Tool-Assisted Security Code Review

Sherif Koussa
OWASP Ottawa

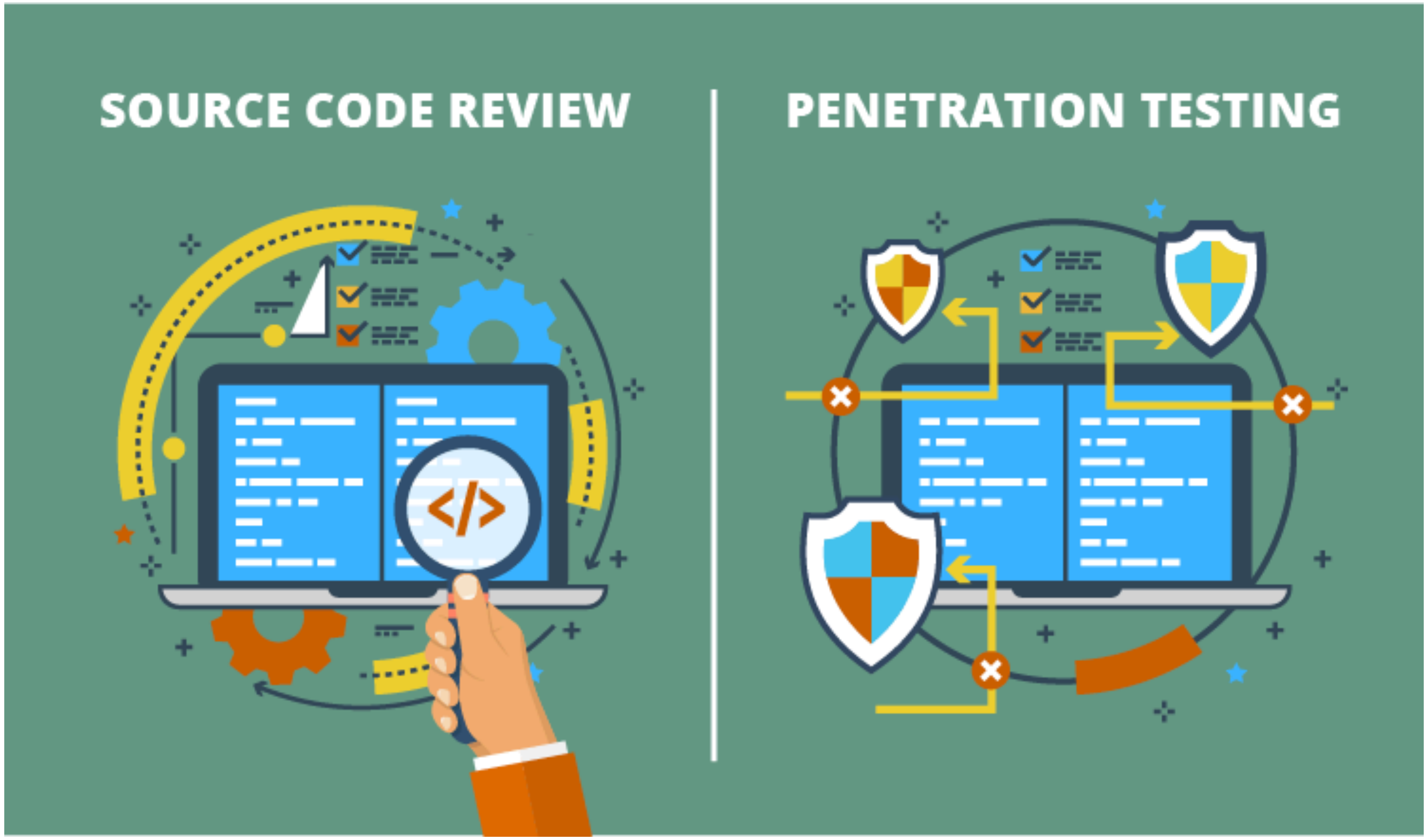
About Sherif

- Developer, hacker, and defender.
- WebGoat 5.0 Lead Developer
- SANS/GIAC Exam Consultant
- WASC SATEC Project Lead
- OWASP Cheat Sheet Ex Project Leader
- Founder and CEO @ Software Secured

What is Security Code Review

Security code review is the process of auditing the source code for an application to verify that the proper security controls are present, that they work as intended, and that they have been invoked in all the right places - OWASP

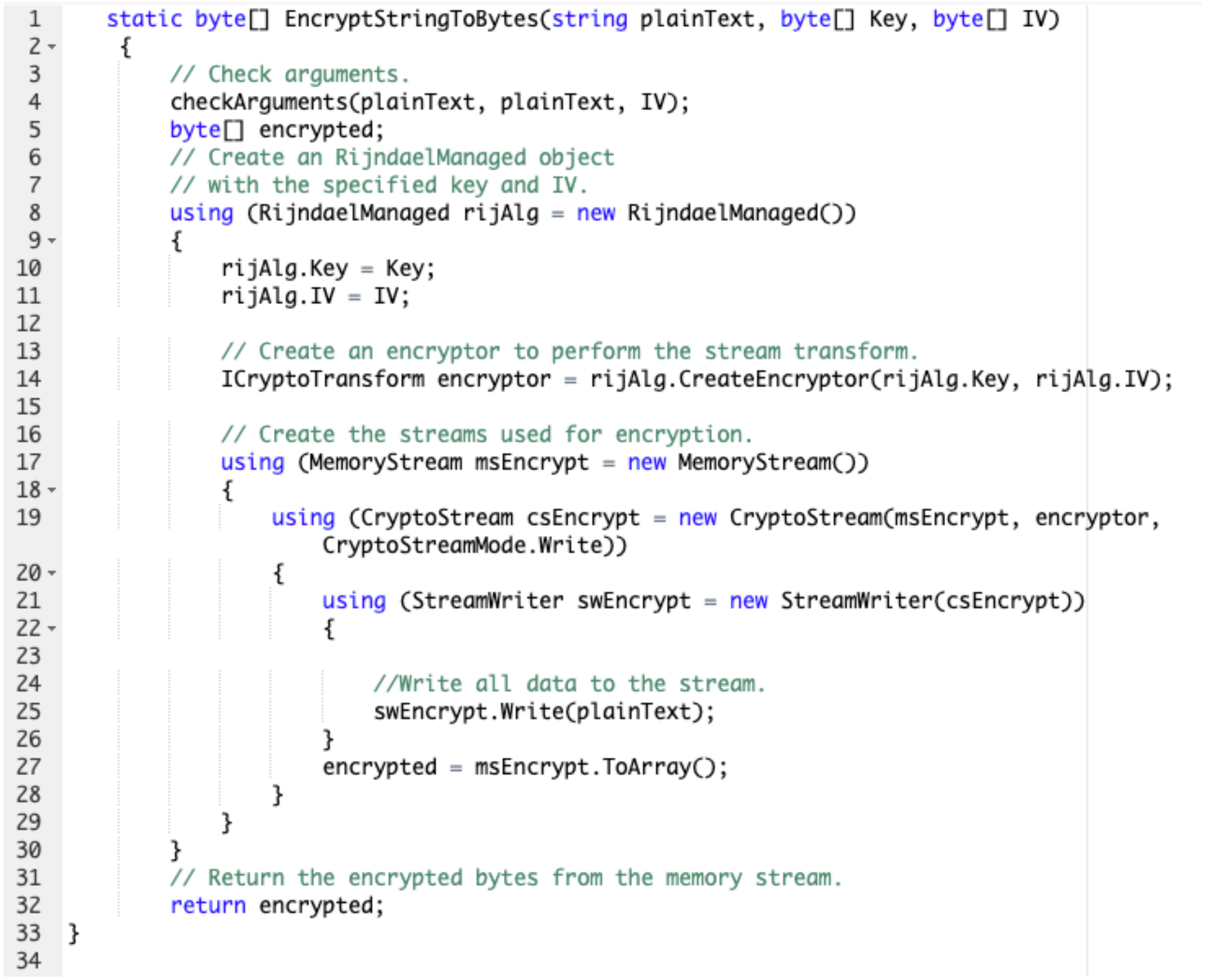
How is it different than Penetration Testing



Why Security Code Review

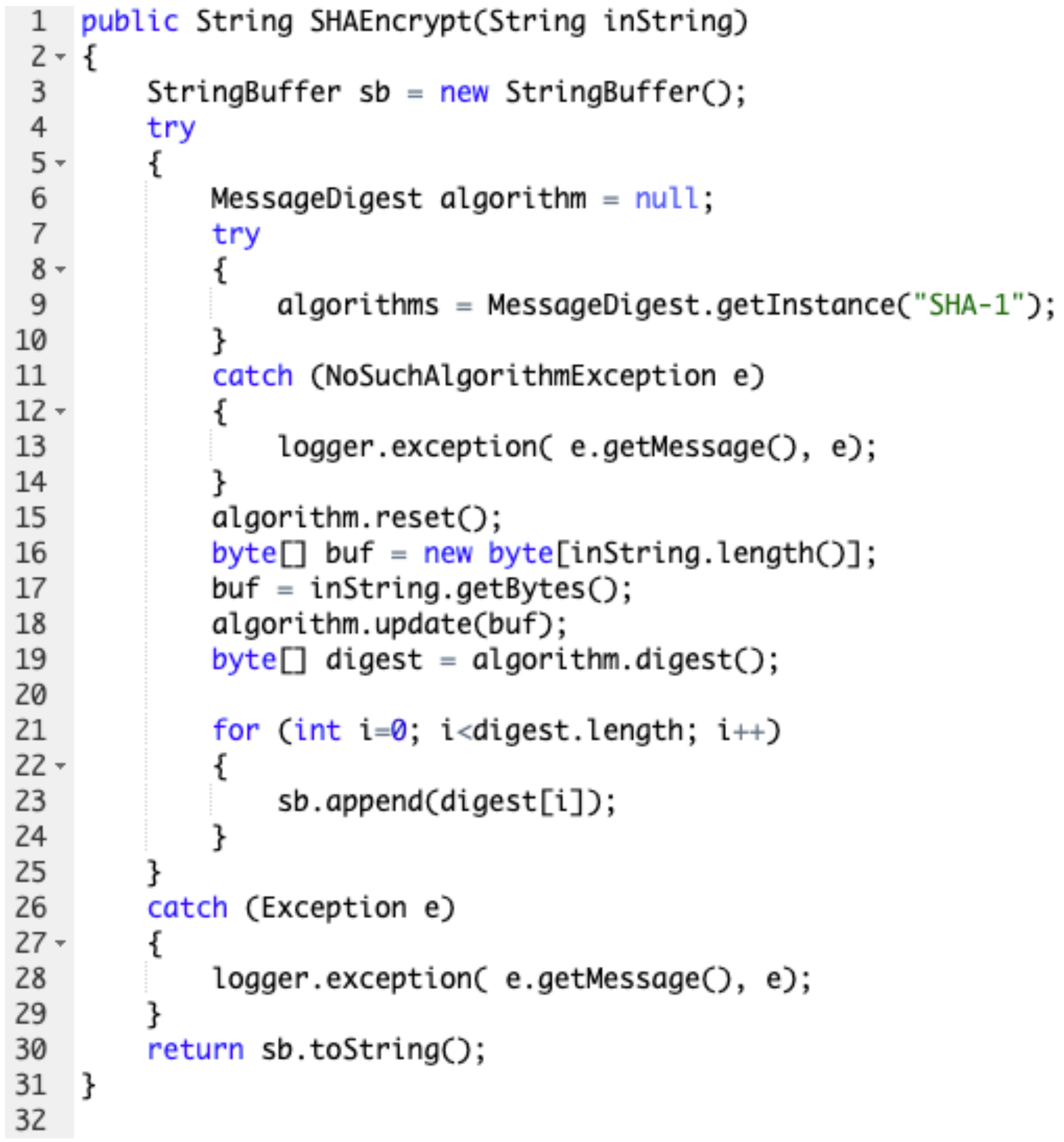
- Attack surface analysis
- Business logic issues
- Authentication & Authorization Issues
- Cryptography issues
- Second order injection issues

Why? Crypto bugs



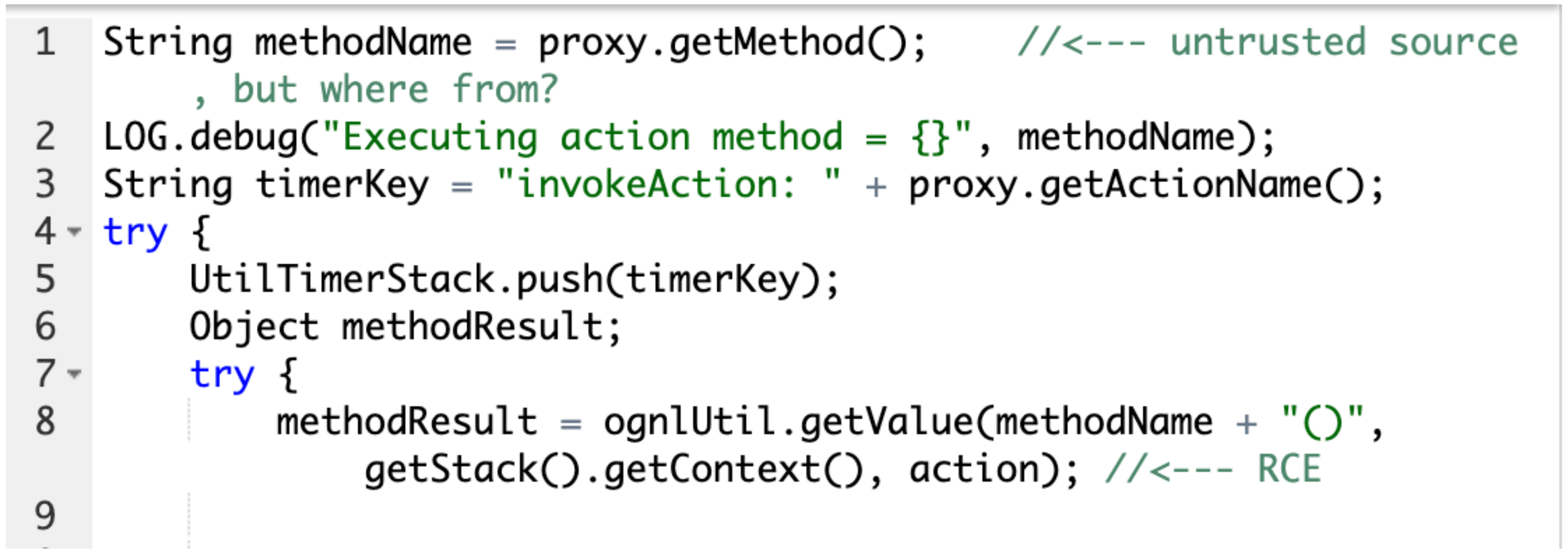
```
static byte[] EncryptStringToBytes(string plainText, byte[] Key, byte[] IV)
 {
    // Check arguments.
    checkArguments(plainText, plainText, IV);
    byte[] encrypted;
    // Create an RijndaelManaged object
    // with the specified key and IV.
    using (RijndaelManaged rijAlg = new RijndaelManaged())
    {
        rijAlg.Key = Key;
        rijAlg.IV = IV;

        // Create an encryptor to perform the stream transform.
        ICryptoTransform encryptor = rijAlg.CreateEncryptor(rijAlg.Key, rijAlg.IV);

        // Create the streams used for encryption.
        using (MemoryStream msEncrypt = new MemoryStream())
        {
            using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor,
                CryptoStreamMode.Write))
            {
                using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                {

                    //Write all data to the stream.
                    swEncrypt.Write(plainText);
                }
                encrypted = msEncrypt.ToArray();
            }
        }
    }
    // Return the encrypted bytes from the memory stream.
    return encrypted;
}

```

Why? Logic bugs



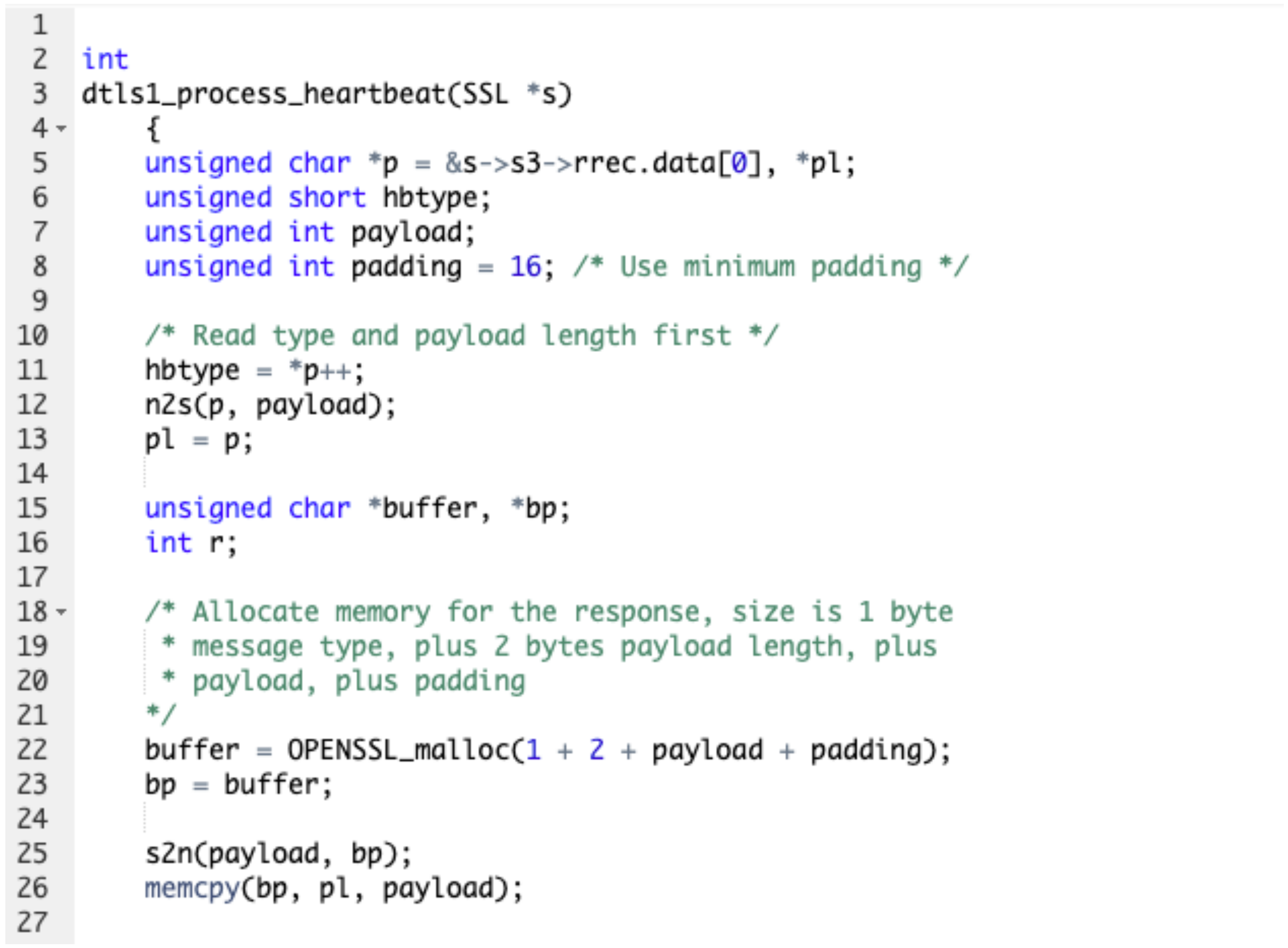
```java
public String SHAEncrypt(String inString)
{
    StringBuffer sb = new StringBuffer();
    try
    {
        MessageDigest algorithm = null;
        try
        {
            algorithms = MessageDigest.getInstance("SHA-1");
        }
        catch (NoSuchAlgorithmException e)
        {
            logger.exception( e.getMessage(), e);
        }
        algorithm.reset();
        byte[] buf = new byte[inString.length()];
        buf = inString.getBytes();
        algorithm.update(buf);
        byte[] digest = algorithm.digest();

        for (int i=0; i<digest.length; i++)
        {
            sb.append(digest[i]);
        }
    }
    catch (Exception e)
    {
        logger.exception( e.getMessage(), e);
    }
    return sb.toString();
}

```

Why? Framework related bugs



```java
String methodName = proxy.getMethod();    //<--- untrusted source, but where from?
LOG.debug("Executing action method = {}", methodName);
String timerKey = "invokeAction: " + proxy.getActionName();
try {
    UtilTimerStack.push(timerKey);
    Object methodResult;
    try {
        methodResult = ognlUtil.getValue(methodName + "()",
            getStack().getContext(), action); //<--- RCE

```

Why? Bugs in plain sight



```

int
dtls1_process_heartbeat(SSL *s)
    {
    unsigned char *p = &s->s3->rrec.data[0], *pl;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */

    /* Read type and payload length first */
    hbtype = *p++;
    n2s(p, payload);
    pl = p;

    unsigned char *buffer, *bp;
    int r;

    /* Allocate memory for the response, size is 1 byte
     * message type, plus 2 bytes payload length, plus
     * payload, plus padding
    */
    buffer = OPENSSL_malloc(1 + 2 + payload + padding);
    bp = buffer;

    s2n(payload, bp);
    memcpy(bp, pl, payload);

```

Why people don't do it?

- For the lack of one or more of the following:
  - Process: a scalable, repeatable and standard way of performing security code reviews.
  - Skills: the knowledge and experience necessary to perform the task.
  - Tools: static code analysis tool suffer from accuracy and recall.
  - Time: integration in the process and the amount of effort allocated to the activity.

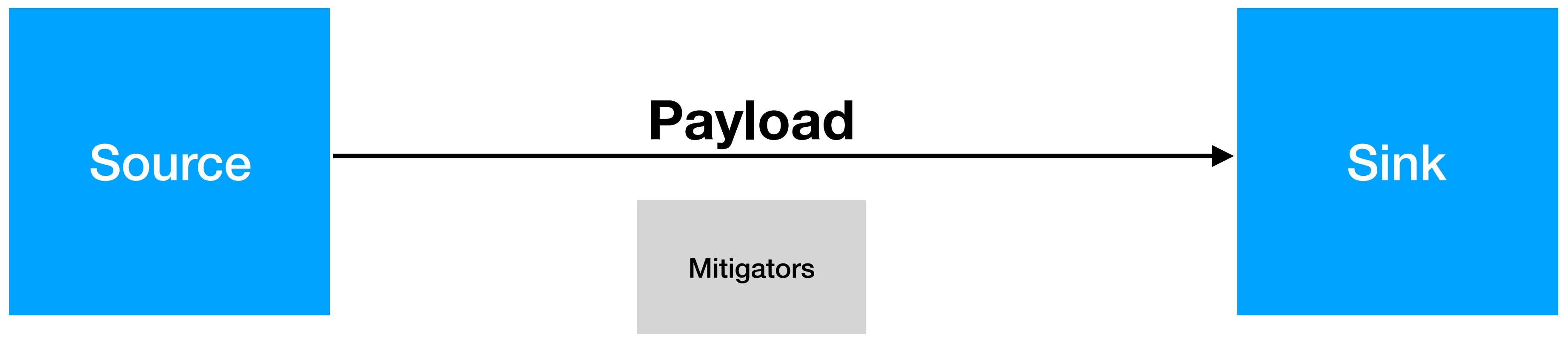
Security Code Review Types

- Ad-Hoc security code review
- Automated code review
- Peer-to-peer security code review
- Hybrid security code review
- Tool-assisted security code review

Tool-Assisted Security Code Review

Tool-assisted security code review is a type of code review that uses a static code analysis tool to guide the process

Simplified Process



- You have to wear two hats: attacker and mailman

Simplified Process
Your mission

- Source: how untrusted is this as a source
- Payload: what kind of payload do you need
- Sink: how easy is it to exploit that sink
- Mitigation controls: anything that can change the nature of the payloads

Skills

- You have to know what you are looking for.
- Easier in tool-assisted security code review.
- OWASP Top 10 is a very good way to start.

Tools

- Fortify
- Checkmarx
- Veracode
- AppScan Source
- Reshift
- FindSecBugs
- PMD
- Etc

Tools: things to watch for

- Language coverage
- False positive rates
- On-premise vs SaaS-based
- Integration into the dev process
- Integration into the DevOps pipeline

Let’s Scan Some Code

- Reshift is an online static code analysis tools:
  - Scans Java Code
  - PR Workflow
  - Adaptive accuracy calculation
  - Built-in integration into Dev and DevOps processes