



1. (a) Erklären Sie die Programmierrunde *Format String Problems (Bug)*. Bei welcher Art von Befehlen tritt diese auf? (3) 2.5
- (b) Geben Sie ein Beispiel in Codeform für einen Format String Bug an. (2) 2
- (c) Welche Konsequenzen kann ein Format String Bug haben? Nennen Sie eine mögliche Gegenmaßnahme. (3) 3

a) Dieser Bug tritt bei Befehlen auf, die einen String als Parameter erwarten, und dieser nicht geprüft wird, sondern es wird ihm blind vertraut. ✓

b) in C/C++:

`printf(buffer)` → würde in dieser Form den enthaltenen String interpretieren

richtig wäre: `printf("%s", buffer)` → behandelt buffer rein als String, egal, was drinstehen würde

c) Ungeprüfte Strings können auf das System einwirken, wenn sie z.B. Code-Stücke / Befehle beinhalten. *Format string Injection*  
Als Gegenmaßnahme eignet es sich, entsprechende Tokens wie den %s zu verwenden, um eine Interpretation / Ausführung zu verhindern.  
User-Input sollte immer geprüft werden.  
Whitelisting wäre eine weitere Maßnahme, sodass nur bestimmte, „saubere“ Strings akzeptiert werden.

Ad Frage:

2. Betrachten Sie den folgenden Source Code:

```
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[])
{
    char input[50];
    int input_lgth = strlen(argv[1]);
    int copy_lgth;
    if(input_lgth > 50)
    {
        copy_lgth = 50;
    }
    else
    {
        copy_lgth = input_lgth + 1;
    }
    strncpy(input, argv[1], copy_lgth);

    printf("%s", input);
    return 0;
}
```

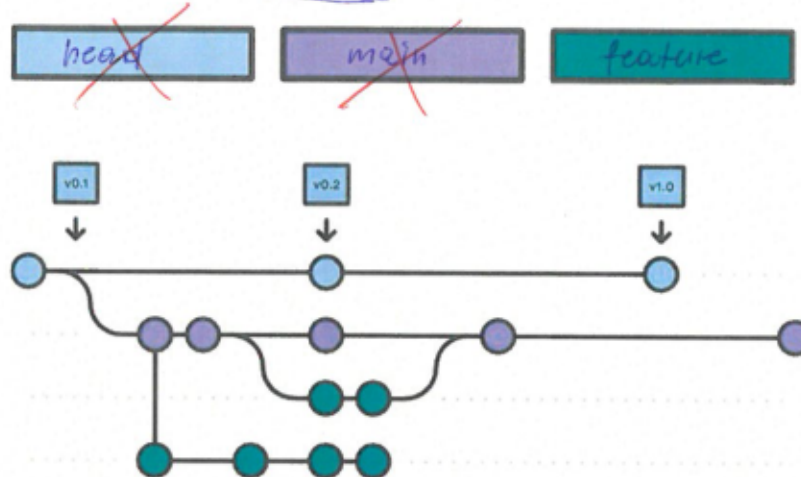
*wo kommt diese Variable her?*

*ich kenne leider  
keinen C/C++ Syntax*

- (a) Welche Sünde verbirgt sich in diesem Beispiel? Wo befindet sich diese? Gibt es Programmiersprachen, die nicht betroffen sind? Wenn ja, welche? (3)
- (b) Welche Konsequenzen verursacht diese Sünde? Geben Sie 2 konkrete Auswirkungen an. (3)
- (c) Wie kann diese verhindert werden? Korrigieren Sie den betreffenden Code Abschnitt. (3)

Ad Frage:

3. (a) Erklären Sie im Kontext von Git die Begriffe *Branches* und *Merge*. (2) 1,5  
 (b) Im folgenden sehen Sie einen Ausschnitt aus einer mit git-flow erstellten git History. Ordnen Sie den Farben den zugehörigen Branch Typ zu. (3) 1



- (c) Wie wird die Integrity in einem Git-Repository gewährleistet? (3) 0,5

a) ein branch ist eine abgetrennte Version vom main repository, in der gearbeitet werden kann, ohne dass es Auswirkung auf die „offizielle“ Version hat. ~~Im~~ Im Branch kann an features gearbeitet werden, die erst nach fertiger Implementierung wieder in die main eingepflegt werden.

ein merge bedeutet, dass ~~Code~~ Code, den ich z.B. vom remote repository geforkt habe, mit dem ~~Code~~ Code in meinem local directory zusammengeführt wird. Merge ist Zusammenfügen von zwei unterschiedlichen Versionen. *Gründung*

c) jede Version enthält die history, wer was wann geändert hat, sodass man Änderungen immer zurückverfolgen kann. *SHA-1!*

Ad Frage:



4. Die folgenden Fragen zielen auf das Security-Modell in Webbrowsern ab.


- (a) Nehmen wir an, in einem Webbrowser sind zwei Tabs offen. In einem steht in der Adressleiste `https://www.example.com:443/dir1/index.html`, und in der Adressleiste des anderen Tabs steht `https://example.com/dir2/other.html`. Werden die beiden Seiten als derselbe *Origin* betrachtet? (2) ☒
- (b) Beschreiben Sie den Unterschied zwischen impliziter und expliziter Authentifizierung bei Webapplikationen. (4) 1
- (c) Was bewirkt die Same-Origin Policy in Webbrowsern? (3) ☒
- (a) Dass beim Aufruf von Ressourcen fremder Seiten die HTTP-Antwort nicht z.B. per JavaScript wörtlich gelesen werden darf.  
☐ Wahr ☒ Falsch
- (b) Dass innerhalb einer Seite (Origin) clientseitig keinerlei HTTP-Requests zu fremden Seiten gesendet werden können. *→ gesendet können sie, werden nur nicht akzeptiert, wenn SOP es nicht erlaubt*  
☐ Wahr ☒ Falsch
- (c) Dass Webseiten keinerlei Ressourcen (JavaScript, CSS, etc.) von fremden Domänen einbinden dürfen.  
☒ Wahr ☐ Falsch
- (d) Dass *Cross-site Request Forgery*-Angriffe vollständig verhindert werden.  
☒ Wahr ☒ Falsch

- a) <sup>ja</sup> ~~nein~~, weil *o* sich "origin" aus der Domäne, Protokoll und Port zusammensetzt; durch die explizite Vorgabe von 443 im ersten URL ändert sich nichts, da beide über https aufgerufen werden; die domain, Protokoll und port bleiben gleich.
- b) implizite Authentifizierung passiert automatisch über Software, z.B. über den Webbrowser;  
explizite passiert über einen Admin oder eine andere Stelle, die einen Zugang explizit erlaubt.



5. (a) Was ist der Unterschied zwischen Authentifizierung und Autorisierung? (3) 3  
(b) Was können Sie bei der Entwicklung einer Webapplikation gegen das Problem „unsichere direkte Objektreferenzen“ tun? Es gibt genau eine richtige Antwort. (2) 2  
(a) Die Zugehörigkeit des aufgerufenen Objektes zum/zur aktuell angemeldeten Benutzer\*In überprüfen.  
☒ Wahr ☐ Falsch  
(b) Bei jeder Anfrage überprüfen, ob das aktuelle Konto die entsprechende Rolle hat, um die Funktion aufzurufen.  
☐ Wahr ☒ Falsch  
(c) TLS für die Transportverschlüsselung einsetzen.  
☐ Wahr ☒ Falsch  
(d) Aufsteigende Objekt-IDs verwenden.  
☐ Wahr ☒ Falsch

2) Authentifizierung ist die Prüfung, ob jemand Zugang zu einer Resource (Website, Server...) bekommen darf.  
Autorisierung passiert nach der Authentifizierung. Sie prüft, was jemand in seiner Rolle tun darf.

6. (a) Welcher ist der Unterschied zwischen den Maßnahmen gegen Reflected Cross-Site Scripting und Stored Cross-Site Scripting? (3) 
- (a) Bei Stored XSS passiert die Ausgabekodierung in der Datenbank, bei Reflected XSS bei der Ausgabe.  
☐ Wahr ☒ Falsch
- (b) Bei Stored XSS ist die Ausgabekodierung unabhängig vom jeweiligen Ausgabekontext.  
☒ Wahr ☐ Falsch
- (c) Es gibt keinen grundsätzlichen Unterschied.  
☐ Wahr ☒ Falsch
- (d) Whitelisting von Eingabeparametern ist bei Stored XSS wirkungslos.  
☐ Wahr ☒ Falsch

7. (a) Wie funktionieren, generisch gesprochen, Injection-Angriffe, und zwar unabhängig von der Technologie (SQL, STMP, LDAP, etc.)? (3) **2**
- (b) Nehmen wir an, es gebe eine OS-Command-Injection-Lücke in einer Webapplikation, die erfolgreich ausgenutzt wird. Im Kontext welches Betriebssystembenutzers werden die injizierten Kommandos *allgemein gesprochen* ausgeführt? (2) **2**
- (a) Mit dem root-Benutzer.  
☐ Wahr ☒ Falsch
- (b) Als privilegierter Betriebssystembenutzer.  
☐ Wahr ☒ Falsch
- (c) Als jener Benutzer, unter dem der Webserver bzw. Applikationsserver läuft.  
☒ Wahr ☐ Falsch
- (d) Als nichtprivilegierter Benutzer.  
☐ Wahr ☒ Falsch

2) Bei Injection-Angriffen werden Parameter vom Angreifer so manipuliert, dass sie nach Eintritt in die Applikation oder Server ein unerwünschtes Verhalten hervorrufen. Diese Injectionen werden meist in einem "legitimen" Kontext eingeschleust.