Всероссийская олимпиада школьников по информационной безопасности

Содержание

1 forensic-a

Описание

Набор лог-файлов с псевдослучайными строками, среди которых присутствует одна «особая» запись.

Решение

В коллекции файлов необходимо идентифицировать запись, содержащую маркер формата `forensic{...}` либо метку строки `meta:`. Наиболее надёжна полнотекстовая выборка с регистронезависимым поиском и последующей экстракцией токена флага по регулярному выражению.

```
#!/usr/bin/env bash
set -euo pipefail
ROOT="forensic-a"
# 1) быстрый поиск строки с флагом
grep -R --line-number --text "forensic{" "$ROOT" || true
# 2) альтернатива через meta:
grep -R --line-number --text "^meta:" "$ROOT" || true
# 3) чистая экстракция токена флага (если встречается вхождение)
find "$ROOT" -type f -print0 \
 | xargs -0 grep -aHo "forensic{[0-9a-fA-F]\{64\}}" 2>/dev/null || true
```

2 forensic-b

Описание

PCAP со множеством TCP-сеансов и небольшими транзакциями, где полезная информация разбита на сегменты.

Решение

Сегменты флага передаются в полезной нагрузке TCP в виде маркеров `FILENAME=...;SEG=...` и/или строк `SEGMENT=...`. Метод: разобрать PCAP, извлечь полезные нагрузки Raw, выделить сегменты, упорядочить и склеить.

```
#!/usr/bin/env python3
import sys
from scapy.all import rdpcap, Raw, TCP

def main(pcap_path):
    pkts = rdpcap(pcap_path)
    segments = {}
    for p in pkts:
        if Raw in p and TCP in p:
            s = bytes(p[Raw].load).decode('utf-8', errors='ignore')
            if 'FILENAME=' in s and 'SEG=' in s:
```

```
                # Пример:
                "SMB_COM_TRANSACTION2:FILENAME=file_seg1.bin;SEG=abcd..."
                seg = None
                for part in s.split(';'):
                    if part.startswith('SEG='):
                        seg = part.split('=', 1)[1]
                        break
                idx = None
                if 'file_seg' in s:
                    try:
                        idx = int(s.split('file_seg',1)[1].split('.',1)[0])
                    except Exception:
                        pass
                segments[idx or (len(segments)+1)] = seg
            elif 'SEGMENT=' in s:
                for token in s.split():
                    if token.startswith('SEGMENT='):
                        segments[len(segments)+1] = token.split('=',1)[1]
    for k in sorted(segments):
        print(f"seg{k}: {segments[k]}")
    print("\nassembled:", ''.join(segments[k] for k in sorted(segments)))

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python3 solve_b.py forensic-b.pcap")
        sys.exit(1)
    main(sys.argv[1])
```

3 forensic-c

Описание

Шумный PCAP с вкраплением 64 меток вида `b<idx>:<bit_id>:<bit_seq>`, кодирующих битовую последовательность.

Решение

Извлечь `bit_id` по возрастанию индекса и собрать их в поток битов. Затем сгруппировать по 8 и интерпретировать как байты. Пригодно для восстановления ASCII/hex-представления ядра флага.

```
#!/usr/bin/env python3
import sys, re
from scapy.all import rdpcap, Raw

pat = re.compile(rb"b(\d+):([01]):([01])")

def main(pcap_path):
    pkts = rdpcap(pcap_path)
```

```
    found = {}
    for p in pkts:
        if Raw in p:
            m = pat.search(bytes(p[Raw].load))
            if m:
                idx = int(m.group(1))
                bit_id = int(m.group(2))
                found[idx] = bit_id
    bits = ''.join(str(found[i]) for i in sorted(found))
    print("bits:", bits)
    bts = []
    for i in range(0, len(bits), 8):
        byte = bits[i:i+8]
        if len(byte) == 8:
            bts.append(int(byte, 2))
    hexs = ''.join(f"{x:02x}" for x in bts)
    print("hex:", hexs)
    try:
        print("ascii:", bytes(bts).decode('utf-8'))
    except Exception:
        print("ascii decode failed")

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python3 solve_c.py forensic-c.pcap")
        sys.exit(1)
    main(sys.argv[1])
```

4 forensic-d

Описание

Один крупный текстовый отчёт с повторяющимися секциями; в одном месте присутствует строка с флагом между маркерами.

Решение

Искать прямое вхождение `forensic{` или выделить содержимое блока между служебными маркерами вроде `BEGIN-...` и `FIN-....`

```
#!/usr/bin/env bash
set -euo pipefail
FILE="forensic-d/report.txt"

echo "[grep] прямой поиск токена:"
grep -n -o "forensic{[0-9a-fA-F]\{64\}}" "$FILE" || true

echo
echo "[awk] содержимое между BEGIN- и FIN-:"
awk '/BEGIN-/{f=1;next} /FIN-/{f=0} f{print}' "$FILE" | grep -o
"forensic{[0-9a-fA-F]\{64\}}"
```

5 forensic-e

Описание

Флаг встречается в base64 внутри вложенного архива; достаточно распаковать и декодировать.

Решение

Нужно извлечь архив(ы), найти файл с расширением `.b64` и декодировать его содержимое: `base64 -decode`. Маршрут извлечения не важен, пока известен путь к архиву верхнего уровня.

```
#!/usr/bin/env bash
set -euo pipefail
ARCH="forensic-e/pack1.tgz"
WORK="$(mktemp -d)"
tar -xzf "$ARCH" -C "$WORK"
echo "[info] распаковано в: $WORK"
find "$WORK" -type f -name "*.b64" -print -exec sh -c 'echo "----- {} -----";
base64 --decode "{}"; echo' \;
```

6 forensic-f

Описание

Среди множества бинарных файлов один усечён до фиксированного размера, а в его начале записан флаг.

Решение

Определить файл размера ровно 512 байт и прочитать человекочитаемые строки/первые байты. Полезны `find -size 512c`, `strings`, `hexdump`.

```
#!/usr/bin/env bash
set -euo pipefail
DIR="forensic-f/storage"
find "$DIR" -type f -size 512c -print -exec sh -c '
  echo "---- {} ----"
  echo "[strings | head]"
  strings -n 4 "{}" | head -n 10
  echo "[hexdump | первые 16 строк]"
  hexdump -C "{}" | sed -n "1,16p"
' \;
```

7 forensic-g

Описание

Исходный текстовый файл был сжат и затем преобразован в шестнадцатеричный дамп; требуется восстановить и распаковать.

Решение

Конвертировать hex-дамп обратно в бинарный gzip (`xxd -r -p`) и распаковать (`gunzip -c`).

```
#!/usr/bin/env bash
set -euo pipefail
HEX="forensic-g/dump.hex"
OUTGZ="$(mktemp)"
xxd -r -p "$HEX" > "$OUTGZ"
gunzip -c "$OUTGZ" | sed -n '1,200p'
```

8 forensic-h

Описание

Крупный PCAP с разнотипным трафиком (DNS/HTTP/ICMP/FTP/TLS) и распределёнными сегментами флага.

Решение

Для DNS извлечь TXT-ответы по запросам `segN.*`; для HTTP — заголовок `X-Flag-Segment`; для ICMP — полезную нагрузку; для FTP — команды `RETR`; для TLS — контрольные признаки в клиентском приветствии. Затем собрать сегменты в порядке `seg1..seg4`.

```
#!/usr/bin/env python3
import sys, re
from collections import defaultdict
from scapy.all import rdpcap, DNS, DNSRR, TCP, UDP, Raw

def main(pcap_path):
    segs = defaultdict(list)
    pkts = rdpcap(pcap_path)

    for p in pkts:
        # DNS TXT
        if p.haslayer(DNS):
            d = p[DNS]
            if d.ancount:
                for i in range(d.ancount):
                    rr = d.an[i]
                    if isinstance(rr, DNSRR) and rr.type == 16:
```

```
                        name = rr.rrname.decode(errors='ignore') if
                        isinstance(rr.rrname, bytes) else str(rr.rrname)
                        m = re.search(r"seg(\d+)", name)
                        if m:
                            idx = int(m.group(1))
                            try:
                                txt = rr.rdata.decode()
                            except Exception:
                                txt = str(rr.rdata)
                            segs[idx].append(txt)

        # TCP/UDP Raw полезные нагрузки
        if Raw in p:
            s = bytes(p[Raw].load).decode('utf-8', errors='ignore')

            m = re.search(r"X-Flag-Segment:\s*([0-9a-fA-F]+)", s)
            if m:
                m2 = re.search(r"segment-(\d+)", s)
                idx = int(m2.group(1)) if m2 else len(segs)+1
                segs[idx].append(m.group(1))

            m = re.search(r"\bSEGMENT=([0-9a-fA-F]+)", s)
            if m:
                segs[len(segs)+1].append(m.group(1))

            m = re.search(r"RETR\s+forensic_seg(\d+)\.txt", s)
            if m:
                idx = int(m.group(1))
                segs[idx].append("RETR-marker")

    print("Сегменты по ключам:")
    for i in sorted(segs):
        print(i, "->", segs[i])
    assembled = ''.join(next((x for x in segs[i] if x), '') for i in
    sorted(segs))
    if assembled:
        print("\nСборка:", assembled)

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python3 solve_h.py forensic-h.pcap")
        sys.exit(1)
    main(sys.argv[1])
```

9 forensic-i

Описание

Во внутреннем смещении бинарного файла содержится строка флага, побайтно XOR-ированная константой.

Решение

Поскольку формат флага фиксированной длины (`forensic{[0-9a-f]{64}}`), достаточно выполнить скользящее окно указанной длины, применяя XOR с ключом и проверяя соответствие регулярному выражению.

```
#!/usr/bin/env python3
import sys, re
KEY = 0x55
L = 74  # 'forensic{'(9) + 64 hex + '}'(1)
rx = re.compile(rb"forensic\{[0-9a-f]{64}\}")

def main(path):
    data = open(path, "rb").read()
    for i in range(0, len(data) - L + 1):
        cand = bytes([b ^ KEY for b in data[i:i+L]])
        if rx.fullmatch(cand):
            print("offset:", i)
            print(cand.decode())
            return
    print("flag not found")

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python3 solve_i.py forensic-i/mystery.bin")
        sys.exit(1)
    main(sys.argv[1])
```