a que ele passaria a ter acesso e consequente confiança da comunicação.
3. **CVE-2020-12400**[10] (*ver lista de reporters no url original*):
   - Descrição: Na conversão de coordenadas no ecrã do utilizador, existia um passo *"modular inversion"* que não era efetuado em tempo constante *O(1)*, permitindo introduzir um ataque do tipo *"Timing attack"*, em que o atacante se aproveita para comprometer um sistema através da análise de complexidade dos algoritmos utilizados e projeta tempos lógicos de ataque.

[8] https://nvd.nist.gov/vuln/detail/CVE-2020-15671
[9] https://nvd.nist.gov/vuln/detail/CVE-2020-15664
[10] https://nvd.nist.gov/vuln/detail/CVE-2020-12400

– Métricas: Na escala *CVSS 3.1* o *score* é médio (4.7). O vetor de ataque é local, ou seja, o atacante tem, de alguma maneira, um acesso priviligiado ao sistema, no entanto, a complexidade do ataque é muito elevada, exige um conhecimento profundo do *software* usado. Este ataque pode levar a exposições de informação confidencial e portanto essa métrica é muito afetada.

Q4: *"Recorrendo ao CWE, descreva dois tipos comuns de problemas relacionados com integridade de dados identificados no desenvolvimento de software."*

R: A **CWE** (*Common Weakness Enumeration*)(6) disponibiliza uma categorização de fragilidades e vulnerabilidades presentes em *Software*. Na secção de *software development*[11], podemos ver que temos uma série de problemas reportados na imagem seguinte:

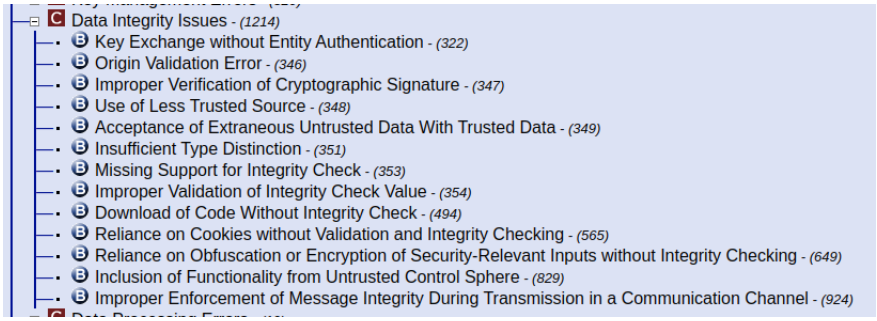

Fig. 9. Problemas de integridade de dados no desenvolvimento de *software*.

Eis dois tipos de problemas relacionados com a temática em questão:

1. *Key Exchange without Entity Authentication*

   Esta situação acontece quando o *software* faz uma troca de chaves criptográficas sem verificar a identidade do terceiro, isto é, quando se utiliza estes algoritmos de criptografia de chaves tem-se em mente garantir a integridade dos dados encriptados com essas chaves, no entanto isto não é

[11] Problemas comuns: https://cwe.mitre.org/data/definitions/699.html

tudo, deve haver uma confirmação ou negociação inicial para garantir que o terceiro é quem diz que é.
A base desta vulnerabilidade permite a um servidor malicioso fazer-se passar por uma entidade confiável levando muitas vezes o cliente ignorar o processo de autenticação ou até uma possível falha da mesma. O ataque pode levar a exposição de dados através de *requests* mal intencionados.

2. *Download of Code Without Integrity Check*

   Neste caso o *software* transfere *scripts*, código aberto e executáveis, procedendo à execução dos mesmos, sem efetuar uma verificação de integridade da origem e do código em questão.
   Obviamente isto pode levar o atacante a executar código malicioso comprometendo o *host* original. Exemplos de ataques podem ser DNS *spoofing*, onde a cache de DNS é envenenada com entradas maliciosas.

Aproveito para referir que a base de dados do *CWE* vai mais longe, disponibilizando uma análise detalhada de onde estes problemas podem ser introduzidos (i.e., em que fazes de desenvolvimento de produtos de *software*) e até exemplos práticos, em Java, PHP, etc., de aplicação dos erros:

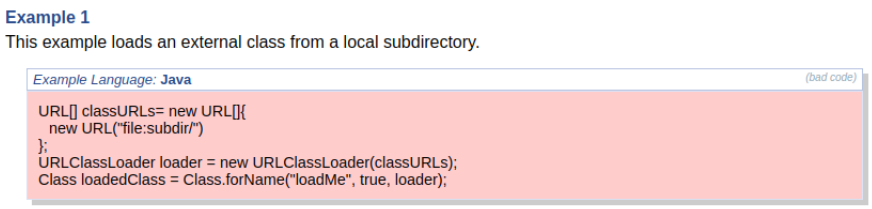

Fig. 10. Exemplo de código errado para o ponto 2.

References

[1] NIST. 2020. National Vulnerability Database (NVD). [online] Available at: https://www.nist.gov/programs-projects/national-vulnerability-database-nvd.

[2] Cvedetails.com. 2020. CVE Security Vulnerability Database. Security Vulnerabilities, Exploits, References And More. [online] Available at: https://www.cvedetails.com/.

[3] Exploit-db.com. 2020. Offensive Security'S Exploit Database Archive. [online] Available at: https://www.exploit-db.com/.

[4] Tools.ietf.org. 2020. Draft-Ietf-Tls-Dtls-Heartbeat-04 - Transport Layer Security (TLS) And Datagram Transport Layer Security (DTLS) Heartbeat Extension. [online] Available at: https://tools.ietf.org/html/draft-ietf-tls-dtls-heartbeat-04

[5] Mozilla. 2020. Security Vulnerabilities Fixed In Firefox For Android 80. [online] Available at: https://www.mozilla.org/en-US/security/advisories/mfsa2020-39/.

[6] Cwe.mitre.org. 2020. CWE - Common Weakness Enumeration. [online] Available at: https://cwe.mitre.org/.